2003

Prescription Express System

Chia-Yu Tsai

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd-project

Part of the Software Engineering Commons

Recommended Citation

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
PRESCRIPTION EXPRESS SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Chia-Yu Tsai
December 2003
PRESCRIPTION EXPRESS SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Chia-Yu Tsai
December 2003

Approved by:

Richard Botting, Chair, Computer Science

Kerstin Voigt

Ernesto Gomez

Date 10/24/03
ABSTRACT

Pocket PC handheld computers are particularly useful for retrieving and storing information for people who are on the move and have busy fast changing schedules. Contacts, upcoming appointments, and brief notes are readily available. Like other professions in the world, there is a need to have a software system to give hospital doctors easy access to up-to-date operating information via wireless device.

The Prescription Express (PE) system is a software package that is developed with the express service of health care in mind. The purpose of this software package is to provide an environment in which doctors can easily checking on the upcoming appointments with patients and convenient to make prescription via wireless network or Internet/Intranet; nurses can maintain patient information, appointment, and the status of the prescription order; pharmacists can convenient to receive via Internet/Intranet and begin to fill up the drug. Utilizing the Microsoft eMbedded Visual tools to create application and system components for the Pocket PC and its wireless connection. In addition, utilizing the object oriented Java language to develop Java Server Pages and Servlets for Internet/Intranet connection. The PE system provides
the doctors, nurses, and pharmacist an effective method of communication.

By switching over to computerized documentation and wireless or online prescription processing, the clinical practice will be more efficient and the patient will have more resting time for the healing. The PE system eliminates medical errors resulting from misinterpreted handwriting. In addition, the system will provide the patients with easy express services without carrying paper prescription and making multiple trips to the pharmacy department.
ACKNOWLEDGMENTS

First, I would like to express my special gratitude to my project advisor Dr. Botting, whose valuable guidance and suggestions makes my project a whole. And thanks to my two project committee professors Dr. Gomez and Dr. Voigt for their strong support in developing this project.

Second, I would like to express my appreciation to my family. Thanks to my parents, witnessing my interest in pursuing higher education and encouraging my motivation to achieve my “American Dream” in US. Their encouragement and patience make my education possible, for it was they who bought my first computer.

Lastly, I want to thank everyone else who has helped me with, shown me the way on, or given me the opportunity in the software development. From online newsgroups and servers to good friends at school, it takes a great environment to make a great programmer.
# TABLE OF CONTENTS

ABSTRACT ................................................ iii

ACKNOWLEDGMENTS ................................. v

LIST OF TABLES ........................................... ix

LIST OF FIGURES ......................................... x

CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction ............................................ 1

1.2 Purpose of the Project ............................. 5

1.3 Context of the Problem ......................... 7

1.4 Significance of the Project ..................... 9

1.5 Assumptions ........................................... 10

1.6 Limitations ........................................... 10

1.7 Definition of Terms ................................. 11

1.8 Organization of the Report ..................... 13

CHAPTER TWO: SOFTWARE DESIGN

2.1 Introduction ........................................... 14

2.2 System Design ........................................ 15

2.2.1 Overall Schema ................................ 15

2.2.2 State Diagram ................................... 16

2.2.3 Product Functions .............................. 18

2.3 System Environment ................................. 22

2.4 Database Design ..................................... 24

2.4.1 Database Entity Relationship .......... 26

2.4.2 Conceptual Database Model ............. 27

2.4.3 Logical Model Table Schema .............. 28
CHAPTER THREE: SOFTWARE QUALITY ASSURANCE

3.1 Introduction ........................................... 33
3.2 Unit Test Plan ........................................... 33
3.3 Integration Test Plan ................................. 35
3.4 System Test Plan ........................................ 37

CHAPTER FOUR: USERS MANUAL

4.1 Introduction ........................................... 39
4.2 Pocket Personal Computer (PC) User ................. 39
  4.2.1 Login Screen ....................................... 39
  4.2.2 Appointment Screen .............................. 40
  4.2.3 Prescription Ordered List Screen .............. 41
  4.2.4 Prescription Form Screen ...................... 42
  4.2.5 Drug List Screen .................................. 43
  4.2.6 Exit Screen ........................................ 44
4.3 Web Browser User ...................................... 44
  4.3.1 Home Page ........................................ 44
  4.3.2 Login Page ......................................... 45
  4.3.3 Doctor Main Menu Page ......................... 46
  4.3.4 View Drug Information Page .................... 47
  4.3.5 Doctor View Appointment Page .................. 47
  4.3.6 Doctor View Patient Information Page ........ 48
  4.3.7 Doctor Make Prescription Page ................. 49
  4.3.8 Doctor View Prescription Records Page ........ 50
  4.3.9 Nurse Main Menu Page ........................... 51
5.1 Summary ........................................ 61
5.2 Concluding Remarks ............................ 62
APPENDIX A: SAMPLE PRESCRIPTION FORM .......................... 64
APPENDIX B: PROJECT SOURCE CODE ............................. 66
REFERENCES ........................................ 190
LIST OF TABLES

Table 1. Definition of Terms .................................. 12
Table 2. Database Dictionary for Table
        EMPLOYEE ........................................... 28
Table 3. Database Dictionary for Table PATIENT ........ 29
Table 4. Database Dictionary for Table EXAM .......... 30
Table 5. Database Dictionary for Table
        PRESCRIPTION ....................................... 31
Table 6. Database Dictionary for Table DRUG .......... 32
Table 7. Database Dictionary for Table
        APPOINTMENT ....................................... 32
Table 8. Unit Testing for Web User .................. 34
Table 9. Unit Testing for Personal Digital
        Assistant (PDA) User .............................. 35
Table 10. Integration Test for Web User ............. 36
Table 11. Integration Test for PDA User .............. 37
Table 12. System Testing ................................. 38
LIST OF FIGURES

Figure 1. Prescription Express (PE) Architecture Diagram .................. 16
Figure 2. State Diagram for the Patients .................. 17
Figure 3. State Diagram for the Prescription .................. 17
Figure 4. Use Case Diagrams .................. 18
Figure 5. Entity Relationship Diagram .................. 26
Figure 6. Conceptual Model Diagram .................. 27
Figure 7. Pocket Personal Computer (PC) Login Screen .................. 40
Figure 8. Pocket PC Appointment Screen .................. 41
Figure 9. Pocket PC Prescription Order List Screen .................. 42
Figure 10. Pocket PC Prescription Foam Screen .................. 43
Figure 11. Pocket PC Drug List Screen .................. 44
Figure 12. Home Page .................. 45
Figure 13. Login Page .................. 46
Figure 14. Doctor Main Page .................. 46
Figure 15. View Drug Information Page .................. 47
Figure 16. Doctor View Appointment Page .................. 48
Figure 17. Patient Information Page .................. 49
Figure 18. Prescription Foam Page .................. 50
Figure 19. Prescription Records .................. 51
Figure 20. Nurse Main Menu Page .................. 51
Figure 21. Create New Patient Page .................. 52
Figure 22. Update Patient Information Page .................. 53
Figure 23. Create New Appointment Page ............... 53
Figure 24. Update Appointment Page ................... 54
Figure 25. View Prescription Status Page ............... 55
Figure 26. Pharmacist Main Menu Page ................ 55
Figure 27. View Prescription List Page .................. 56
Figure 28. Update Prescription Status Page .......... 57
Figure 29. Change Refill Page ......................... 57
Figure 30. Administrator Main Page .................... 58
Figure 31. Create Employee Account Page ............ 59
Figure 32. Update Employee Account Page ............ 59
Figure 33. Change Password Page ..................... 60
Figure 34. Logout Page ................................ 60
CHAPTER ONE
SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

Pocket PC handheld computers are particularly useful for retrieving and storing information for people who are on the move and have busy fast changing schedules. As those small computers become more powerful with every generation, the possibly of their being really useful in a meaningful way is also increasing. Many analyzers suggest that more than 25% [13] and up to 40% [11] of all physicians in the US are using some form of a Personal Digital Assistant (PDA). The most recent 2002 HIMSS/AstraZeneca Clinician Wireless Survey [8] looked at how physician practices used information technology (IT) such as the use computer, handheld devices, electronic medical record and future purchasing decision, and found that 98% of hospital has Internet access, 72% of doctors use handheld devices, 70% of physicians used handhelds as a reference on pharmaceuticals before prescribing drugs for patients, 41% of doctors used handhelds for patient scheduling, 28% of hospitals use electronic medical records, and 21% of doctors exchange e-mail with their
patients. The number of physician who uses IT in the practice of medicine is increasing.

Many analyzers predicts that by 2004 more than half of all US physicians will be using some form of a handheld device for everything from scheduling to data collection, to prescribing to decision support [10]. This will make handheld devices one of the fastest growing technologies ever adopted by physicians. Currently, most physicians use PDA as a portable drug reference resource and scheduling. As the number of health related applications available for PDA has skyrocketed, there is a need to have a hospital software system which can provide hospital doctors easy access to keep appointment schedule, make prescription, and maintain up-to-date operating information via PDA.

This project presents the Prescription Express (PE) system which aims at the exploitation of the wireless network of Pocket PC and wired network of the server and the clients within different departments of a hospital. Within the PE system, the nurse will schedule daily patient appointment from the desktop computers at the nurse station. After the doctor exam the patients, instead of scrawling the prescription on a piece of paper, the doctor enters it into a computer or mobile device at the office. From there, it can be printed out on a local printer and
automatically routed to hospital pharmacy department, so the medicine is ready for the patient when they stop by.

The primary request of the doctor for use PDA is to answer questions that occur at the point-of-care. Due to the restrictions on size, memory and viewable screen, PDAs are limited in their ability to provide traditional content in traditional ways. Therefore, it is important to integrate the software for PDA with the familiar content and sources clinicians use daily [13]. Many companies have attempted use the power of the PDA in the emerging field of "Electronic Prescribing"; those companies include PocketScript, ePhysician ePad, and AllScripts [7]. Those three companies develop their systems somewhat differently. PocketScript and AllScripts both use Microsoft Pocket PC system hand-held device and implement a wireless local area network in the physician's office to move information from the hand-held device to a desktop computer server in the physician's office. Then the server is connected to the vendor's computer through a modem or other high-speed access device to get drug information updates. Later, the office computer also sends prescriptions directly to pharmacies either as an electronic message or by fax. However, ePhysician implemented on the Palm OS based system hand-held device and uses a personal computer in
the physician's office to load patient demographic information onto the hand-held using a docking cradle. Then the prescriptions must send directly to a central computer server at the vendor's offices, and then the Vendor's computer will sends prescriptions to the selected pharmacy, either electronically or by fax. Those three systems function in the environment where the doctor office and the pharmacy store are not in the same hospital. However, the PE system integrates the systems for the doctor office, nurse station, and pharmacy department for one same hospital. The PE system uses sophisticated encryption to protect the confidentiality of data as it is sent between computers in the same hospital.

The PE system that does nothing other than produce legible prescriptions in a shorter time than it would take to write one by hand can benefit many physicians. The PE system would provide an accurate record of previous prescriptions that allows the rapid renewal of multiple prescriptions is even more valuable. Another important potential benefit of an electronic prescribing system is fewer errors, including those caused by illegible handwriting, incorrect dosage selections, drug-drug or drug-disease interactions, and drug allergies. In 2000, the Institute of Medicine (IOM) published a report which
stunned the medical community [9]. The IOM report stated that medical errors are one of the nation's leading causes of death and injury and estimated that as many as 44,000 to 98,000 people die in U.S. hospitals each year as the result of medical errors. This means that more people die from medical errors than from motor vehicle accidents, breast cancer, or AIDS [12]. The good news is that many medical errors are preventable, such as the medical errors resulting from misinterpreted handwriting prescription. The PE system will eliminate the risk of medical misadventure due to illegible prescriptions by switching the paperwork to electronic medical records.

The contents of Chapter one presents an overview of the project. The contexts of the problem are discussed followed by the purpose, significance of the project, and assumptions. Next, the limitations that apply to the project are reviewed. Finally, definitions of terms are presented.

1.2 Purpose of the Project

The goal of this Master's projects is to build the PE system. The PE system is developed with the express service of health care in mind. The express service is mainly for outpatient with the need of behind counter
prescription care. Currently clinics still use a paper filing system for medication prescription that is relatively inefficient and takes up valuable storage space. In addition, most patients will need to go to the pharmacy store with the paper prescription to order and wait for their medication. This is relatively inconvenient and takes up valuable time. By switching over to computerized documentation and wireless or online prescription processing, the clinical practice will be more efficient and the patient will have more time for healing.

This system will enable the registered hospital doctors to write up a prescription via a handheld device. The prescription can be printed out and save in the central secured hospital server via a wireless network. In addition, the system can also transmit the prescription order to the hospital’s pharmacy store. The pharmacy store will immediately process the order and respond back to the medical doctor with a confirmation and the medication pickup time. This system eliminates medical errors resulting from misinterpreted handwriting. In addition, the system will provide the patients with easy express services without carrying paper prescription and making multiple trips to the pharmacy department.
The PE system will enable the medical doctors to share with the pharmacy store appropriate patient information such as allergies and insurance information. So the pharmacists can meet their obligation in dispensing medications that suits a patient; this will lower incidence of in-pharmacy drug switching. Information exchanged is solely for the purposes of professional medical and pharmacy practice.

1.3 Context of the Problem

There are two aspects of the problem encountered in the development of PE system. The first aspect related to the database management and the second relates to the security issue of communication and data transaction.  

First, given the number of registered hospital employee, patient, exam records, and prescription orders each day being maintained by the PE system, it is evident that careful database management must occur to ensure the appropriate utilization of the system resources to satisfy the objective and requirement of the daily hospital work loads. Further, the PE system must ensure a separate backup and recovery or maintenance functions for the database being undertaken. All contents and logs shall be generated dynamically and automatically so no human
interaction is needed. The server shall be up twenty-four hours a day and seven days a week, with exception that periodical system maintaining needs to be conducted depending on the reliability of the system server. The system should handle network packet loss smoothly. The system should not save inconsistent data or incomplete data into database.

Second, the issues related to the security issue. The Wireless communication security and the Pocket PC security have arisen recently.

Questions about cellular communication security go way back to the beginning of cell phones. Traditional Cellular communications use analog signals which are relatively easy to intercept and decode. However, the current digital wireless communications would solve all those problems. Because of the compressed nature of digital data and the improved encryption technologies, digital signals are much more difficult to be intercepted. Digital communication can access to secure private connections anywhere and anytime.

The web browser such as Internet Explorer supports accessing Web sites with 128-bit encryption using PCT (Private Communications Technology) and (Secure Sockets Layer) SSL protocols. While this sounds like it should
work great to access the web, but the Web browser must be
version 4.0 or later. Because Pocket Internet Explorer
only matches version 3.02, most secure Web sites do not
allow the Pocket PC users to access them. Hence, the PE
system must generate an application using the VPN service
on the web server and to ensure the security of access.

1.4 Significance of the Project

The significance of the project was the
implementation of wireless technology in today's fast
paced health care field. This project attempted to use the
power of the Pocket PC in the emerging field of electronic
prescribing. Instead of scrawling the prescription on a
piece of paper, the doctor enters the prescription order
into a Pocket PC at anywhere or a desktop at the office.
From there, the prescription can be printed out on a local
printer and automatically routed to the hospital. Among
other advantages, this project will reduce the risk of
medical misadventure due to illegible hand wrote
prescriptions.
1.5 Assumptions

The following assumptions were made regarding the project:

1. All users have experiences with Microsoft operation system.
2. All users have basic experience with Internet communication and browser usage.
3. All doctors have some experience with the usage of Pocket PC.

1.6 Limitations

During the development of the project, a number of limitations were noted. One of the limitations is that PDAs are limited in both memory and screen size. PDAs are designed to work with, rather than replace, laptop and desktop computers. PDAs generally do not have hard drives, but store programs and information in memory. This has the advantage that with a PDA, the programs do not need to be loaded into memory when the PDA turns on, but are instantly available. The disadvantage is that with current technology there is a limit to memory space, so the operating systems, programs and data are all sharply reduced from what might run on a full-sized computer.
Another limitation is that Microsoft includes a VPN client for connecting the Pocket PC to a VPN running on a Windows NT or Windows 2000 Server. Microsoft’s VPN client does not provide support for other servers or routers that offer VPN connectivity. Due to this limitation, the hospital server must running on a Windows NT or Windows 2000 Server.

The other limitation is that the Pocket PC user could not access the PE system through Pocket Internet Explorer. The Pocket Internet Explorer only matches version 3.02 which does not support 128-bit encryption. Due to this limitation, the PE system implements the VPN client component in the Pocket PC to access the VPN hospital server.

1.7 Definition of Terms

The following terms are defined as they apply to the project.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSUSB</td>
<td>California State University at San Bernardino</td>
</tr>
<tr>
<td>Database</td>
<td>A generic term for any system that has a collection of related information. Used in Computer Science fields to denote a compilation of data that when combined serves to give the user useful information.</td>
</tr>
<tr>
<td>Encryption</td>
<td>A method used to scramble sensitive information when transferring the information over an unsecured line. A decryption algorithm on the receiving side is needed to unscramble the message.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HDML</td>
<td>Handheld Device Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol, the client/server protocol that defines how messages are formatted and transmitted on the World Wide Web</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>MB</td>
<td>Megabytes</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PE</td>
<td>Prescription Express system</td>
</tr>
<tr>
<td>PCT</td>
<td>Microsoft Private Communications Technology protocols</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistants</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirements Specification</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer protocols</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>Windows CE</td>
<td>They are based on the Microsoft Windows operating system but are designed for including or embedding in mobile and other space-constrained devices. They are 32-bit multitasking, multithreading operating systems. They support SSL, VPN, 40- and 128 bit encryption, etc.</td>
</tr>
<tr>
<td>Pocket PC</td>
<td></td>
</tr>
<tr>
<td>Microsoft eMbedded Visual Basic 3.0</td>
<td>The Microsoft eMbedded Visual Basic® 3.0 integrated development environment (IDE) is the most productive way for developers to build applications for the next generation of Windows CE-based communication, entertainment, and information-access device. Uses well-known Visual Basic programming tools and techniques.</td>
</tr>
</tbody>
</table>
Microsoft eMbedded Visual C++ 3.0 is the most powerful way for developers to build applications for the next generation of Windows CE-based communication, entertainment, and information-access devices. This standalone integrated development environment (IDE) brings a new level of productivity to Windows CE development, without compromising flexibility, performance, or control.

WLAN
Wireless local area networks

802.11(b)
802.11 is a family of specifications for wireless local area networks (WLANs) developed by a working group of the Institute of Electrical and Electronics Engineers (IEEE). 802.11b uses the Ethernet protocol and CSMA/CA (carrier sense multiple access with collision avoidance) for path sharing. The 802.11b standard - often called Wi-Fi - is backward compatible with 802.11. The modulation method selected for 802.11b is known as complementary code keying (CCK), which allows higher data speeds and is less susceptible to multi-path propagation interference.

1.8 Organization of the Report

This report is divided into five chapters. Chapter one provides software requirements specification, an introduction to the context of the problem, purpose of the project, significance of the project, limitations, and definitions of terms. Chapter Two consists of the software design. Chapter Three documents the steps used in testing the project. Chapter Four presents the users manual from the project. Chapter Five presents conclusions drawn from the development of the project. The Appendices containing the project follows Chapter Five. Finally, the references for the project are presented.
CHAPTER TWO
SOFTWARE DESIGN

2.1 Introduction

The PE system consists three parts, one is the application for wireless handheld device, the Internet based hospital server, and desktop application for the reception counter and pharmacy department. The applications for wireless handheld device will consist of all available drugs carried by the hospital's pharmacy department, menu/list of appointments, patient information menu, drug reference, and prescription menu. However, the drug reference menu will not be developed in this project. There are quiet a few good drug reference software systems on the market. Every morning, the doctor will need to load the appointment list and patient information into his/her PDA with the cradle. After each examine, the patient prescription data will process to the hospital server from the wireless handheld device. All information that is exchanged with the hospital provider over the Internet is encrypted with 128-bit SSL technology. 128-bit SSL is the standard for commercial usage, which will provide the highest level of available encryption when sending and receiving data over the Internet and protects information
from interception and hacking. The implement of Pocket PC 2002 edition offers the power on password, usage of VPN (Virtual Private Network), and access Web sites that support 128-bit encryption or e-mail security. In addition to the application security measures of the handheld device, important information, such as patient names are always encrypted on the handheld device and the patient social security number will be substituted with arbitrary number. The main operations requirement for the hospital server is that it must be available 24 hours a day, 7 days per week. All actions are user initiated. Separate backup and recovery or maintenance functions are required as that is handled by system administration on the hospital hosting server machine. The database will also managed by the full time system administrator of hospital.

2.2 System Design

2.2.1 Overall Schema

The PE system can be divided into three parts. One is the application for handheld device, the other is the central secured hospital server, and the last one is the computers at the doctor office, nurse station, and pharmacy department. The hospital will be require to wired all its examine room with wireless access points, using
wireless Ethernet 802.11(b). By using the 802.11b wireless protocol, the wireless LAN modules can be attached to the PDAs to provide Internet connectivity to the hospital server.

Figure 1. Prescription Express (PE) Architecture Diagram

2.2.2 State Diagram

A State diagram is an analysis tool that can be used when a system or component of a system passes through a series of discrete states during operation. The following provides a state transaction diagram for the patients.
The following provides a state transaction diagram for the prescription flow.

Figure 2. State Diagram for the Patients

Figure 3. State Diagram for the Prescription
2.2.3 Product Functions

The following provides a Use Case Diagram that graphically depicts the users and principal functions of the PE. Further description of the functionality of the product is further described in section 2.2.3.1 to 2.2.3.4.

Figure 4. Use Case Diagrams
2.2.3.1 Functions for Doctors. The functions provided in the section are for all registered medical doctors who are an employee in the hospital. Further description of the functionality of the product for doctor is further described in section 2.2.3.1.1 to 2.2.3.1.5.

2.2.3.1.1 View Drug Information. Each doctor can use the system as a drug reference book. This function will assist to make a professional prescribing decision.

2.2.3.1.2 View Patient Information. Each doctor can view the patient information and medical history before making any professional prescribing decision.

2.2.3.1.3 View Prescription Orders. Each doctor can view the prescription orders that he or she made.

2.2.3.1.4 Update/Delete Prescription Orders. Each doctor can modify the prescription orders that he or she made while the prescription is not being process.

2.2.3.1.5 Make Prescription.

Step 1: Write up prescription

Doctor uses the PDA to write up a prescription scenario.

Step 2: Send prescription

Doctor sends the patient’s prescription to the hospital’s pharmacy department and orders the medication with the patient’s agreement.
Step 3: Print prescription

Doctor prints out the patient's prescription.

Step 4: Save prescription

System saves the patient's prescription to the central hospital secured server. The information will save in the patient's medical history. This will also assist the hospital in searching patients for any medication call back.

2.2.3.2 Functions for Pharmacists. The PE system provides functions to let the pharmacists to assist the patients with better and express service. Further description of the functionality of the product for pharmacist is further described in section 2.2.3.2.1 to 2.2.3.2.4

2.2.3.2.1 View Drug Information. Each pharmacist can use the system as a drug reference book. This function will assist to make a professional prescribing decision.

2.2.3.2.2 View Patient Information. Each pharmacist can view the patient information and medical history.

2.2.3.2.3 View Prescription Orders. Each pharmacist can view all prescription orders that are not been fulfilled.
2.2.3.2.4 Modified Prescription Order Status. The pharmacist will able to modify the prescription ordering status upon the completion.

2.2.3.3 Functions for Administrators. The functions provided in the section are for the administrator to create a secured network environment in the hospital. Further description of the functionality of the product for administrator is further described in section 2.2.3.3.1 to 2.2.3.3.2

2.2.3.3.1 Create User Account. The administrator will create accounts for all professional medical doctors who have the privileges and the license to write prescription and currently working in the hospital.

2.2.3.3.2 Update/Delete User Account. The administrator can modify user account.

2.2.3.4 Functions for Nurses. The functions provided in the section are for the nurses to assist patients. Further description of the functionality of the product for nurse is further described in section 2.2.3.4.1 to 2.2.3.4.5

2.2.3.4.1 View Prescription Status. The nurse will require to login in order to check the automatic scheduled medication pick-up time for each patient. The nurse can check the prescription ordering status for each patient.
2.2.3.4.2 Create New Patient Account. The nurse will be required to gather patient information and inputting them into the system.

2.2.3.4.3 Update/Delete Patient Information. The nurse will be able to update patient information.

2.2.3.4.4 Make Appointment. The nurse will be able to make appointment for the patient by schedule the date and time according to the doctor's office hours.

2.2.3.4.5 Update/Delete Appointment. The nurse will be able to update or delete appointment in the patient's request.

2.3 System Environment

The PE system uses following hardware. To use the PE system, the hospital server computer should meet or exceed the following requirements:

- Windows® compatible PC
- Pentium® IV 1.5 GHz (1 GHZ or better recommended)
- 256MB RAM minimum
- 150MB available hard disk space
- Windows® 2000 Server, or Windows NT® 4.0
- Microsoft Internet Explorer 5.5 or newer
- CD-ROM drive
• Printer
• 15" monitor with 800x600 resolution (or better)

To use the PE system, the handheld device should meet or exceed the following requirements:
• Approved Brands: Compaq, Hewlett-Packard, Casio, and others
• Operating System: Windows® CE pocketPC2002
• Memory: 64MB of memory is strongly recommended, however 32MB handheld can run portions of the PE system.

To use the PE system, the wireless connectivity between the handheld device and hospital server computer should meet or exceed the following requirements:
• Instant Wireless Network Access Point
• Fully IEEE 802.11a & 802.11b compatible
• High speed transfer rate up to 22 Mbps in 2.4 GHz or 72 Mbps in 5GHz
• A Range of operation of up to 1600 feet
• Hardware router and advanced firewall
• 128-bit Encryption
The PE system uses the following software for developing.

- Jakarta Apache Tomcat 4.0.3
- Microsoft Server 2000 Driver for JDBC
- JDK Development Kit 1.4.0
- Microsoft SQL Server 2000 Window CE Edition
- Microsoft SQL Server 2000 Enterprise Edition
- Microsoft eMbedded Visual Tools 3.0
- Microsoft Windows 2000 Server
- Internet Explorer web browser

2.4 Database Design

The database design of the PE system is developed and normalized based upon the prescription writing modules of Department of Pharmacology and Experimental Therapeutics Tufts University School of Medicine [5]. A hospital has many employee and patients. Each employee has a unique employee ID, user ID, and password. The EMPLOYEE table will store each employee's employee ID, user ID, password, name, phone number, address, department, employee type, and social security number. Each doctor has many patients. Each patient has a unique arbitrary ID number. The PATIENT table will store each patient's ID, name, phone number, social security number, email, address, sex, age, height,
weight, and medical history. The patient will be required
to make an appointment to see a doctor. The APPOINTMENT
table will consists of appointment's ID, time, date,
doctor's ID, and patient's ID. The doctor will meet and
examine the patient at the appointment time. Each EXAM
table will consist of doctor's ID, patient's ID, date,
exam notes, and exam's ID. Each exam will order
prescriptions. The PRESCRIPTION table will store the
prescription ID, drug name, drug dose and unit, quantity
to dispense, drug usage, substitution for other drugs,
ordered time, number of medication refill, the ordering
status, doctor's ID, patient's ID, exam's ID, and
pharmacist's ID. The DRUG table will consist of the most
commercial drugs in the market, their names, treatment,
indication information, units, favorite, and the
availability in pharmacy department of the hospital.
2.4.1 Database Entity Relationship

Figure 5. Entity Relationship Diagram
2.4.2 Conceptual Database Model

![Conceptual Model Diagram](image)

Figure 6. Conceptual Model Diagram
2.4.3 Logical Model Table Schema

The database dictionary of the PE system is presented in the following.

Table 2. Database Dictionary for Table EMPLOYEE

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Definition</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee_Id</td>
<td>Employee identification number</td>
<td>Int</td>
</tr>
<tr>
<td>User_Id</td>
<td>Employee User Name</td>
<td>Varchar</td>
</tr>
<tr>
<td>Password</td>
<td>Employee’s encrypted password</td>
<td>Varchar</td>
</tr>
<tr>
<td>Email</td>
<td>Employee’s mail address</td>
<td>Varchar</td>
</tr>
<tr>
<td>F_name</td>
<td>Employee’s First name</td>
<td>Varchar</td>
</tr>
<tr>
<td>M_name</td>
<td>Employee’s Middle name</td>
<td>Varchar</td>
</tr>
<tr>
<td>L_name</td>
<td>Employee’s Last name</td>
<td>Varchar</td>
</tr>
<tr>
<td>Dept</td>
<td>Department name where the employee work at</td>
<td>Varchar</td>
</tr>
<tr>
<td>Type</td>
<td>Employee’s job type</td>
<td>Varchar</td>
</tr>
<tr>
<td>Essn</td>
<td>Employee’s Social security number</td>
<td>Int</td>
</tr>
<tr>
<td>Phone</td>
<td>Employee’s Contact phone number</td>
<td>Int</td>
</tr>
<tr>
<td>Street</td>
<td>Street name of contact address</td>
<td>Varchar</td>
</tr>
<tr>
<td>City</td>
<td>City name of contact address</td>
<td>Varchar</td>
</tr>
<tr>
<td>State</td>
<td>State name of contact address</td>
<td>Varchar</td>
</tr>
<tr>
<td>Zip</td>
<td>Zip code of contact address</td>
<td>Varchar</td>
</tr>
</tbody>
</table>
### Table 3. Database Dictionary for Table PATIENT

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Definition</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient_Id</td>
<td>Patient identification number</td>
<td>Int</td>
</tr>
<tr>
<td>Email</td>
<td>Patient’s email address</td>
<td>Varchar</td>
</tr>
<tr>
<td>F_name</td>
<td>Patient’s first name</td>
<td>Varchar</td>
</tr>
<tr>
<td>M_name</td>
<td>Patient’s middle name</td>
<td>Varchar</td>
</tr>
<tr>
<td>L_name</td>
<td>Patient’s Last name</td>
<td>Varchar</td>
</tr>
<tr>
<td>Dob</td>
<td>Patient’s date of birth</td>
<td>Date</td>
</tr>
<tr>
<td>Ssn</td>
<td>Patient’s social security number</td>
<td>Int</td>
</tr>
<tr>
<td>Phone</td>
<td>Patient’s contact phone number</td>
<td>Int</td>
</tr>
<tr>
<td>Street</td>
<td>Street name of contact address</td>
<td>Varchar</td>
</tr>
<tr>
<td>City</td>
<td>City name of contact address</td>
<td>Varchar</td>
</tr>
<tr>
<td>State</td>
<td>State name of contact address</td>
<td>Varchar</td>
</tr>
<tr>
<td>Zip</td>
<td>Zip code of contact address</td>
<td>Varchar</td>
</tr>
<tr>
<td>Gender</td>
<td>Patient’s gender</td>
<td>Char</td>
</tr>
<tr>
<td>Weight</td>
<td>Patient’s weight</td>
<td>Varchar</td>
</tr>
<tr>
<td>Height</td>
<td>Patient’s height</td>
<td>Varchar</td>
</tr>
<tr>
<td>History</td>
<td>Patient’s medical history</td>
<td>Text</td>
</tr>
</tbody>
</table>
Table 4. Database Dictionary for Table EXAM

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Definition</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exam_Id</td>
<td>Appointment identification number</td>
<td>Int</td>
</tr>
<tr>
<td>Doctor</td>
<td>Employee identification number of the doctor who have this appointment</td>
<td>Int</td>
</tr>
<tr>
<td>Patient</td>
<td>Patient identification number of the patient who have this appointment</td>
<td>Int</td>
</tr>
<tr>
<td>Date</td>
<td>Date of ordered prescription</td>
<td>Date</td>
</tr>
<tr>
<td>Note</td>
<td>Notes of the Exam</td>
<td>Varchar</td>
</tr>
</tbody>
</table>
Table 5. Database Dictionary for Table PRESCRIPTION

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Definition</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prescription_Id</td>
<td>Prescription identification number</td>
<td>Int</td>
</tr>
<tr>
<td>Dispense</td>
<td>Drug dispense quantity</td>
<td>Int</td>
</tr>
<tr>
<td>Substitute</td>
<td>Drug substitute for generic brand</td>
<td>Char</td>
</tr>
<tr>
<td>Refill</td>
<td>Number of prescription refill</td>
<td>Int</td>
</tr>
<tr>
<td>Time</td>
<td>Time of ordered prescription</td>
<td>Time</td>
</tr>
<tr>
<td>Status</td>
<td>Status of order prescription</td>
<td>Varchar</td>
</tr>
<tr>
<td>Exam</td>
<td>Exam number</td>
<td>Int</td>
</tr>
<tr>
<td>Doctor</td>
<td>Employee identification number of the doctor who order the prescription</td>
<td>Int</td>
</tr>
<tr>
<td>Patient</td>
<td>Patient identification number of the patient who will receive the prescription</td>
<td>Int</td>
</tr>
<tr>
<td>Pharmacist</td>
<td>Employee identification number of the pharmacist who fill the prescription</td>
<td>Int</td>
</tr>
<tr>
<td>Drug</td>
<td>Drug identification number for the prescription</td>
<td>Int</td>
</tr>
</tbody>
</table>
Table 6. Database Dictionary for Table DRUG

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Definition</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drug_Id</td>
<td>Drug identification number</td>
<td>Int</td>
</tr>
<tr>
<td>Name</td>
<td>Drug name</td>
<td>Varchar</td>
</tr>
<tr>
<td>Unit</td>
<td>Drug dose</td>
<td>Varchar</td>
</tr>
<tr>
<td>Treatment</td>
<td>Drug usage</td>
<td>Varchar</td>
</tr>
<tr>
<td>Indication</td>
<td>Drug information (ie warming, directions, drug facts)</td>
<td>Varchar</td>
</tr>
<tr>
<td>Availability</td>
<td>Availability of the drug in the pharmacy department</td>
<td>Char</td>
</tr>
<tr>
<td>Favorite</td>
<td>Bookmarked favorite for the doctor</td>
<td>Char</td>
</tr>
</tbody>
</table>

Table 7. Database Dictionary for Table APPOINTMENT

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Definition</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appointment_Id</td>
<td>Appointment identification number</td>
<td>Int</td>
</tr>
<tr>
<td>Appointment_Date</td>
<td>Date of the appointment</td>
<td>Date</td>
</tr>
<tr>
<td>Appointment_Time</td>
<td>Time of the appointment</td>
<td>Time</td>
</tr>
<tr>
<td>Doctor</td>
<td>Employee identification number of the doctor who have this appointment</td>
<td>Int</td>
</tr>
<tr>
<td>Patient</td>
<td>Patient identification number of the patient who have this appointment</td>
<td>Int</td>
</tr>
</tbody>
</table>
CHAPTER THREE
SOFTWARE QUALITY ASSURANCE

3.1 Introduction
This chapter defines and controls the quality of the PE System by defining the techniques, procedures, processes and standards that will be used to develop the final prototype. The final prototype will be checked for conformance to the functional and performance requirements.

3.2 Unit Test Plan
The Unit Test Plan for the final PE System Prototype instructs a tester, what tests to perform on each module in the system. A module is a component of the system, which may be defined by a project to be implemented as a procedure, function, class, or program. The Unit Test Plan objective is to ensure that the particular module under test works properly and performs all the desired functions. Interaction between modules and overall system performance is not tested during this phase. The operation of each module function is tested and the results of the tests are summarized in Table.
<table>
<thead>
<tr>
<th>Unit Test</th>
<th>Test Performed</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>Tested for login into the system</td>
<td>OK</td>
</tr>
<tr>
<td>Logout</td>
<td>Tested for successful logout</td>
<td>OK</td>
</tr>
<tr>
<td>Change Password</td>
<td>Tested for change password in database</td>
<td>OK</td>
</tr>
<tr>
<td>Create new patient account</td>
<td>Tested for valid transaction in creating patient account in database</td>
<td>OK</td>
</tr>
<tr>
<td>Update patient account</td>
<td>Tested for update existing patient account in database</td>
<td>OK</td>
</tr>
<tr>
<td>Make appointment</td>
<td>Tested for valid transaction in creating appointment in database</td>
<td>OK</td>
</tr>
<tr>
<td>Update appointment</td>
<td>Tested for modify existing appointment in database</td>
<td>OK</td>
</tr>
<tr>
<td>View prescription status</td>
<td>Tested for displaying correct prescription status information on the browser</td>
<td>OK</td>
</tr>
<tr>
<td>View drug information</td>
<td>Tested for displaying correct drug information on the browser</td>
<td>OK</td>
</tr>
<tr>
<td>View patient information</td>
<td>Tested for display correct patient information</td>
<td>OK</td>
</tr>
<tr>
<td>View prescription list for doctor</td>
<td>Tested for display correct prescription orders that the user prescribed today.</td>
<td>OK</td>
</tr>
<tr>
<td>View prescription list for pharmacist</td>
<td>Tested for display correct prescription orders that are not been fulfilled.</td>
<td>OK</td>
</tr>
<tr>
<td>Make prescription</td>
<td>Tested for inserting correct values into the database</td>
<td>OK</td>
</tr>
<tr>
<td>Update prescription order</td>
<td>Tested for modify values in the database while the prescription order is not been processed</td>
<td>OK</td>
</tr>
<tr>
<td>Update prescription status</td>
<td>Tested for modify prescription status in the database</td>
<td>OK</td>
</tr>
<tr>
<td>Create New employee account</td>
<td>Tested for creating new employee account in the database</td>
<td>OK</td>
</tr>
<tr>
<td>Update employee account</td>
<td>Tested for update employee information in the database</td>
<td>OK</td>
</tr>
</tbody>
</table>
Table 9. Unit Testing for Personal Digital Assistant (PDA)

<table>
<thead>
<tr>
<th>Unit Test</th>
<th>Test Performed</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>Tested for login into the system</td>
<td>OK</td>
</tr>
<tr>
<td>Logout</td>
<td>Tested for successful logout</td>
<td>OK</td>
</tr>
<tr>
<td>View appointment</td>
<td>Tested for displaying correct appointment information on pda from the database</td>
<td>OK</td>
</tr>
<tr>
<td>View prescription list</td>
<td>Tested for displaying ordered prescription for each patient from the database</td>
<td>OK</td>
</tr>
<tr>
<td>Make prescription</td>
<td>Tested for inserting correct values into the database</td>
<td>OK</td>
</tr>
<tr>
<td>Update prescription order</td>
<td>Tested for modify values in the database while the prescription order is not been processed</td>
<td>OK</td>
</tr>
<tr>
<td>Bookmark favorite drug</td>
<td>Tested for book marking favorite drug in the database</td>
<td>OK</td>
</tr>
</tbody>
</table>

3.3 Integration Test Plan

Integration Test Plan for the final PE System

Prototype verifies that the software is functioning accordingly to the design as specified in chapter two. Each test is composed of a series of steps. Each step will be executed in sequence. The tester will initial or sign each step as it is completed. If the step is completed successfully, then it will be indicated as such in the test report. The operation of each series of steps is tested and the results of the tests are summarized in Table.
Table 10. Integration Test for Web User

<table>
<thead>
<tr>
<th>Unit Test</th>
<th>User Input</th>
<th>Expected Outcome</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make Prescription</td>
<td>Click the 'Make prescription' link</td>
<td>Link to 'Search Drug' page</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Input Drug name and click search bottom to find drug</td>
<td>Display drug within search condition</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Click the 'select' link</td>
<td>Display prescription form with selected drug</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Input patient id</td>
<td>Display patient id</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Select dispense quantity</td>
<td>Display dispense quantity</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Select substitute</td>
<td>Display 'yes' or 'no'</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Input refill number</td>
<td>Display refill number</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Click ‘Submit’ prescription order</td>
<td>Link to confirmation page</td>
<td>OK</td>
</tr>
</tbody>
</table>

36
### Table 11. Integration Test for PDA User

<table>
<thead>
<tr>
<th>Unit Test</th>
<th>User Input</th>
<th>Expected Outcome</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make Prescription</td>
<td>Click the appointment page</td>
<td>Display appointment list of the day</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Select patient from the appointment list</td>
<td>Link to the prescription form with patient id</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Click ‘whole’ or ‘favorite’</td>
<td>Link to the whole drug list or favorite drug list</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Click the drug from the list</td>
<td>Display ‘the drug is selected’</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Click ‘select’ bottom</td>
<td>Link back to the prescription form with the selected drug</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Select dispense quantity</td>
<td>Display dispense quantity</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Select substitute</td>
<td>Display ‘yes’ or ‘no’</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Input refill number</td>
<td>Display refill number</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Click ‘Save’ prescription order</td>
<td>Display the prescription order on prescription list</td>
<td>OK</td>
</tr>
</tbody>
</table>

### 3.4 System Test Plan

The system test plan for the final PE System Prototype will describe the test scenarios, test conditions and test cycles that must be performed to ensure system testing follows a precise schedule and that the system is thoroughly tested.
<table>
<thead>
<tr>
<th>Unit Test</th>
<th>Test performed</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup database system</td>
<td>Tested for making database backup file</td>
<td>OK</td>
</tr>
<tr>
<td>Change database system</td>
<td>Tested for changing user privileges. Tested for changing database name</td>
<td>OK</td>
</tr>
<tr>
<td>Log file for database connection pool</td>
<td>Tested for changing log file location</td>
<td>OK</td>
</tr>
<tr>
<td>Log file for system error</td>
<td>Tested for changing name and location of error log file</td>
<td>OK</td>
</tr>
<tr>
<td>Import database system</td>
<td>Tested for importing the database system from the backup file</td>
<td>OK</td>
</tr>
</tbody>
</table>
4.1 Introduction

Included in Chapter Four was a presentation of the user's manual of the project. The manual consists of two parts, one for Pocket PC user and one for web browser user.

4.2 Pocket Personal Computer (PC) User

4.2.1 Login Screen

The doctor will be shown a login screen, which will require the login name and encrypted password. This page will ensure the identification of the doctor for the security reason. Login into the PE system will allow the appointment screen, the prescription list screen, the prescription form screen, and the drug list screen to be activated.
4.2.2 Appointment Screen

This screen will show the list of login doctor’s appointments of the day. Upon clicking on the appointment, the prescriptions of the selected patient will be shown in the prescription list screen and the patient ID will be automatic loaded in the prescription form screen.
Figure 8. Pocket PC Appointment Screen

4.2.3 Prescription Ordered List Screen

The screen will show the list of login doctor's ordered prescription of the day for the selected patient. Upon clicking on the prescription, the complete prescription information will be automatic loaded in the prescription form screen for any further modification and update.
4.2.4 Prescription Form Screen

The screen will allow the doctor to order prescription. Upon clicking on the 'Whole List', the complete drug list will be shown; clicking on the 'Favorite', the complete favorite drug list will be shown. After selection of the drug, the doctor can select the dispense quantity, substitute, and the times of refill.
4.2.5 Drug List Screen

The screen will present the complete list of drug. The drug must be selected first, then the doctor can bookmark/delete/reset his/her favorite drug from the drug list.
Figure 11. Pocket PC Drug List Screen

4.2.6 Exit Screen

Clicking the exit screen will exit the program.

4.3 Web Browser User

4.3.1 Home Page

Home page is the main web page of the PE system, which allows employee to access to the login page by clicking the hyperlink.
4.3.2 Login Page

The hospital employee will be shown a login page, which will require the login name, and encrypted password. This page will ensure the identification of the employee for the security reason. After password validation, the employee will access to their individual type menu page.
4.3.3 Doctor Main Menu Page

The main menu provides doctor access to detailed pages via a menu.

Figure 14. Doctor Main Page
4.3.4 View Drug Information Page

This page allows the doctor/pharmacist to view selected drug information.

![View Drug Information Page](Image)

**Figure 15. View Drug Information Page**

4.3.5 Doctor View Appointment Page

This page will show the list of login doctor’s appointments of the day.
4.3.6 Doctor View Patient Information Page

This page will allow the doctor/pharmacist to view the selected Patient Information. This page will allow the doctor to search and select the individual patient. After the selection of the patient, the doctor will be able to access patient information by hyper linking to the 'View Patient Information Page'.
4.3.7 Doctor Make Prescription Page

This page will allow the doctor to make prescription. First, the doctor needs to select the medication. After the selection of the medication, the doctor will able to access prescription form by hyper linking 'select' to the Prescription Foam Page.
4.3.8 Doctor View Prescription Records Page

This page will allow the doctor to view all ordered prescriptions of the day. The doctor can further select the prescription and modifying the ordering by hyperlinking to the ‘Modify Prescription Record Page’.
4.3.9 Nurse Main Menu Page

The main menu provides nurse access to detailed pages via a menu.
4.3.10 Nurse Create New Patient Page

This page allows the nurse to create new patient.

![Create New Patient Page](image)

Figure 21. Create New Patient Page

4.3.11 Nurse Update Patient Information Page

This page will allow the nurse to search and select the individual patient. After the selection of the patient, the nurse will be able to access and update patient information by hyperlinking to the 'Update Patient Information Page'. Then nurse to update patient information.
4.3.12 Nurse Create New Appointment Page

This page will allow the nurse to search the filled appointment of the individual doctor and create new appointment.
4.3.13 Nurse Update Appointment Information Page

This page will allow the nurse to search and select the made appointment and update the made appointment information.

![Update Appointment Information](image)

Figure 24. Update Appointment Page

4.3.14 Nurse View Prescription Status Page

This page will allow the nurse to select and view the prescription status of individual patient.
4.3.15 Pharmacist Main Menu Page

The main menu provides pharmacist access to detailed pages via a menu.
4.3.16 **Pharmacist View Prescription List Page**

This page will allow the pharmacist to view all prescription orders which are not picked up by the patients. The pharmacist will be able to select and access and update the prescription status by hyper linking to the 'Update Prescription Status Page'.

![Image of the Prescription List Page]

**Figure 27. View Prescription List Page**

4.3.17 **Pharmacist Update Prescription Status Page**

This page will allow the pharmacist to update prescription status and input his/her employee identification number into the prescription order.
4.3.18 Pharmacist Change Refill Page

This page will allow the pharmacist select the prescription and refill prescription status.

Figure 28. Update Prescription Status Page

Figure 29. Change Refill Page
4.3.19 Administrator Main Menu Page

The main menu provides administrator access to detailed pages via a menu.

![Administrator Main Page](image)

Figure 30. Administrator Main Page

4.3.20 Administrator Create Employee Account Page

This page will allow the administrator to create new employee account.
4.3.21 Administrator Update Employee Account Page

This page will allow the administrator to search and select the individual employee account and update employee information.

Figure 31. Create Employee Account Page

Figure 32. Update Employee Account Page
4.3.22 Change Password Page

This page will allow the doctor, nurse, pharmacist, and administrator to modify his/her password.

![Change Password Page]

Figure 33. Change Password Page

4.3.23 Logout Page

The page shows the successful logout.

![Logout Page]

Figure 34. Logout Page
CHAPTER FIVE

CONCLUSIONS

5.1 Summary

Included in Chapter Five was a presentation of the conclusions gleamed as a result of completing the project. The Prescription Express System applies the newest technology to create a communication network between wireless network of Pocket PC devices and wired network of the hospital server; this communication network provides the functional, manageable and easy-to-use approach for all hospital employees. The PE system integrates the systems for the doctor office, nurse station, and pharmacy department in the same hospital. This system enables the registered hospital doctors to write up a prescription via a Pocket PC device or any networked hospital computer. The prescription can be printed out and save in the central secured hospital server via a wireless or wired network. The system will also automatic transmit the prescription order to the hospital’s pharmacy store in the patient’s request. Then the pharmacy store can immediately process the order and respond back to the medical doctor with a confirmation and the medication pickup time. This system eliminates medical errors resulting from misinterpreted
handwriting. In addition, the system will provide the patients with easy express services without carrying paper prescription and making multiple trips to the pharmacy department.

5.2 Concluding Remarks

Pocket PC along with properly structured content can positively affect how doctors care for their patients. The PE system serves as a model for clinical prescription practices; current prescription information awareness and convenient and secure prescription ordering can be pushed to the wireless handheld device where it is appropriate. And all of this can be integrated with the full power of wireless and web-based resources and technology. The current prototype of PE system is developed with the Microsoft visual development tool kits. All sources will be organized using development packages. This structure will aid in maintaining all modules organized and therefore maximizing maintenance facility. However, because of the limitations of Pocket PC Emulation device and the support of technologies, there are still many limitation encountered in software development.

The software design is extremely critical to usability. It is important to identify the functionality
that is available from the Pocket PC system before designing the software, the outcome of the system will be more likely to have high usability. Is there a mechanism for centralized groupware access? What about software management, data synchronization, and dynamic content? How is internal administrator support managed? All of those factors play important roles in the usability of PE system.

For the future development suggestion, the wireless side project will be able to integrate the system with different handheld devices. On the web based side of project, more features can be added to the client side; in such, the clients can check their medical history or refill status at the comfort of their home computer.

At the current stage of technology, PDAs are designed to work with, rather than replacing laptop and desktop computers. The implication of a Pocket PC application is to provide user to accomplish a specific task. As the demonstration of this project, the future looks bright for Pocket PC in the medical field. The Pocket PC will be used not only to help increasing the efficiency of physicians in practice, but also help to elevate the level of patient care.
APPENDIX A

SAMPLE PRESCRIPTION FORM
NEW ENGLAND PRESCRIPTION CENTER  
100 Pad St., Write-on-me, MA 02345  
Main #: (617) 123-4567  
Pharmacy #: (617) 123-4576

This prescription can be filled at the hospital pharmacy or at a community pharmacy.

<table>
<thead>
<tr>
<th>PATIENT NAME</th>
<th>AGE</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

NOT VALID FOR Schedule Two Controlled Substances. DO NOT REFILL AFTER ONE YEAR.

<table>
<thead>
<tr>
<th>DRUG</th>
<th>MG or ML</th>
<th>DIRECTIONS</th>
<th>QUANT.</th>
<th>REFILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

DEA #: 11  
PRESCRIBER? SIGNATURE: 12  
MD; IF NON-MD, SPECIFY: 13

NAME OF SUPERVISING PHYSICIAN IF PRESCRIBER IS NOT AN MD: 14  
PRESCRIBER? NAME (PRINT): 15

ORIGINAL  
H2-007-P (11/95)  
INTERCHANGE IS MANDATED UNLESS THE PRACTITIONER WRITES THE WORDS 'O SUBSTITUTION?' IN THE ABOVE SPACE.
Option Explicit

' setting the following properties the will assign default values for the session to be applied automatically

Const strInternetURLDflt = "http://localhost\sscesa\sscesa10.dll"
Const strServerNameDflt = "JARVIS-WEB"
Const strServerDatabaseNameDflt = "pe"
Const strLocalDatabase = "PE"
Private strLocalTableName As String
Private strPrescription As String
Private strPrescriptionUpdate As String
Private strExamDelete As String
Private strExam As String
Private gngPatientId As Long
Private gngDrugId As Long
Private strPickTable As String
Private strLocalTable As String
Private gngPatientId As Long
Private strLocalTable As String

Const strServerAgentDLL = "sscesa10.dll"
Const strSQLProvider = "provider=sqloledb"
Const strSQLProvider = "Provider=Microsoft.SQLServer.OLEDB.CE.1.0;data source=
Const strTitleBar = "Microsoft SQL Server CE"
Const tsvValue = -255
Const strPullTable = "pull_source"
Const strSQLSchemaTables = "SELECT TABLE.NAME FROM INFORMATION_SCHEMA.TABLES"
Const strSQLServerTables = "SELECT TABLE_NAME FROM pull_source"
Const strSQLDropPullSource = "DROP TABLE pull_source"

Dim cat As ADOXCE.Catalog
Dim rda As SSCE.RemoteDataAccess
Dim cn As ADOCE.Connection
Dim rx As ADOCE.Connection
Dim strRemoteConn As String
Dim bRDAPropsApplied As Boolean

Private Sub cmdAddFavorite_Click()
    AddDrugFavorite
End Sub

Private Sub cmdFavorite_Click()
    DgDrug.Rows = 1
    DgDrug.Clear
    PopulateDrugFavorite
    frmDrugList.Visible = True
End Sub

Private Sub cmdRemoveAllFavorite_Click()
    RemoveAllDrugFavorite
End Sub

Private Sub cmdRemoveFavorite_Click()
    RemoveDrugFavorite
End Sub

Private Sub cmdSelectedDrug_Click()
    AddDrug
    frmPrescriptionForm.Visiable = True
    frmDrugList.Visible = False
End Sub

Private Sub cmdWhole_Click()
    DgDrug.Rows = 1
    DgDrug.Clear
    PopulateDrugGrid
    frmDrugList.Visible = True
End Sub

* Initialize global session variables
Sub Form_Load()
    Set cn = Nothing
    Set rx = Nothing
    Set cat = Nothing
    Set rda = Nothing
    Dim i As Integer
    * setup the valid tab form
    frmLogin.Left = 0
    frmLogin.Top = 0
    frmPrescriptionForm.Left = 0
End Sub
Private Sub data_pull()
    close_connection
    If chk_rda_props = False Then
        MsgBox "Error: Missing one or more required RDA parameters.", vbCritical, strTitleBar
        Exit Sub
    End If
    On Error Resume Next
    rda.Pull "APPOINTMENT", "SELECT * FROM APPOINTMENT WHERE DOCTOR = (select employee_id from employee where user_id = " + txtUserld.Text + ") AND APPOINTMENT_DATE = " + CStr(Date) + ", strRemoteConn, TRACKING, "APPOINTMENT_Err"
    rda.Pull "PATIENT", "SELECT PATIENT_ID, L_NAME, F_NAME FROM PATIENT WHERE PATIENT_ID IN (SELECT PATIENT_ID FROM APPOINTMENT WHERE DOCTOR = (select employee_id from employee where user_id = " + txtUserld.Text + ")", strRemoteConn, TRACKING, "PATIENT_Err"
    rda.Pull "PRESCRIPTION", "SELECT * FROM PRESCRIPTION WHERE EXAM IN (select exam_id from exam where doctor = (select employee_id from employee where user_id = " + txtUserld.Text + ") AND EXAM_DATE = " + CStr(Date) + ", strRemoteConn, TRACKING, "PRESCRIPTION_Err"
    rda.Pull "EMPLOYEE", "SELECT EMPLOYEE_ID, USER_ID, PASSWORD, L_NAME, F_NAME FROM EMPLOYEE WHERE USER_ID = " + txtUserld.Text + ", strRemoteConn, TRACKING, "EMPLOYEE_Err"
    rda.Pull "EXAM", "SELECT * FROM EXAM WHERE DOCTOR = (select employee_id from employee where user_id = " + txtUserld.Text + ")", strRemoteConn, TRACKING, "EXAM_Err"
    If Err.Number <> 0 Then
        Call show_rda_errs
        close_connection
        Exit Sub
    End If
    On Error GoTo 0
    ' close the database so that other operations can continue
    close_connection
    MsgBox "Data loading completed", vbInformation, strTitleBar
    Exit Sub
End Sub

Private Sub cmdSavePrescription_Click()
    If strExamld = "Null" Then
        PopulateNewID
        saveExam
    Else
        savePrescription
    End If
End Sub
End Sub

Private Function saveExam() As Boolean
    If chk_rda_props = False Then
        MsgBox "Error: Missing one or more required RDA parameters.", vbCritical, strTitleBar
        Exit Function
    End If

' Submit SQL connects requires an exclusive connection to the database so we must close the current connection
close_connection
On Error Resume Next

' pass the command to the server
strExam = "SET identity_insert EXAM ON * + 
    'INSERT INTO EXAM (EXAM_ID, DOCTOR, PATIENT, DATE) VALUES (" + _
    " + CStr(strExamId) + ", ", " + _
    " + CStr(strEmployId) + ", ", " + _
    " + CStr(Total) + ", ", " + _
    " + CStr(Date) + ")" + _
    "SET identity_insert EXAM OFF"

rda.SubmitSQL strExam, strRemoteConn

' if an error occurred then display the RDA errors
If Err.Number <> 0 Then
    Call show_rda_err, strError, strRemoteConn
    close_connection
    Exit Function
End If
On Error GoTo 0

' close the database so that other operations can continue
close_connection
End Function

Private Function savePrescription() As Boolean
    If chk_rda_props = False Then
        MsgBox "Error: Missing one or more required RDA parameters.", vbCritical, strTitleBar
        Exit Function
    End If

' Submit SQL connects natively and requires an exclusive connection to the database so we must close the current connection
close_connection
On Error Resume Next

' pass the command to the server
strPrescription = "INSERT INTO PRESCRIPTION (RX_NAME, DOSE, DISPENSE, TREATMENT, SUBSTITUTE, STATUS, REFILL, EXAM, DOCTOR, PATIENT, TIME) VALUES (" + _
    " + CStr(strExamId) + ", ", " + _
    " + CStr(strEmployId) + ", ", " + _
    " + CStr(strExamId) + ", ", " + _
    " + CStr(strEmployId) + ", ", " + _
    " + CStr(Total) + ", ", " + _
    " + CStr(Date) + ")"

rda.SubmitSQL strPrescription, strRemoteConn

' if an error occurred then display the RDA errors
If Err.Number <> 0 Then
    Call show_rda_err, strError, strRemoteConn
    close_connection
    Exit Function
End If
On Error GoTo 0

' close the database so that other operations can continue
close_connection
MsgBox "Prescription is Saved", vbInformation, strTitleBar
End Function

' Apply all the sessions properties to the associated objects necessary for data connectivity. This routine will create a database with the specified local database name
Private Sub cmdApplyProperties_Click()
    ' we potentially will create a new database so close it now
close_connection

    ' Initialize the remote data access object
    Set rda = Nothing
    Set rda = CreateObject("SSCE.RemoteDataAccess.1.0")

    ' set properties on RDA object
    rda.LocalConnectionString = strSQLProvider + strLocalDatabase
    rda.InternetURL = strInternetURL
    ' tell RDA about IIS Server. Note that if DNS is not present, then IP address can be used.
    ' setup IIS login properties
    rda.InternetLogin = txtInternetLogin.Text
rda.InternetPassword = txtLoginPassword.Text

' build the connection string for the remote server.
strRemoteConn = strSQLProvider +
  "data source=" + strServerNameDflt +
  ";initial catalog=" + strServerDatabaseNameDflt +
  ";user id=" + "sa" +
  ";password=" + "3332727" + ";";

' indicate that the user has set RDA properties
bRDAPropsApplied = True
If FileSystem1.Dir(strLocalDatabase) = "" Then
  create_db
data_pull
Else
  FileSystem1.Kill (strLocalDatabase)
  create_db
data_pull
End If

InitializeGridVariables
grdAppointment.Rows = 1
grdAppointment.Clear
grdPrescription.Rows = 1
grdPrescription.Clear
grdDrug.Clear

' make sure we have an open connection
open_connection
Dim rs As ADOCE.Recordset
Dim fid As ADOCE.Field
Dim strLName As String
Set rs = Nothing

' execute the command text showing any errors may have occurred
On Error Resume Next
Set rs = cn.Execute("SELECT EMPLOYEE_ID, L_NAME FROM EMPLOYEE WHERE USER_ID='" + txtUserld.Text + " AND PASSWORD = " + txtPassword.Text + ");

' test for command errors
If cn.Errors.Count > 0 Then
  show_errs
  Exit Sub
End If

' test for row returning commands, if no rowset was returned
' then we have nothing to display
If chk_rowreturning(rs) = False Then
  MsgBox "There is no such doctor", vbInformation, strTitleBar
  Exit Sub
End If
On Error GoTo 0

strLName = rs.Fields("L_NAME").Value
strEmployeeId = rs.Fields("EMPLOYEE_ID").Value
PopulateAppointmentGrid
End Sub

' Check for all the required RDA properties.
Function chk_rda_props() As Boolean
  chk_rda_props = True
  If FileSystem1.Dir(strLocalDatabase) = "" And _
    bRDAPropsApplied = False Then
    chk_rda_props = False
  End If
End Function

' Returns true if local database properties
Function chk_local_props() As Boolean
  chk_local_props = True
  If FileSystem1.Dir(strLocalDatabase) = "" Then
    MsgBox "Error: No local database in use.", vbCritical, strTitleBar
    chk_local_props = False
  End If
End Function
Create a local database for use in the session.

Function create_db() As Boolean
Dim cat As ADOXCE.Catalog
Dim rslt As VbMsgBoxResult
Set cat = CreateObject("ADOXCE.Catalog.3.1")
create_db = False
On Error Resume Next
  ' use the filesystem control to detect if a database
  ' by this name already exists. If so delete it so
  ' that we can create a new database.
  If FileSystem1.Dir(strLocalDatabase) <> "" Then
    open_connection
  Else
    cat.Create strSQLEProvider + strLocalDatabase
  End If
  If Err.Number <> 0 Then
    MsgBox "Error: error loading local database. ", vbCritical, strTitleBar
    create_db = False
  Else
    create_db = True
  End If
Set cat = Nothing
End Function

Sub close_connection() On Error Resume Next
  cn.Close
  rx.Close
Set cn = Nothing
Set rx = Nothing
On Error GoTo 0
End Sub

Function open_connection() As Boolean
On Error Resume Next
  open_connection = True
  If cn Is Nothing Then
    Set cn = CreateObject("ADOXCE.Connection.3.1")
    cn.Open strSQLEProvider + strLocalDatabase
    If cn.Errors.Count > 0 Then
      Show_errs
      close_connection
      open_connection = False
    End If
  End If
  If rx Is Nothing Then
    Set rx = CreateObject("ADOXCE.Connection.3.1")
    rx.Open strSQLEProvider + "DRUG"
    If rx.Errors.Count > 0 Then
      close_connection
      open_connection = False
    End If
End If
End Function

Function chk_rowreturning(rs As ADOXCE.Recordset) As Boolean
On Error Resume Next
  Dim testnum As Integer
  testnum = rs.RecordCount
  If testnum = tstValue Then
    chk_rowreturning = False
  Else
    chk_rowreturning = True
  End If
On Error GoTo 0
End Function

Sub show_rda_errs() Dim strErr As String Dim rdaerr As Object For Each rdaerr In rda.ErrorRecords
  strErr = strErr & "Source: " & vbTab & vbTab & rdaerr.Source & vbCrLf
strErr = strErr & "Error number: " & vbTab & Trim(rdaerr.Number)) & vbCrLf
strErr = strErr & "Native Number: " & vbTab & Trim(rdaerr.NativeError) & vbCrLf
strErr = strErr & "Error Description: " & vbTab & rdaerr.Description
MsgBox strErr, vbCritical, strTitleBar
strErr = ""
Next rdaerr

On Error GoTo 0
Err.Number = 0
End Sub

' Loop through the connection object and display any provider specific error messages.
Sub show_errs()
    Dim strErr As String
    Dim terror As Integer
    Dim iparam As Integer
    Dim adoerr As ADOCE.Error
    For ierror = 0 To cn.Errors.Count - 1
        Set adoerr = cn.Errors(ierror)
        strErr = "desc = " & adoerr.Description & vbCrLf
        strErr = strErr & "number = " & Hex(adoerr.Number) & vbCrLf
        strErr = strErr & "nativeerror = " & adoerr.NativeError & vbCrLf
        For iparam = 0 To adoerr.ErrorParameters.Count - 1
            strErr = strErr & "param " & iparam & " = " & adoerr.ErrorParameters(iparam) & vbCrLf
        Next parm
        strErr = strErr & "source = " & adoerr.Source
        MsgBox strErr, vbCritical, strTitleBar
    Next ierror
End Sub

' Toggle to the appropriate tab section upon a user click event
Private Sub tabMain_Click()
    If tabMain.Tabs(1).Selected = True Then
        frmLogin.Visible = True
        frmPrescriptionForm.Visible = False
        frmAppointment.Visible = False
        frmPrescriptionOrder.Visible = False
        frmDrugList.Visible = False
        Exit Sub
    End If
    If tabMain.Tabs(2).Selected = True Then
        frmLogin.Visible = False
        frmPrescriptionForm.Visible = False
        frmAppointment.Visible = True
        frmPrescriptionOrder.Visible = False
        frmDrugList.Visible = False
        Exit Sub
    End If
    If tabMain.Tabs(3).Selected = True Then
        frmLogin.Visible = False
        frmPrescriptionForm.Visible = False
        frmAppointment.Visible = False
        frmPrescriptionOrder.Visible = True
        frmDrugList.Visible = False
        Exit Sub
    End If
    If tabMain.Tabs(4).Selected = True Then
        frmLogin.Visible = False
        frmPrescriptionForm.Visible = True
        frmAppointment.Visible = False
        frmPrescriptionOrder.Visible = False
        frmDrugList.Visible = False
        Exit Sub
    End If
    If tabMain.Tabs(5).Selected = True Then
        grdDrug.Clear
        PopulateDrugGrid
        frmDrugList.Visible = True
        frmLogin.Visible = False
        frmPrescriptionForm.Visible = False
        frmAppointment.Visible = False
        frmPrescriptionOrder.Visible = False
        Exit Sub
    End If
    If tabMain.Tabs(6).Selected = True Then
        App.End
        End If
    End If
End Sub

Private Sub cmdDBPwdHelp_Click()
    MsgBox "This is the password used for the PE Server database login.", vbInformation, strTitleBar
End Sub

Private Sub cmdDBUidHelp_Click()
MsgBox "This is the user id used for the PE Server database login.", vbInformation, strTitleBar
End Sub

Private Sub cmdLoginPwdHelp_Click()
    MsgBox "The password associated with the login user id for IIS authentication.", vbInformation, strTitleBar
End Sub

Private Sub cmdLoginUidHelp_Click()
    MsgBox "The name of the login account for IIS authentication.", vbInformation, strTitleBar
End Sub

Private Sub cmdRDAHelp_Click()
    MsgBox "The properties on this page are only required if you intend to use RDA to connect to a SQL Server from the 'Sync Tab'. * & _
    "If you do not supply valid properties here you will only be able to use local database operations from the 'Query Tab' * & _
    1 vblnformation, strTitleBar
End Sub

Sub PopulateAppointmentGrid()
    Dim rs As ADOCE.Recordset
    Dim fid As ADOCE.Field
    Dim strResult As String
    Dim irow, icol As Integer
    Set rs = Nothing
    ' execute the command text showing any errors may have occurred
    On Error Resume Next
    Set rs = cn.Execute("SELECT APPOINTMENT_TIME AS TIME, PATIENT_ID, F_NAME + "'" + L_NAME AS NAME FROM PATIENT INNER JOIN APPOINTMENT ON PATIENT.PATIENT_ID = APPOINTMENT.PATIENT_ORDER BY APPOINTMENT_TIME DESC")
    ' test for command errors
    If cn.Errors.Count > 0 Then
        show_errs
        Exit Sub
    End If
    ' test for row returning commands, if no rowset was returned
    ' then we have nothing to display
    If chk_rowreturning(rs) = False Then
        MsgBox "There is no appointment today", vbInformation, strTitleBar
        Exit Sub
    End If
    On Error GoTo O
    ' for all columns display column names into the grdAppointment control
    grdAppointment.Row = 0
    icol = 0
    For Each fld In rs.Fields
        grdAppointment.Col = icol
        grdAppointment.Text = rs(icol).Name
        icol = icol + 1
    Next fld
    ' for all rows and poke values into the grdAppointment control
    irow = 1
    Do While Not rs.EOF
        grdAppointment.AddItem **
        grdAppointment.Row = irow
        irow = irow + 1
        icol = 0
        For Each fld In rs.Fields
            grdAppointment.Col = icol
            If Not IsNull(fld.Value) Then
                grdAppointment.Text = fld.Value
            Else
                grdAppointment.Text = "Null"
            End If
            icol = icol + 1
        Next fld
        grdAppointment.RowData(grdAppointment.Rows - 1) = rs.Fields("PATIENT_ID").Value
        rs.MoveNext
    Loop
    End Sub

Function InitializeGridVariables() As Boolean
    On Error Resume Next
    ' set the return value to error condition
    InitializeGridVariables = False
    grdAppointment.ColWidth(0) = 1000
    grdAppointment.ColWidth(1) = 1000
    grdAppointment.ColWidth(2) = 1500
End Function
' check for errors, if none, return True
If Err.Number <> 0 Then
  InitializeGridVariables = True
End If
End Function

Private Sub grdAppointment_Click()
   On Error Resume Next
   MsgBox "Value:" & & grinAppointment.TextMatrix(grdAppointment.Row, grdAppointment.Col) , vbCrLf, strTitleBar
   MsgBox "Value:" & & grinAppointment.RowData(grdAppointment.Row) , vbCrLf, strTitleBar
   glngPatientId = grinAppointment.RowData(grdAppointment.Row)
   Me.TxtPatient.Text = glngPatientId
   grdPrescription.Rows = 1
   For Each fid In rs.Fields
      grdPrescription.Col = icol
      If Not IsNull(fld.Value) Then
         grdPrescription.Text = fld.Value
      Else
         grdPrescription.Text = "Null"
      End If
      icol = icol + 1
   Next fid
   grdPrescription.RowData(grdPrescription.Rows - 1) = rs.Fields("ID").Value
End Sub

Function PopulatePrescriptionGrid() As Boolean
   Dim rs As ADOCE.Recordset
   Dim fld As ADOCE.Field
   Dim strSQL As String
   Set rs = Nothing
   strSQL = "SELECT PRESCRIPTION_ID AS ID, RX_NAME AS Rx, DOSE, DISPENSE, TREATMENT AS USAGE, SUBSTITUTE, REFILL, STATUS FROM PRESCRIPTION WHERE PATIENT = " & CStr(glngPatientId)
   Set rs = cn.Execute(strSQL)
   If rs.Errors.Count > 0 Then
      ShowErrs
      Exit Function
   End If
   For Each fid In rs.Fields
      grdPrescription.Col = icol
      If Not IsNull(fld.Value) Then
         grdPrescription.Text = fld.Value
      Else
         grdPrescription.Text = "Null"
      End If
      icol = icol + 1
   Next fid
   grdPrescription.RowData(grdPrescription.Rows - 1) = rs.Fields("ID").Value
End Function
rs.MoveNext
Loop
End Function

Private Sub grdPrescription_Click()
On Error Resume Next
MsgBox Value: & grdPrescription.RowData(grdPrescription.Row), vbCrLf, strTitleBar
MsgBox Value: & grdPrescription.TextMatrix(grdPrescription.Row, grdPrescription.Col), vbCrLf, strTitleBar
' make sure we have an open connection
open_connection
RefreshFormPrescription
End Sub

Private Function RefreshFormPrescription() As Boolean
Dim rs As ADOCE.Recordset
Dim fid As ADOCE.Field
Dim strSQL
Dim irow, icol As Integer
Dim strSQL
Set rs = Nothing
' execute the command text showing any errors may have occurred
On Error Resume Next
strSQL = "SELECT * FROM PRESCRIPTION WHERE PRESCRIPTION_ID = " + CStr(strPrescriptionID) + ""
Set rs = cn.Execute(strSQL)
' test for command errors
If cn.Errors.Count > 0 Then
    show_errs,
    Exit Function
End If
' test for row returning commands. if no rowset was returned
' then we have nothing to display
If chk_rowreturning(rs) = False Then
    MsgBox "Operation complete", vbCrLf, strTitleBar
    Exit Function
End If
On Error GoTo 0
' do we have a Prescription ID?
If Not IsNull(rs.Fields("PRESCRIPTION_ID"), Value) Then
    ' set the name
    strPrescriptionID = rs.Fields("PRESCRIPTION_ID"), Value
End If
' do we have a RX Name?
If Not IsNull(rs.Fields("RX_NAME"), Value) Then
    ' set the name
    Me.TxtPrescriptionName.Text = rs.Fields("RX_NAME"), Value
End If
' do we have a RX Dose?
If Not IsNull(rs.Fields("DOSE"), Value) Then
    ' set the dose
    Me.TxtDose.Text = rs.Fields("DOSE"), Value
End If
' for dispense?
If Not IsNull(rs.Fields("DISPENSE"), Value) Then
    ' set the dispense
    Me.cboDispense.Text = rs.Fields("DISPENSE"), Value
End If
' for usage?
If Not IsNull(rs.Fields("TREATMENT"), Value) Then
    ' set the usage
    Me.TxtUsage.Text = rs.Fields("TREATMENT"), Value
End If
' for substitute?
If Not IsNull(rs.Fields("SUBSTITUTE"), Value) Then
    ' set the number
    Me.cboSubstitute.Text = rs.Fields("SUBSTITUTE"), Value
End If
' for refill?
If Not IsNull(rs.Fields("REFILL"), Value) = 0 Then
    ' set the refill number
    Me.cboRefill.Text = rs.Fields("REFILL"), Value
End If
' do we have a Patient ID?
If Not IsNull(rs.Fields("PATIENT").Value) Then
    * set the ID
    Me.TxtPatient.Text = rs.Fields("PATIENT").Value
End If
If Err.Number = 0 Then
    RefreshFormPrescription = True
End If
End Function

Private Sub cmdDeletePrescription_Click()
    If chk_rda_props = False Then
        MsgBox "Error: Missing one or more required RDA parameters.", vbCritical, strTitleBar
        Exit Sub
    End If
    close_connection
    On Error Resume Next
    * pass the command to the server
    strSQLDelete = "DELETE FROM PRESCRIPTION WHERE PRESCRIPTION_ID = " + CStr(strPrescriptionId)
    rda.SubmitSQL strSQLDelete, strRemoteConn
    If Err.Number <> 0 Then
        Call show_rda_errs
        close_connection
        MsgBox "Prescription is Deleted", vbInformation, strTitleBar
    End Sub
End Sub

Private Sub cmdUpdatePrescription_Click()
    If chk_rda_props = False Then
        MsgBox "Error: Missing one or more required RDA parameters.", vbCritical, strTitleBar
        Exit Sub
    End If
    close_connection
    On Error Resume Next
    * pass the command to the server
    strSQLUpdate = "UPDATE PRESCRIPTION SET " + _
    "RX_NAME = " + TxtPrescriptionName.Text + " , " + _
    "DOSE = " + TxtDose.Text + " , " + _
    "DISPENSE = " + cboDispense.Text.Text + " , " + _
    "TREATMENT = " + TxtUsage.Text.Text + " , " + _
    "SUBSTITUTE = " + cboSubstitute.Text.Text + " , " + _
    "REFILL = " + cboRefill.Text.Text + " , " + _
    "WHERE PRESCRIPTION_ID = " + CStr(strPrescriptionId)
    rda.SubmitSQL strSQLUpdate, strRemoteConn
    If Err.Number <> 0 Then
        Call show_rda_errs
        close_connection
        MsgBox "Prescription is Updated", vbInformation, strTitleBar
    End Sub
End Sub

Private Function PopulateExistingID() As Boolean
    Dim rs As ADOCE.Recordset
    Dim fid As ADOCE.Field
    Dim irow, icol As Integer
    Dim strSQL
    Set rs = Nothing
    * execute the command text showing any errors may have occurred
    On Error Resume Next

strSQL = "SELECT MAX(EXAM_ID) AS EXAM_ID FROM EXAM WHERE DATE = " + CStr(Date) + " AND PATIENT = " + CStr(glngPatientId)
Set rs = cn.Execute(strSQL)
* test for command errors
if cn.Errors.Count > 0 then
  show_errs
  Exit Function
End If
* set strExamId to the max + 1
if rs.Fields("EXAM_ID").Value <> Null then
  strExamId = rs.Fields("EXAM_ID").Value
Else
  strExamId = "1"
End If
* check for errors
if Err.Number > 0 then
  PopulateUsageText = True
End If
End Function
Private Function PopulateNewID() As Boolean
Dim rs As ADOCE.Recordset
Dim fid As ADOCE.Field
Dim strSQL As String
Dim irow, icol As Integer
Set rs = Nothing
On Error Resume Next
Set rs = cn.Execute("SELECT MAX(EXAM_ID) + 1) AS EXAM_ID FROM EXAM")
* test for command errors
if cn.Errors.Count > 0 then
  show_errs
  Exit Function
End If
* set strExamId to the max + 1
if rs.Fields("EXAM_ID").Value <> Null then
  strExamId = rs.Fields("EXAM_ID").Value
Else
  strExamId = "1"
End If
* check for errors
if Err.Number = 0 then
  PopulateUsageText = True
End If
End Function
Sub PopulateDrugGrid()
Dim rs As ADOCE.Recordset
Dim fid As ADOCE.Field
Dim strSQL As String
Dim irow, icol As Integer
Set rs = Nothing
On Error Resume Next
Set rs = cn.Execute("SELECT DRUG_ID AS ID, FAVORITE AS FAV, NAME, UNIT FROM DRUG")
* test for row returning commands, if no rowset was returned
* then we have nothing to display
If chk_rowreturning(rs) = False Then
  MsgBox "Please login first!", vbInformation, strTitleBar
Exit Sub
End If
On Error GoTo 0
For Each fid In rs.Fields
  grdDrug.Col = icol
  grdDrug.Text = rs(icol).Name
  icol = icol + 1
Next fid
* for all rows and poke values into the grdDrug control
irow = 1
Do While Not rs.EOF
  grdDrug.AddItem"
  grdDrug.Row = irow
  irow = irow + 1
  icol = 0
For Each fid In rs.Fields
  grdDrug.Col = icol
  icol = icol + 1
Next fid
End Sub
If Not IsNull(fld.Value) Then
grdDrug.Text = fld.Value
Else
    grdDrug.Text = "Null"
End If
icol = icol + 1
Next fid
grdDrug.RowData(grdDrug.Rows - 1) = rs.Fields("ID").Value
rs.MoveNext
Loop
End Sub

Private Sub grdDrug_Click()
    On Error Resume Next
    MsgBox "Rx is selected", vbInformation, strTitleBar
    glngDrugld = grdDrug.RowData(grdDrug.Row)
End Sub

Function AddDrug() As Boolean
    Dim rs As ADOCE.Recordset
    Dim i As Long
    Dim strSQL As String
    On Error Resume Next
    AddDrug = False
    Set rs = Nothing
    strSQL = "SELECT * FROM DRUG WHERE DRUG_ID = " & CStr(glngDrugld)
    Set rs = rs.Execute(strSQL)
    ' test for row returning commands, if no rowset was returned
    If chk_rowreturning(rs) = False Then
        MsgBox "Drug info can not be loaded", vbInformation, strTitleBar
        Exit Function
    End If
    On Error GoTo 0
    Me.TxtPrescriptionName.Text = rs.Fields("NAME").Value
    Me.TxtDose.Text = rs.Fields("UNIT").Value
    Me.TxtUsage.Text = rs.Fields("TREATMENT").Value
    ' check for errors, if none, return True
    If Err.Number = 0 Then
        AddDrug = True
    End If
End Function

Sub PopulateDrugFavorite()
    Dim rs As ADOCE.Recordset
    Dim fid As ADOCE.Field
    Dim strResult As String
    Dim irow, icol As Integer
    Dim strSQL
    Set rs = Nothing
    strSQL = "SELECT DRUG_ID AS ID, FAVORITE AS FAV, NAME, UNIT FROM DRUG WHERE FAVORITE = " & & CStr(glngDrugld)
    Set rs = rs.Execute(strSQL)
    ' test for row returning commands, if no rowset was returned
    If chk_rowreturning(rs) = False Then
        MsgBox "There is no bookmarked favorites!!", vbInformation, strTitleBar
        Exit Sub
    End If
    On Error GoTo 0
    For Each fid In rs.Fields
        grdDrug.Col = icol
        grdDrug.Text = rs.Fields(fid).Name
        icol = icol + 1
    Next fid
    ' for all rows and poke values into the grdDrug control
    irow = 1
    Do While Not rs.EOF
        grdDrug.AddItem"
        grdDrug.RowData(grdDrug.Rows - 1) = rs.Fields("ID").Value
        rs.MoveNext
        Loop
    End If
For Each fid In rs.Fields
    grdDrug.Col = icol
    If Not IsNull(fld.Value) Then
        grdDrug.Text = fld.Value
    Else
        grdDrug.Text = "Null"
    End If
    icol = icol + 1
Next fid
grdDrug.RowData(grdDrug.Rows - 1) = rs.Fields("ID").Value
rs.MoveNext
Loop
End Sub

Sub AddDrugFavorite()
    Dim rs As ADOCE.Recordset
    Dim fld As ADOCE.Field
    Dim strSQL
    Set rs = Nothing
    On Error Resume Next
    strSQL = "UPDATE DRUG SET FAVORITE = 'Y' WHERE DRUG_ID = " & CStr(glngDrugId)
    Set rs = rx.Execute(strSQL)
    If chk_rowreturning(rs) = False Then
        grdDrug.Rows = 1
        grdDrug.Clear
        PopulateDrugGrid
        MsgBox "Selected Favorite is Added", vbInformation, strTitleBar
        Exit Sub
    End If
    On Error GoTo 0
End Sub

Sub RemoveDrugFavorite()
    Dim rs As ADOCE.Recordset
    Dim fld As ADOCE.Field
    Dim strSQL
    Set rs = Nothing
    On Error Resume Next
    strSQL = "UPDATE DRUG SET FAVORITE = 'N' WHERE DRUG_ID = " & CStr(glngDrugId)
    Set rs = rx.Execute(strSQL)
    If chk_rowreturning(rs) = False Then
        grdDrug.Rows = 1
        grdDrug.Clear
        PopulateDrugGrid
        MsgBox "Selected Favorite is Removed", vbInformation, strTitleBar
        Exit Sub
    End If
    On Error GoTo 0
End Sub

Sub RemoveAllDrugFavorite()
    Dim rs As ADOCE.Recordset
    Dim fld As ADOCE.Field
    Dim strSQL
    Set rs = Nothing
    On Error Resume Next
    strSQL = "UPDATE DRUG SET FAVORITE = 'N'"
    Set rs = rx.Execute(strSQL)
    If chk_rowreturning(rs) = False Then
        grdDrug.Rows = 1
        grdDrug.Clear
        PopulateDrugGrid
        MsgBox "All Favorites are Removed", vbInformation, strTitleBar
        Exit Sub
    End If
    On Error GoTo 0
End Sub
AuthBean.java

* Description: The class AuthBean handles user authentication.
* Jarvis (Cha-Yu), Tsai
* CSCI CSUSB

package pe.auth;
import javax.servlet.http.*;
import java.math.BigInteger;
import pe.db.DBUser;

public class AuthBean {
    /*
     * particular user types existed for the system *
     *
     * public static final int DOCTOR = 6;
     * public static final int NURSE = 2;
     * public static final int ADMINISTRATOR = 4;
     * public static final int PHARMACIST = 8;
     * public static final int ALL = 31;
     */

    /*
     * email of user *
     */
    private String email;

    /*
     * password, it is encrypted *
     */
    private String password;

    /*
     * login page *
     */
    private static String loginPage = "login.jsp";

    /*
     * type of the user *
     */
    private int accountType;

    /*
     * menu page of the user *
     */
    private String menuPage;

    private String firstName;
    private String middleName;
    private String lastName;
    private String employeeld;

    /*
     * id of the user *
     */
    private String userid;

    public AuthBean() {
        
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String pswd) {
        this.password = pswd;
        (pswd == null) ? null : DigestMD5.digest(pswd);
    }

    public String getLoginPage() {
        return loginPage;
    }

    public void setLoginPage(String page) {
        loginPage = page;
    }

    public void setAccountType(int type) {
        accountType = type;
    }

    public void setMenuPage(String page) {
        menuPage = page;
    }

    public int getAccountType() {
        return accountType;
    }

    public String getMenuPage() {
        return menuPage;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getMiddleName() {
        return middleName;
    }
}
public void setMiddleName(String middleName) {
    this.middleName = middleName;
}

public String getMiddleName() {
    return middleName;
}

public void setUserId(String userId) {
    this.userid = userId;
}

public String getUserId() {
    return userId;
}

public void setEmployeeId(String employeeId) {
    this.employeeId = employeeId;
}

public String getEmployeeId() {
    return employeeId;
}

public String getEmployeeld() {
    return employeeld;
}

public void setEmployeeld(String employeeld) {
    this.employeeld = employeeld;

/**
 *  allows a logged-in user view all pages
 */
public void authenticate(HttpServletRequest req, HttpServletResponse res) throws Exception {
    authenticate(req, res, ALL);
}

/**
 *  determine if a user has right to view a page with specific privilege
 */
public void authenticate(HttpServletRequest req, HttpServletResponse res, int userType) throws Exception {
    if (accountType == 0) {
        throw new AuthFailedException("sorry, your session has expired, please <a href="
                + req.getContextPath() + loginPage + "">login again via</a> ");
    } else if (userType & accountType == 0) {
        System.out.println("usrType=" + userType);
        System.out.println("accountType = " + accountType);
        throw new AuthFailedException("sorry, you don't have right to access this page, click * +
                "<a href = "+ req.getContextPath() + menuPage + "/">here</a> to access your menu");
    }
}

/**
 *  login and initiate information of the user
 */
public void processLogin(HttpServletRequest req, HttpServletResponse res) throws Exception {
    if(req.getParameter("login") == null) {
        return;
    }
    if(userId == null || password == null) {
        throw new AuthFailedException("both user name and password need to be provided");
    }
    new DBUser().login(this);
    HttpSession session = req.getSession();
    session.setMaxInactiveInterval(1800);
    resp.sendRedirect(menuPage).forward(req, res);
}

/**
 *  logout and invalidate the session
 */
public void processLoginOut(HttpServletRequest req, HttpServletResponse res) throws Exception {
    if(req.getParameter("logout") == null) {
        HttpSession session = req.getSession();
        session.invalidate();
    }
}

/**
 *  allows a logged-in user change his/her password
 */
public void processChangePassword(HttpServletRequest req, HttpServletResponse res) throws Exception {
    authenticate(req, res);
    if(req.getParameter("changePassword") == null) {
        return;
    }
    String oldPassword = req.getParameter("oldPassword");
    String newPassword1 = req.getParameter("newPassword1");
    String newPassword2 = req.getParameter("newPassword2");
    if(oldPassword == null || newPassword1 == null || newPassword2 == null) {
        throw new AuthFailedException("all fields must be filled.");
    }
    if(oldPassword.equals("") ||
        !newPassword1.equals(newPassword2)) {
        throw new AuthFailedException("please provide your original password");
    }
    if((newPassword1.equals(newPassword2)) {
        throw new AuthFailedException("your input of new passwords did not match");
    }
    oldPassword = DigestMD5.digest(oldPassword);
    if(!oldPassword.equals(password)) {
        throw new AuthFailedException("the old password you provided is incorrect");
    }
    new DBUser().changePassword(userId, newPassword1);
    req.getSession().invalidate();
    req.getRequestDispatcher("login.jsp").forward(req, res);
package pe.auth;
import pe.util.UASException;

public class AuthFailedException extends UASException {
    public AuthFailedException()
    {
        title = "Authentication Failed";
    }

    public AuthFailedException(String errorMsg)
    {
        super(errorMsg);
        title = "Authentication Failed";
    }

    public AuthFailedException(String title, String errorMsg)
    {
        super(errorMsg, title);
    }
}
/**
 * Title: DigestMD5.java
 * Description: this class provides a function to digest a string with MD5.
 * Jarvis (Chia-Yu), Tsai
 * CSCI 3808F
 */

package pe.auth;
import java.security.*;
import java.math.*;

public class DigestMD5 {
    public static String digest(String input) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(input.getBytes());
            // convert digested byte array to a positive BigInteger
            BigInteger bint = new BigInteger(1, md.digest());
            // convert to 16-radix string
            return bint.toString(16);
        } catch (Exception e) {
            e.printStackTrace();
            return "";
        }
    }
}
package pe.account;

public abstract class Account {

    /** Holds value of property userid. */
    private String userid;

    /** Holds value of property userName. */
    private String email;

    /** Holds value of property firstName. */
    private String firstName;

    /** Holds value of property lastName. */
    private String lastName;

    /** Holds value of property middleName. */
    private String middleName;

    /** Creates new Account */
    public Account() {
    }

    /** Getter for property userid. */
    public String getUserld() {
        return userid;
    }

    /** Setter for property userid. */
    public void setUserld(String userid) {
        this.userid = userid;
    }

    /** Getter for property userName. */
    public String getEmail() {
        return email;
    }

    /** Setter for property userName. */
    public void setEmail(String email) {
        this.email = email;
    }

    /** Getter for property firstName. */
    public String getFirstName() {
        return firstName;
    }

    /** Setter for property firstName. */
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    /** Getter for property lastName. */
    public String getLastName() {
        return lastName;
    }

    /** Setter for property lastName. */
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    /** Getter for property middleName. */
    public String getMiddleName() {
        return middleName;
    }

    /** Setter for property middleName. */
    public void setMiddleName(String middleName) {
        this.middleName = middleName;
    }
}
package pe.db;
import pe.util.UASException;

public class AppointmentFailedException extends UASException {
    public AppointmentFailedException() {
        title = "Appointment Failed";
    }
    public AppointmentFailedException(String errorMsg) {
        super(errorMsg);
        title = "Appointment Failed";
    }
    public AppointmentFailedException(String title, String errorMsg) {
        super(errorMsg, title);
    }
}
This class is the main object that holds all personal information stored in the Database. It uses a static TreeMap for quick access.

```java
package pe.db;
import java.lang.*;
import java.sql.*;
import java.util.*;
import pe.form.AppointmentInput;
import pe.form.NoSuchUserException;
import pe.util.UASException;
import pe.db.AppointmentFailedException;

public class DBAppointments {
    /** Creates new DBAppointments */
    public DBAppointments() {
    }

    public void createAppointment(AppointmentInput person) throws UASException {
        Connection conn = null;
        Statement stmt = null;
        try {
            conn = DBUAS.getConnection();
            stmt = conn.createStatement();

            if (person.getPatientId().length() > 9) {
                throw new AppointmentFailedException("Sorry, the patient ID you provided is invalid");
            }

            if (person.getDt().equals("")) {
                throw new AppointmentFailedException("Sorry, the appointment day you provided is invalid");
            }

            if (person.getAppointmentTime().equals("")) {
                throw new AppointmentFailedException("Sorry, the appointment time you provided is invalid");
            }

            if (person.getDoctorId().equals("")) {
                throw new AppointmentFailedException("Sorry, the doctor name you provided is invalid");
            }

            ResultSet rs = stmt.executeQuery("Select * from Patient where patient_id = "+person.getPatientId());
            if (!rs.next() || person.getPatientId().equals("")) {
                throw new AppointmentFailedException("Sorry, the patient ID you provided is invalid");
            }
            rs.close();

            String sql = "insert into appointment (APPOINTMENT_TIME, APPOINTMENT_DAY,
                +"DOCTOR, PATIENT) values" + "+" + person.getDt() + "," + "+" + person.getAppointmentTime() + "," + "+" + person.getDoctorId() + "," + "+" + person.getPatientId() + ");";

            stmt.executeUpdate(sql);
        } catch (SQLException e) {
            throw new UASException(e.getMessage());
        } finally {
            try {
                DBUAS.freeConnection(conn);
                if (stmt != null)
                    stmt.close();
            } catch (SQLException ignore) {
            }
        }
    }

    public void updateAppointment(AppointmentInput person) throws UASException {
        Connection conn = null;
        Statement stmt = null;
        try {
            conn = DBUAS.getConnection();
            stmt = conn.createStatement();

            if (person.getPatientId().length() > 9) {
                throw new AppointmentFailedException("Sorry, the patient ID you provided is invalid");
            }

            if (person.getDt().equals("")) {
                throw new AppointmentFailedException("Sorry, the appointment day you provided is invalid");
            }

            if (person.getAppointmentTime().equals("")) {
                throw new AppointmentFailedException("Sorry, the appointment time you provided is invalid");
            }

            if (person.getDoctorId().equals("")) {
                throw new AppointmentFailedException("Sorry, the doctor name you provided is invalid");
            }

            String sql = "update appointment (APPOINTMENT_TIME, APPOINTMENT_DAY,
                +"DOCTOR, PATIENT) set" + "+" + person.getDt() + "," + "+" + person.getAppointmentTime() + "," + "+" + person.getDoctorId() + "," + "+" + person.getPatientId() + ");";

            stmt.executeUpdate(sql);
        } catch (SQLException e) {
            throw new UASException(e.getMessage());
        } finally {
            try {
                DBUAS.freeConnection(conn);
                if (stmt != null)
                    stmt.close();
            } catch (SQLException ignore) {
            }
        }
    }
}
```
throw new AppointmentFailedException("Sorry, the appointment time you provided is invalid");
}
if (person.getDoctorId().equals(""))
throw new AppointmentFailedException("Sorry, the doctor name you provided is invalid");
)

ResultSet rs = stmt.executeQuery("SELECT * FROM Patient WHERE patient_id = " + person.getPatientId() + ");
if (!rs.next())
person.setPatientId(rs.getString("PATIENT_ID"));

try{
  Statement stmt = conn.createStatement();
  stmt.executeUpdate(sql);
  String sql = "update appointment set " + "APPOINTMENT_TIME = '" + person.getAppointmentTime() + ";" + "APPOINTMENT_DAY = '" + person.getAppointmentDay() + ";" + "DOCTOR = '" + person.getDoctorId() + ";" + "PATIENT = '" + person.getPatientId() + ";" + "WHERE APPOINTMENT_ID = '" + person.getAppointmentId() + ";";
  stmt.executeUpdate(sql);
}

private void loadAppointment(AppointmentInfo input) throws UASException {
  Connection conn = null;
  Statement stmt = null;
  try{
    conn = DBUAS.getConnection();
    stmt = conn.createStatement();
    String sql = "SELECT APPOINTMENT_ID, CAST(DATEPART(hh, APPOINTMENT_TIME) AS varchar(5)) + CAST(DATEPART(mi, APPOINTMENT_TIME) AS varchar(5)) AS DOCTOR, APPOINTMENT_DAY AS APPOINTMENT_TIME, CAST(MONTH(APPOINTMENT_DAY) AS varchar(5)) + CAST(YEAR(APPOINTMENT_DAY) AS varchar(5)) AS APPOINTMENT_DAY, DOCTOR, PATIENT FROM APPOINTMENT WHERE APPOINTMENT_ID = '" + person.getAppointmentId() + ";";
    ResultSet rs = stmt.executeQuery(sql);
    if (!rs.next())
      throw new NoSuchUserException(person.getAppointmentId());
    String time = rs.getString("APPOINTMENT_TIME");
    int hours = time.substring(0, 1);
    String ampm = time.substring(1);
    if (hours >= 12)
      if (ampm.equals("PM"))
        hours -= 12;
    if (hours < 12)
      if (ampm.equals("AM"))
        hours += 12;
    String minutes = time.substring(2, 8);
    String temp = hours + minutes + "" + ":" + ampm;
    person.setAppointmentTime(temp);
    person.setAppointmentDay(rs.getString("DOCTOR"));
    person.setPatientId(rs.getString("PATIENT"));
  }
  catch (SQLException e)
    throw new UASException(e.getMessage());
  finally{
    try{
      DBUAS.freeConnection(conn);
      if (stmt != null)
        stmt.close();
    }
    catch (SQLException ignore) {
    }
  }
}

for doctor to get his appointment list of the selected day
public ResultSet getAppointmentList(String constraints) throws UASException {
  Connection conn = null;
  Statement stmt = null;
  try{
    conn = DBUAS.getConnection();
    stmt = conn.createStatement();
    StringBuffer sql = new StringBuffer("select a.APPOINTMENT_ID, CAST(DATEPART(hh, a.APPOINTMENT_TIME) AS varchar(5)) + CAST(DATEPART(mi, a.APPOINTMENT_TIME) AS varchar(5)) AS APPOINTMENT_TIME, CAST(MONTH(a.APPOINTMENT_DAY) AS varchar(5)) + CAST(YEAR(a.APPOINTMENT_DAY) AS varchar(5)) AS APPOINTMENT_DAY, Dr.*, c.C_NAME AS DOCTOR, b.L_NAME AS PATIENT FROM APPOINTMENT a, PATIENT b, EMPLOYEE c WHERE a.PATIENT = b.PATIENT_ID AND a.DOCTOR = c.EMPLOYEE_ID;"");
    if (constraints.length() != 0)
      sql.append(" AND ")
    .append(constraints);
    ResultSet rs = stmt.executeQuery(sql.toString());
    return rs;
  }
  finally{
    try{
      DBUAS.freeConnection(conn);
      if (stmt != null)
        stmt.close();
    }
    catch (SQLException ignore) {
    }
  }
}
catch (SQLException e) {
    throw new UASException(e.getMessage());
}

public void deleteAppointment(String id) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        stmt.executeUpdate("delete from appointment where APPOINTMENT_ID=\"" + id + \"\";
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    } finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        } catch (SQLException ignore) {
        }
    }
}
/**
 * DBDrugs.java
 * This class is the main object that holds all personal information stored in the
 * Database. It uses a static TreeMap for quick access.
 * Jarvis (Chia-Yu), Tsai
 * CSCI CSUSB
 */

package pe.db;
import java.lang.*;
import java.sql.*;
import java.util.*;
import pe.form.DruglnfoInput;
import pe.form.NoSuchUserException;
import pe.util.UASException;

public class DBDrugs {
    public DBDrugs() {}

    public void loadDrug(DruglnfoInput person) throws UASException {
        Connection conn = null;
        Statement stmt = null;
        try {
            conn = DBUAS.getConnection();
            stmt = conn.createStatement();
            String sql = "select NAME, UNIT, TREATMENT from DRUG where DRUG_ID = \"" + person.getDrugld() + \"\";
            ResultSet rs = stmt.executeQuery(sql);
            if (!rs.next())
                throw new NoSuchUserException(person.getDrugld());
            person.setRxName(rs.getString("NAME"));
            person.setDose(rs.getString("UNIT"));
            person.setUsage(rs.getString("TREATMENT"));
        } catch (SQLException e) {
            throw new UASException(e.getMessage());
        }
        finally {
            try {
                DBUAS.freeConnection(conn);
                if (stmt != null)
                    stmt.close();
            } catch (SQLException ignore) {
            }
        }
    }

    public ResultSet getDrugList(String constraints) throws UASException {
        Connection conn = null;
        Statement stmt = null;
        try {
            conn = DBUAS.getConnection();
            stmt = conn.createStatement();
            StringBuffer sql = new StringBuffer("select DRUG_ID, NAME, UNIT, TREATMENT, INDICATION, AVAILABILITY from DRUG");
            if (constraints.length() != 0) {
                sql.append(" where ").append(constraints);
            }
            ResultSet rs = stmt.executeQuery(sql.toString());
            return rs;
        } catch (SQLException e) {
            throw new UASException(e.getMessage());
        }
    }
}
This class is the main object that holds all personal information stored in the database. It uses a static TreeMap for quick access.

public class DBPatients {
    /** Creates new DBPatients */
    public DBPatients() {

        public void createPatient(PatientInfoInput person) throws UASException {
            Connection conn = null;
            Statement stmt = null;
            try {
                conn = DBUAS.getConnection();
                stmt = conn.createStatement();
                String sql = "insert into patient (EMAIL, F_NAME, M_NAME, L_NAME, STREET, CITY, STATE, ZIP, PHONE, SSN, GENDER, HEIGHT, WEIGHT, DOB, HISTORY) values("
                        + "'" + person.getEmail() + ","
                        + "'" + person.getFirstName() + ","
                        + "'" + person.getLastName() + ","
                        + "'" + person.getStreet() + ","
                        + "'" + person.getCity() + ","
                        + "'" + person.getState() + ","
                        + "'" + person.getZip() + ","
                        + "'" + person.getPhone() + ","
                        + "'" + person.getSsn() + ","
                        + "'" + person.getGender() + ","
                        + "'" + person.getHeight() + ","
                        + "'" + person.getWeight() + ","
                        + "'" + person.getDob() + ","
                        + ");";
                stmt.executeUpdate(sql);
                throw new UASException(e.getMessage());
            } finally {
                try {
                    DBUAS.freeConnection(conn);
                    if (stmt != null)
                        stmt.close();
                } catch (SQLException ignore) {
                }
            }

            public void updatePatient(PatientInfoInput person) throws UASException {
                Connection conn = null;
                Statement stmt = null;
                try {
                    conn = DBUAS.getConnection();
                    stmt = conn.createStatement();
                    String sql = "update patient set 
                        + "EMAIL=阳区 + person.getEmail() + ","
                        + "F_NAME=阳区 + person.getFirstName() + ","
                        + "M_NAME=阳区 + person.getMidName() + ","
                        + "L_NAME=阳区 + person.getLastName() + ","
                        + "STREET=阳区 + person.getStreet() + ","
                        + "CITY=阳区 + person.getCity() + ","
                        + "STATE=阳区 + person.getState() + ","
                        + "ZIP=阳区 + person.getZip() + ","
                        + "PHONE=阳区 + person.getPhone() + ","
                        + "SSN=阳区 + person.getSsn() + ","
                        + "DOB=阳区 + person.getDob() + ","
                        + "HEIGHT=阳区 + person.getHeight() + ","
                        + "WEIGHT=阳区 + person.getWeight() + ","
                        + "HISTORY=阳区 + person.getHistory() + ","
                        + "GENDER=阳区 + person.getGender() + ","
                        + "where PATIENT_ID=阳区 + person.getUserld() + ");"
                } catch (SQLException e) {
                }
                try {
                    stmt.executeUpdate(sql);
                    throw new UASException(e.getMessage());
                } finally {
                    try {
                        DBUAS.freeConnection(conn);
                        if (stmt != null)
                            stmt.close();
                } catch (SQLException ignore) {
                }
            }

            /*
             * DBStudents.java
             */
             package pe.db;
             import java.lang.*;
             import java.sql.*;
             import java.util.*;
             import java.util.FormaternoInput;
             import pe.form.NoSuchUserException;
             import pe.util.UASException;
             
             public class DBStudents {
                /** Creates new DBStudents */
                public DBStudents() {

                
                public void createStudent(StudentInfoInput person) throws UASException {
                    Connection conn = null;
                    Statement stmt = null;
                    try {
                        conn = DBUAS.getConnection();
                        stmt = conn.createStatement();
                        String sql = "insert into students (STUDENT_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE, SSN, CITY, STATE, ZIP, HISTORY) values("
                                + ");";
                        stmt.executeUpdate(sql);
                        throw new UASException(e.getMessage());
                    } finally {
                        try {
                            DBUAS.freeConnection(conn);
                            if (stmt != null)
                                stmt.close();
                        } catch (SQLException ignore) {
                        }
                    }

                    public void updateStudent(StudentInfoInput person) throws UASException {
                        Connection conn = null;
                        Statement stmt = null;
                        try {
                            conn = DBUAS.getConnection();
                            stmt = conn.createStatement();
                            String sql = "update students set 
                                + "FIRST_NAME=阳区 + person.getFirstName() + ","
                                + "LAST_NAME=阳区 + person.getLastName() + ","
                                + "EMAIL=阳区 + person.getEmail() + ","
                                + "PHONE=阳区 + person.getPhone() + ","
                                + "SSN=阳区 + person.getSsn() + ","
                                + "CITY=阳区 + person.getCity() + ","
                                + "STATE=阳区 + person.getState() + ","
                                + "ZIP=阳区 + person.getZip() + ","
                                + "HISTORY=阳区 + person.getHistory() + ","
                                + "where STUDENT_ID=阳区 + person.getUserld() + ");"
                        } catch (SQLException e) {
                        }
                        try {
                            stmt.executeUpdate(sql);
                            throw new UASException(e.getMessage());
                        } finally {
                            try {
                                DBUAS.freeConnection(conn);
                                if (stmt != null)
                                    stmt.close();
                            } catch (SQLException ignore) {
                            }
                        }
                    }
                }
            */
DBUAS.freeConnection(conn);
if (stmt != null)
    stmt.close();
}
catch (SQLException ignore) {
}
}

public void loadPatient(PatientInfoInput person) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        String sql = "select * from patient where PATIENT_ID = \"" + person.getUserID() + \"\";
        ResultSet rs = stmt.executeQuery(sql);
        if (!rs.next())
            throw new NoSuchUserException(person.getUserID());
        person.setEmail(rs.getString("EMAIL"));
        person.setFirstName(rs.getString("F_NAME"));
        person.setMiddleName(rs.getString("M_NAME"));
        person.setLastName(rs.getString("L_NAME"));
        person.setStreet(rs.getString("STREET"));
        person.setCity(rs.getString("CITY"));
        person.setState(rs.getString("STATE"));
        person.setZip(rs.getString("ZIP"));
        person.setPhone(rs.getString("PHONE"));
        person.setDob(rs.getString("DOB"));
        person.setGender(rs.getString("GENDER"));
        person.setSsn(rs.getString("SSN"));
        person.setHeight(rs.getString("HEIGHT"));
        person.setWeight(rs.getString("WEIGHT"));
        person.setHistory(rs.getString("HISTORY"));
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    }
    finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        } catch (SQLException ignore) {
        }
    }
}

public ResultSet getPatientList(String constraints) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        StringBuffer sql = new StringBuffer("select PATIENT_ID, L_NAME, M_NAME, F_NAME from PATIENT");
        if (constraints.length() != 0) {
            sql.append(" where ");
            String append = constraints;
            ResultSet rs = stmt.executeQuery(sql.toString());
            return rs;
        } catch (SQLException e) {
            throw new UASException(e.getMessage());
        }
    }
}

public void deletePatient(String id) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        stmt.executeUpdate("delete from patient where PATIENT_ID = \""+ id + \"\";
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    }
    finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        } catch (SQLException ignore) {
        }
    }
}
/**
 * DBPE.java
 * Description: this class constructs a connection pool for the system. all
 * other classes in the edu.csusb.csci.uas.db package get database connection
 * Jarvis (Chia-Yu), Tsai
 * CSCI CSUSB
 */

package pe.db;
import java.sql.*;
import java.io.IOException;
import com.javaexchange.dbConnectionBroker.DbConnectionBroker;

public class DBPE {
   /**
    * the broker object that provides the connection pool
    */
   private DbConnectionBroker connectionBroker;

   /**
    * the singleton object of this class
    */
   private static DBPE dbpe;

   /**
    * the private constructor of this singleton object
    */
   private DBPE() throws java.sql.SQLException {
      String url = "jdbc:mysql://localhost:6080/pe";
      String userName = "jarviscyt";
      String password = "ujs383";
      String jdbcDriver = "org.gjt.mm.mysql.Driver";
      try {
         connectionBroker = new DbConnectionBroker(jdbcDriver, url, userName, password, 2, 30, "dbConnection.log", 1.0);
      } catch (IOException e) { throw new java.sql.SQLException(e.toString()); }
   }

   /**
    * get a connection from the connection pool
    */
   public static synchronized Connection getConnection() throws java.sql.SQLException {
      if (dbpe == null) dbpe = new DBPE();
      return dbpe.connectionBroker.getConnection();
   }

   /**
    * return a connection to the pool
    */
   public static synchronized void freeConnection(Connection conn) {
      if (dbpe != null) dbpe.connectionBroker.freeConnection(conn);
      else try {
         conn.close();
      } catch (Exception ignore) {} 
   }
}
```java
/*
 * DBPrescription.java
 * This class is the main object that holds all personal information stored in the
 * Database. It uses a static TreeMap for quick access.
 * Java (Chia-Yu), Tsai
 * CSC 6568
 */
package pe.db;
import java.sql.*;
import java.util.*;
import pe.form.PrescriptionlnfoInput;
import pe.util.UASException;
import java.util.*;
import java.sql.*;

public class DBPrescription {
    //** Creates new DBStudents */
    public DBPrescription() {
    }

    public void makePrescription(PrescriptionlnfoInput person) throws UASException {
        Connection conn = null;
        Statement stmt = null;
        try {
            Calendar cal = new GregorianCalendar();
            int year = cal.get(Calendar.YEAR); // 2002
            int month = cal.get(Calendar.MONTH) + 1; // 0=Jan, 1=Feb, ...
            int day = cal.get(Calendar.DAY_OF_MONTH); // 1...
            String temp = month + "*" + day + "*" + String.valueOf(year).substring(2, 4);
            conn = DBUAS.getConnection();
            stmt = conn.createStatement();
            String sql2 = "INSERT INTO EXAM*
              + "(DOCTOR, PATIENT, NOTES, DATE) *
              + "VALUES (*
              + "+" + person.getDoctorId() + "*
              + "+" + person.getPatientId() + "*
              + "+" + person.getNotes() + "*
              + "+" + temp + "*");
            int s = 0;
            String sql3 = "SELECT MAX(EXAM_ID) AS EXAM_ID FROM EXAM WHERE PATIENT = " + person.getPatientId() + " AND DATE = " + temp + "*";
            String sql4 = "SELECT EXAM_ID FROM EXAM WHERE PATIENT = " + person.getPatientId() + " AND DATE = " + temp + "*";
            ResultSet rs = stmt.executeQuery(sql3);
            while (rs.next()) {
                s = rs.getInt("EXAM_ID");
            }
            if (s == 0) {
                stmt.executeUpdate(sql2);
                ResultSet rs2 = stmt.executeQuery(sql4);
                while (rs2.next()) {
                    s = rs2.getInt("EXAM_ID");
                }
            }
            String sql1 = "INSERT INTO PRESCRIPTION *
              + "(RX_NAME, DOSE, DISPENSE, TREATMENT, SUBSTITUTE, STATUS, REFILL, EXAM, DOCTOR, PATIENT, TIME) *
              + "VALUES (*
              + "+" + person.getRxName() + "*
              + "+" + person.getDose() + "*
              + "+" + person.getDispense() + "*
              + "+" + person.getTreatments() + "*
              + "+" + person.getSubstitutes() + "*
              + "+" + person.getRefills() + "*
              + "+" + person.getDoctorId() + "*
              + "+" + person.getPatientId() + "*
              + "+" + getExami()");
            stmt.executeUpdate(sql1);
        } catch (SQLException e) {
            throw new UASException(e.getMessage());
        } finally {
            try {
                DBUAS.freeConnection(conn);
                if (stmt != null)
                    stmt.close();
            } catch (SQLException Ignore) {}
        }
    }

    //This one is for pharmacist to update the prescription status upon received order, processing, completed, picked-up
    public void updatePrescriptionStatus(PrescriptionlnfoInput person) throws UASException {
        Connection conn = null;
    }
```

Statement stmt = null;
try {
    conn = DBUAS.getConnection();
    stmt = conn.createStatement();
    String sql = "update prescription set "
        + "STATUS='V' + person.getStatus() + '+'",
        + "PHARMACIST='V' + person.getPharmacistId() + '+'",
        + "REFILL='V' + person.getRefill() + '+'",
        + "tme=time() + '+'",
        + "where PRESCRIPTION_ID='V' + person.getPid() + '+'";
    stmt.executeUpdate(sql);
} catch (SQLException e) {
    throw new UASException(e.getMessage());
} finally {
    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException ignore) {
    }
}

public void loadPrescription(PrescriptionInp person) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        String sql = "select * from prescription where PRESCRIPTION_ID='V' + person.getPid() + '+'";
        ResultSet rs = stmt.executeQuery(sql);
        if (!rs.next())
            throw new NoSuchUserException(person.getPid());
        person.setPrescriptionId(rs.getString("PRESCRIPTION_ID"));
        person.setRxName(rs.getString("RX_NAME"));
        person.setDose(rs.getString("DOSE"));
        person.setDispense(rs.getString("DISPENSE"));
        person.setUsage(rs.getString("TREATMENT"));
        person.setSubstitute(rs.getString("SUBSTITUTE"));
        person.setRefill(rs.getString("REFILL"));
        person.setDoctorId(rs.getString("DOCTOR"));
        person.setPatientId(rs.getString("PATIENT"));
        person.setPharmacistId(rs.getString("PHARMACIST"));
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    } finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        } catch (SQLException ignore) {
        }
    }
}

//this one is for nurse to check prescription status for the patients
public ResultSet loadPrescriptionStatus(String constraints) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        StringBuffer sql = new StringBuffer("select PRESCRIPTION_ID, STATUS from PRESCRIPTION");
        if (constraints.length() != 0) {
            sql.append(" where ").append(constraints);
        }
        ResultSet rs = stmt.executeQuery(sql.toString());
        return rs;
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    }
}

//this one is for pharmacist to check refill prescritions for individual patient
public ResultSet loadRefillPrescription(String constraints) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        StringBuffer sql = new StringBuffer("select * from PRESCRIPTION");
        if (constraints.length() != 0) {
            sql.append(" where ").append(constraints);
        }
        ResultSet rs = stmt.executeQuery(sql.toString());
        return rs;
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    }
}
throw new UASEException(e.getMessage());
        }
    }
}

//This one is for pharmacist to view all prescription order which are not picked up by the patients
public ResultSet loadPrescriptionOrders() throws UASEException {
    Connection conn = null;
    Statement stmt = null;
    try {
        Calendar cal = new GregorianCalendar();
        int year = cal.get(Calendar.YEAR); // 2002
        int month = cal.get(Calendar.MONTH) + 1; // 1=Jan, 2=Feb, ...
        int day = cal.get(Calendar.DAY_OF_MONTH); // 1...
        String temp = month + "" + day + "" + String.valueOf(year).substring(2, 4);
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        String sql = "SELECT a.PRESCRIPTION_ID, a.DISPENSE, a.TREATMENT, a.SUBSTITUTE, a.STATUS, a.REFILL, a.EXAM, b.F_NAME+""+b.L_NAME AS PATIENT, a.PHARMACIST, a.RX_NAME, a.DOSE, a.TIME FROM PRESCRIPTION a, PATIENT b, PRESCRIPTION_ORDER c WHERE a.PATIENT=b.PATIENT_ID and a.ORDER=DATEPART(year, GETDATE()) ORDER by a.TIME;";
        ResultSet rs = stmt.executeQuery(sql);
        return rs;
    } catch (SQLException e) {
        throw new UASEException(e.getMessage());
    }
}

//This one is for doctor to view all prescription order that he/she ordered
public ResultSet loadPrescriptionOrders(String userid) throws UASEException {
    Connection conn = null;
    Statement stmt = null;
    try {
        Calendar cal = new GregorianCalendar();
        int year = cal.get(Calendar.YEAR); // 2002
        int month = cal.get(Calendar.MONTH) + 1; // 1=Jan, 2=Feb, ...
        int day = cal.get(Calendar.DAY_OF_MONTH); // 1...
        String temp = month + "" + day + "" + String.valueOf(year).substring(2, 4);
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        String sql = "SELECT a.PRESCRIPTION_ID, a.DISPENSE, a.TREATMENT, a.SUBSTITUTE, a.STATUS, a.REFILL, a.EXAM, Dr."+c.F_NAME+""+c.L_NAME AS PATIENT, a.PHARMACIST, a.RX_NAME, a.DOSE, a.TIME FROM PRESCRIPTION a, PATIENT b, PRESCRIPTION_ORDER c WHERE a.DOCTOR=""+userid+"" and a.EXAM Ngày = 1 and a.ORDER=DATEPART(year, GETDATE()) ORDER by a.time;";
        ResultSet rs = stmt.executeQuery(sql);
        return rs;
    } catch (SQLException e) {
        throw new UASEException(e.getMessage());
    }
}

//for doctor to delete prescription
public void deletePrescription(String id) throws UASEException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        stmt.execute("DELETE FROM PRESCRIPTION WHERE PRESCRIPTION_ID="+id+""");
    } catch (SQLException e) {
        throw new UASEException(e.getMessage());
    } finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        } catch (SQLException ignore) {} 
    }
}

//This one is for doctor to modify the prescription order
public void modifyPrescription(PrescriptionInfo input person) throws UASEException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        String sql = "UPDATE PRESCRIPTION SET "+"RX_NAME= "" + input.person.getRxName() + ","
               + "DOSE= "" + input.person.getDose() + ",";
        stmt.executeUpdate(sql);
        return;
    }
}

96
```java
$stmt.executeUpdate(sql);
} catch (SQLException e) {
    throw new UASException(e.getMessage());
}
finally {
    try {
        DDUBAS.freeConnection(conn);
        if (stmt != null)
            stmt.close();
    } catch (SQLException ignore) {
    }
    catch (SQLException ignore) {
    }
}
```
package pe.db;

import java.sql.*;
import java.io.IOException;
import com.microsoft.jdbc.sqlserver.SQLServerDriver;
import com.javaexchange.dbConnectionBroker.DbConnectionBroker;

public class DBUAS {

    /** the broker object that provides the connection pool */
    private DbConnectionBroker connectionBroker;
    /** the singleton object of this class */
    private static DBUAS dbuas;

    /** the private constructor of this singleton object */
    private DBUAS() throws java.sql.SQLException {
        String databaseName = "pe";
        String userName = "sa";
        String password = "33227277";
        // Informs the driver to use server-side cursors,
        // which permits more than one active statement
        // on a connection.
        String selectMethod = "cursor";

        String url = "jdbc:microsoft:sqlserver://localhost:1433" + databaseName;
        String jdbcDriver = "com.microsoft.jdbc.sqlserver.SQLServerDriver";
        try {
            connectionBroker = new DbConnectionBroker(jdbcDriver, url, userName, password, 2, 30, "DBConnection.log", 1.0);
        } catch (IOException e) { throw new java.sql.SQLException("The driver is not good") + e.toString(); }
    }

    /** get a connection from the connection pool */
    public static synchronized Connection getConnection() throws java.sql.SQLException {
        if (dbuas == null) dbuas = new DBUAS();
        return dbuas.connectionBroker.getConnection();
    }

    /** return a connection to the pool */
    public static synchronized void freeConnection(Connection conn) {
        if (dbuas != null) dbuas.connectionBroker.freeConnection(conn);
        else try {
            conn.close();
        } catch (Exception ignore) {} 
    }

}
/**
 * Title: DBUser.java
 * Description: this class handles account related functions such as login,
 * changed password, create account and delete account
 * Jarvis (Chia-Yu), Tsai
 * CSCI CSUSB
 */

package pe.db;
import java.sql.*;
import java.math.*;
import pe.auth.*;
import pe.form.*;
import pe.util.UASException;

public class DBUser {
    public DBUser() {
    }

    /**
     * allows a user to login
     * @param authBean which will be populated with account information after
     * the call
     */
    public void login(AuthBean auth) throws UASException {
        Connection conn = null;
        Statement stmt = null;
        try {
            conn = DBUAS.getConnection();
            stmt = conn.createStatement();
            String password = auth.getPassword();
            String userid = auth.getUserid();
            String sql = "select * from EMPLOYEE where USER_ID = '" + userid + "';";
            ResultSet rs = stmt.executeQuery(sql);
            if (!rs.next()) {
                throw new AuthFailedException("Sorry, your data is not recorded in our database.");
            }
            String stored = new BigInteger(rs.getString("PASSWORD"), 16).toString(16);
            if (!password.equals(stored)) {
                throw new AuthFailedException("Sorry, the password you provided is invalid");
            }
            auth.setUserid(rs.getString("USER_ID"));
            auth.setFirstName(rs.getString("F_NAME"));
            auth.setMiddleName(rs.getString("M_NAME"));
            auth.setLastName(rs.getString("L_NAME"));
            auth.setEmployeeld(rs.getString("EMPLOYEE_ID"));
            String type = rs.getString("TYPE");
            if (type.equals("doctor")) {
                auth.setAccountType(pe.auth.AuthBean.DOCTOR);
                auth.setMenuPage("/doctor/doctor.jsp");
            } else if (type.equals("administrator")) {
                auth.setAccountType(pe.auth.AuthBean.ADMINISTRATOR);
                auth.setMenuPage("/administrator/administrator.jsp");
            } else if (type.equals("nurse")) {
                auth.setAccountType(pe.auth.AuthBean.NURSE);
                auth.setMenuPage("/nurse/nurse.jsp");
            } else if (type.equals("pharmacist")) {
                auth.setAccountType(pe.auth.AuthBean.PHARMACIST);
                auth.setMenuPage("/pharmacist/pharmacist.jsp");
            }
        } catch (SQLException e) {
            throw new UASException("The driver cannot be loaded" + e.toString());
        } finally {
            try {
                DBUAS.freeConnection(conn);
                if (stmt != null)
                    stmt.close();
            } catch (SQLException ignore) {
            } catch (SQLException e) {
                throw new UASException("The driver cannot be loaded" + e.toString());
            } finally {
                try {
                    DBUAS.freeConnection(conn);
                    if (stmt != null)
                        stmt.close();
                } catch (SQLException ignore) {
                } catch (SQLException e) {
                    throw new UASException("The driver cannot be loaded" + e.toString());
                }
        }
    }

    /**
     * change password of a user
     * @param email email is used as the key for a user
     * @param newPassword new password
     */
    public void changePassword(String userid, String newPassword) throws UASException {
        Connection conn = null;
        Statement stmt = null;
        try {
            conn = DBUAS.getConnection();
            stmt = conn.createStatement();
            stmt.executeUpdate("update employee set PASSWORD = '" + newPassword + "' where USER_ID = '" + userid + "';");
            if (stmt.getUpdateCount() != -1) {
                throw new NoSuchUserException(userid);
            }
        } finally {
            try {
                DBUAS.freeConnection(conn);
                if (stmt != null)
                    stmt.close();
            } catch (SQLException ignore) {
            } catch (SQLException e) {
                throw new UASException("The driver cannot be loaded" + e.toString());
            }
        }
    }
}
catch (SQLException e) {
    throw new UASException("database connection error");
}

/**
 * create a use account
 * @param userid use id
 * @param email
 * @param password password
 * @param type user type
 */
public void createEmployee(EmployeeInput person) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        String sql = "insert into employee (USER_ID, PASSWORD, EMAIL, F_NAME, M_NAME, L_NAME, DEPT, TYPE, ESSN, PHONE, STREET, CITY, STATE, ZIP) values(";
        sql += person.getUserid() + ",";
        sql += person.getPassword() + ",";
        sql += person.getEmail() + ",";
        sql += person.getFirstName() + ",";
        sql += person.getMidName() + ",";
        sql += person.getLastName() + ",";
        sql += person.getDept() + ",";
        sql += person.getType() + ",";
        sql += person.getSsn() + ",";
        sql += person.getPhone() + ",";
        sql += person.getStreet() + ",";
        sql += person.getCity() + ",";
        sql += person.getState() + ",";
        sql += person.getZip() + ");";
        stmt.executeUpdate(sql);
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    } finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        } catch (SQLException ignore) {} 
    }
}

/UPDATE EMPLOYEE
public void updateEmployee(EmployeeInput person) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        String sql = "update employee set ";
        sql += "USER_ID = " + person.getUserid() + ",";
        sql += "EMAIL = " + person.getEmail() + ",";
        sql += "F_NAME = " + person.getFirstName() + ",";
        sql += "M_NAME = " + person.getMidName() + ",";
        sql += "L_NAME = " + person.getLastName() + ",";
        sql += "STREET = " + person.getStreet() + ",";
        sql += "CITY = " + person.getCity() + ",";
        sql += "STATE = " + person.getState() + ",";
        sql += "ZIP = " + person.getZip() + " where USER_ID = " + person.getUserid() + ");";
        stmt.executeUpdate(sql);
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    } finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        } catch (SQLException ignore) {} 
    }
}

public void loadEmployee(EmployeeInput person) throws UASException {
    Connection conn = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        String sql = "select ";
        sql += "USER_ID, PASSWORD, EMAIL, F_NAME, M_NAME, L_NAME, DEPT, TYPE, ESSN, PHONE, STREET, CITY, STATE, ZIP ";
        sql += "from employee where USER_ID = " + person.getUserid() + ");";
        stmt.executeUpdate(sql);
    } catch (SQLException e) {
        throw new UASException(e.getMessage());
    } finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        } catch (SQLException ignore) {} 
    }
}
try {
    conn = DBUAS.getConnection();
    stmt = conn.createStatement();
    String sql = "select * from employee where USER_ID = " + person.getUserld() + "";
    ResultSet rs = stmt.executeQuery(sql);
    if (!rs.next())
        throw new NoSuchUserException(person.getUserld());
    person.setEmail(rs.getString("EMAIL"));
    person.setUserId(rs.getString("USERJD"));
    person.setFirstName(rs.getString("F_NAME"));
    person.setMiddleName(rs.getString("M_NAME"));
    person.setLastName(rs.getString("L_NAME"));
    person.setCity(rs.getString("CITY"));
    person.setStreet(rs.getString("STREET"));
    person.setZipCode(rs.getString("ZIP"));
    person.setPhone(rs.getString("PHONE"));
    person.setType(rs.getString("TYPE"));
    person.setDepartment(rs.getString("DEPT"));
    person.setSSN(rs.getString("SSN"));
    person.setPassword(rs.getString("PASSWORD"));
}
}

public ResultSet getEmployeeList(String constraints) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        StringBuffer sql = new StringBuffer("select USERJD, L_NAME, M_NAME, F_NAME from EMPLOYEE");
        if (constraints.length() != 0) {
            sql.append(" where ");
            if (constraints.startsWith("or", 0))
                sql.append(constraints);
            else
                sql.append(" and ");
        }
        ResultSet rs = stmt.executeQuery(sql.toString());
        return rs;
    }
        catch (SQLException e) { 
            throw new UASException(e.getMessage());
    }
}

/**
 * load doctor name in the combo box for nurse to make new appointment
 */
public ResultSet loadDoctors() throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        StringBuffer sql = new StringBuffer("select employeejd as ID, 'Dr. ' + F_NAME + ' ' + L_NAME AS NAME from EMPLOYEE where type = 'doctor' ");
        ResultSet rs = stmt.executeQuery(sql.toString());
        return rs;
    }
        catch (SQLException e) { 
            throw new UASException(e.getMessage());
    }
}

/**
 * delete an account
 */
public void deleteEmployee(String id) throws UASException {
    Connection conn = null;
    Statement stmt = null;
    try {
        conn = DBUAS.getConnection();
        stmt = conn.createStatement();
        stmt.executeUpdate("delete from employee where USER_ID = " + id + ");
    }
        catch (SQLException e) { 
            throw new UASException(e.getMessage());
    }
    finally {
        try {
            DBUAS.freeConnection(conn);
            if (stmt != null)
                stmt.close();
        }
        catch (SQLException ignore) {}
public class AppointmentFilter {
    private String appointmentTime;
    private String dt;
    private String doctorId;
    private String patientId;

    public ResultSet getAppointmentList(HttpServletRequest req, HttpServletResponse res) throws UASException {
        if (req.getParameter("find") == null) {
            return null;
        }
        verifyO();
        if (patientId != null && !patientId.equals("")) {
            verifyPatient();
            String constraints = getConstraints();
            return new DBAppointments().getAppointmentList(constraints);
        }
    }

    public void verifyO() throws UASException {
        try {
            if (dt.equals("")) {
                throw new AppointmentFailedException("sorry, the search criteria you provided is invalid");
            }
        } catch (Exception e) {
            throw new AppointmentFailedException("sorry, the search criteria you provided is invalid");
        }
    }

    public void verifyPatient() throws UASException {
        ResultSet rst;
        Connection conn = null;
        Statement stmt = null;
        try {
            conn = DBUAS.getConnection();
            stmt = conn.createStatement();
            String query = "Select * from Patient where patient_id = " + patientId + ";
            if (stmt.next()) {
                throw new AppointmentFailedException("sorry, the patient ID you provided is invalid");
            }
            rst.close();
        } catch (SQLException e) {
            throw new AppointmentFailedException("sorry, the patient ID you provided is invalid");
        } finally {
            DBUAS.freeConnection(conn);
            if (stmt != null) stmt.close();
        }
    }

    private String getConstraints() {
        StringBuffer constraints = new StringBuffer();
        String str = "";
        if (appointmentTime != null && appointmentTime.equals("")) {
            str = "a.APPOINTMENT_TIME = " + appointmentTime + "\n";
            if (constraints.length() == 0) constraints.append(str);
        }
        if (dt != null && dt.equals("")) {
            str = "a.APPOINTMENT_DAY = " + dt + "\n";
            if (constraints.length() == 0) constraints.append(str);
        } else constraints.append(" and " + str);
    }
}
if (doctorld == null && !doctorld.equals("")) {
    str = "a.DOCTOR\n" + doctorld + ";
    if (constraints.length() == 0)
        constraints.append(str);
    else
        constraints.append(" and " + str);
}
if (patientld != null && !patientld.equals("")) {
    str = "a.PATIENT\n" + patientld + ";
    if (constraints.length() == 0)
        constraints.append(str);
    else
        constraints.append(" and " + str);
}
return constraints.toString();
}
public void setAppointmentTime(String appointmentTime) {
    this.appointmentTime = appointmentTime;
}
public String getAppointmentTime() {
    return appointmentTime;
}
public void setDt(String dt) {
    this.dt = dt;
}
public String getDt() {
    return dt;
}
public void setDoctorId(String doctorId) {
    this.doctorId = doctorId;
}
public String getDoctorId() {
    return doctorId;
}
public void setPatientId(String patientId) {
    this.patientId = patientId;
}
public String getPatientId() {
    return patientId;
}
/**
 * AppointmentInput.java
 * Jarvis (Chia-Yu), Tsai
 * CSCI CSUSB
 */

package pe.form;
import java.lang.*;
import java.util.*;
import javax.servlet.http.*;
import pe.db.DBAppointments;
import pe.util.UASException;
public class AppointmentInput {
    private String appointmentId = "";
    private String appointmentTime = "";
    private String dt = "";
    private String doctorId = "";
    private String patientId = "";

    /** Creates new PersinfoInput */
    public AppointmentInput() {
    }

    public void processRequest(HttpServletRequest req, HttpServletResponse res) throws Exception {
        if (req.getParameter("id") != null) {
            this.appointmentId = req.getParameter("id");
            loadAppointmentInfo();
        } else if (req.getParameter("save") != null && req.getParameter("create") != null) {
            createAppointmentInfo();
            req.getRequestDispatcher("submenu.jsp?id=");
            req.getRequestDispatcher("submenu.jsp?id=");
            req.getRequestDispatcher("submenu.jsp?id=");
            req.getRequestDispatcher("submenu.jsp?id=");
        } else if (req.getParameter("save") != null) {
            updateAppointmentInfo(req);
            req.getRequestDispatcher("submenu.jsp?id=");
            req.getRequestDispatcher("submenu.jsp?id=");
        } else if (req.getParameter("delete") != null) {
            new DBAppointments().deleteAppointment(appointmentId);
            req.getRequestDispatcher("submenu.jsp?id=");
            req.getRequestDispatcher("submenu.jsp?id=");
            req.getRequestDispatcher("submenu.jsp?id=");
            req.getRequestDispatcher("submenu.jsp?id=");
        }
    }

    private void loadAppointmentInfo() throws UASException {
        new DBAppointments().loadAppointment(this);
    }

    private void createAppointmentInfo() throws UASException {
        new DBAppointments().createAppointment(this);
    }

    private void updateAppointmentInfo(HttpServletRequest req) throws UASException {
        setAppointmentId(req.getParameter("appointmentId"));
        setAppointmentTime(req.getParameter("appointmentTime"));
        setDtn(req.getParameter("dt"));
        setDoctorId(req.getParameter("doctorId"));
        setPatientId(req.getParameter("patientId"));
        new DBAppointments().updateAppointment(this);
    }

    public void setAppointmentId(String appointmentId) {
        this.appointmentId = appointmentId;
    }

    public void setAppointmentTime(String appointmentTime) {
        this.appointmentTime = appointmentTime;
    }

    public void setDtn(String dt) {
        this.dt = dt;
    }

    public void setDoctorId(String doctorId) {
        this.doctorId = doctorId;
    }

    public void setPatientId(String patientId) {
        this.patientId = patientId;
    }

    public String getAppointmentId() {
        return appointmentId;
    }

    public String getAppointmentTime() {
        return appointmentTime;
    }

    public String getDtn() {
        return dt;
    }

    public String getDoctorId() {
        return doctorId;
    }

    public String getPatientId() {
        return patientId;
    }
}
/**
 * Title: DrugFilter.java
 * Description: retrieves a list of student given some conditions
 * Jarvis (Chia-Yu), Tsai
 * CSCI CSUSB
 */
package pe.form;
import javax.servlet.http.*;
import java.util.*;
import java.sql.*;
import pe.db.DBDrugs;
import pe.utiLUASException;
public class DrugFilter {
    public DrugFilter() {};
    private String select1;
    private String rxName;
    public ResultSet getDrugList(HttpServletRequest req, HttpServletResponse res)
        throws UASException {
        if (req.getParameter("find") == null)
            return null;
        String constraints = getConstraints();
        return new DBDrugs().getDrugList(constraints);
    }
    private String getConstraints() {
        StringBuffer constraints = new StringBuffer();
        String str = "";
        if (rxName != null && !rxName.equals("") ) {
            if (select1 == null || select1.equals(""))
                str = " NAME LIKE " + rxName + "%";
            else
                str = " NAME LIKE " + rxName + "%";
        }
        constraints.append(str);
        return constraints.toString();
    }
    public String getSelect1() {
        return select1;
    }
    public void setSelect1(String select1) {
        this.select1 = select1;
    }
    public void setRxName(String rxName) {
        this.rxName = rxName;
    }
    public String getRxName() {
        return rxName;
    }
}
package pe.form;
import java.util.*;
import javax.servlet.http.*;
import pe.db.DBDrugs;
import pe.util.UASEException;

public class DrugInfoInput{
    private String drugld = "";
    private String rxName = "";
    private String dose = "";
    private String usage = "";
    /** Creates new PerslnfoInput */
    public DrugInfoInput() {
    }
    public void processRequest(HttpServletRequest req, HttpServletResponse res) throws Exception {
        if (req.getParameter("id") != null) {
            this.drugld = req.getParameter("id");
            loadDrug();
        }
    }
    private void loadDrug() throws UASEException {
        new DBDrugs().loadDrug(this);
    }
    public void setDrugld(String drugld) {
        this.drugld = drugld;
    }
    public String getDrugld() {
        return drugld;
    }
    public void setRxName(String rxName) {
        this.rxName = rxName;
    }
    public String getRxName() {
        return rxName;
    }
    public void setDose(String dose) {
        this.dose = dose;
    }
    public String getDose() {
        return dose;
    }
    public void setUsage(String usage) {
        this.usage = usage;
    }
    public String getUsage() {
        return usage;
    }
}
/**
 * Title: EmployeeFilter.java
 * Description: retrieves a list of student given some conditions
 * Jarvis (Chia-Yu), Tsai
 * "CSCI CSUSB"
 */
package pe.form;
import java.util.*;
import java.sql.*;
import pe.db.DBUser;
import pe.util.UASException;
public class EmployeeFilter {
    private String select1;
    private String select2;
    private String lastName;
    private String firstName;
    private String userid;
    private String ssn;
    public EmployeeFilter() {
    }
    public ResultSet getEmployeeList(HttpServletRequest req,
            HttpServletResponse res)
            throws UASException {
        if (req.getParameter("find").equals(null))
            return null;
        String constraints = getConstraints();
        return new DBUser().getEmployeeList(constraints);
    }
    private String getConstraints() {
        StringBuffer constraints = new StringBuffer();
        String str = "L_NAME=V" + lastName + "%V";
        constraints.append(str);
        if (firstName != null && firstName.equals("")) {
            String select2 = null;
            if (select2.equals("=V") || select2.equals("=*")) {
                str = "F_NAME=V" + firstName + "%V";
                constraints.append(str);
            } else {
                constraints.append(" and " + str);
            }
        } else {
            str = "USER_ID=V" + userid + "%V";
            if (constraints.length() == 0) {
                constraints.append(str);
            } else {
                constraints.append(" and " + str);
            }
        }
        if (ssn != null && ssn.equals("")) {
            str = "ESSN=V" + ssn + "%V";
            if (constraints.length() == 0) {
                constraints.append(str);
            } else {
                constraints.append(" and " + str);
            }
        }
        return constraints.toString();
    }
    public String getSelect1() {
        return select1;
    }
    public void setSelect1(String select1) {
        this.select1 = select1;
    }
    public String getSelect2() {
        return select2;
    }
    public void setSelect2(String select2) {
        this.select2 = select2;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getUserid() {
        return userid;
    }
    public void setUserid(String userid) {
        this.userid = userid;
    }
}
public void setSsn(String ssn) {
    this.ssn = ssn;
}

public String getSsn() {
    return ssn;
}
package pe.form;
import java.lang.*;
import java.util.*;
import javax.servlet.http.*;
import pe.db.DBUser;
import pe.util.UASException;
public class EmployeeInfoInput {
    private String email = "";
    private String password = "";
    private String firstName = "";
    private String midName = "";
    private String lastName = "";
    private String street = "";
    private String city = "";
    private String state = "";
    private String zip = "";
    private String phone = "";
    private String ssn = "";
    private String userid = "";
    private String type = "";
    private String dept = "";

    /** Creates new Perslnfo1nput */
    public EmployeeInfoInput() {
    }

    public void processRequest(HttpServletRequest req, HttpServletResponse res)
    throws Exception {
        if (req.getParameter("id") == null) {
            this.userld = req.getParameter("id");
            loadEmployeeInfo();
        } else if (req.getParameter("save") != null
                && req.getParameter("create") == null) {
            createEmployeeInfo();
            req.getRequestDispatcher("submenu.jsp?id=userid&name=first+\name+\"+lastName").forward(req, res);
        } else if (req.getParameter("save") == null) {
            updateEmployeeInfo(req);  
            req.getRequestDispatcher("submenu.jsp?id=userid&name=first\name+\"+lastName").forward(req, res);
        } else if (req.getParameter("delete") != null) {
            new DBUser().deleteEmployee(userld);
            req.getRequestDispatcher("submenu.jsp?id=userid&name=\first+\name+\"+lastName").forward(req, res);
        }
    }

    private void loadEmployeeInfo() throws UASException {
            new DBUser().loadEmployee(this);
    }

    private void createEmployeeInfo() throws UASException {
            new DBUser().createEmployee(this);
    }

    private void updateEmployeeInfo(HttpServletRequest req) throws UASException {
            setEmail(req.getParameter("email"));
            setFirstName(req.getParameter("firstName"));
            setMidName(req.getParameter("midName"));
            setLastName(req.getParameter("lastName"));
            setStreet(req.getParameter("street"));
            setCity(req.getParameter("city"));
            setState(req.getParameter("state"));
            setZip(req.getParameter("zip"));
            setPhone(req.getParameter("phone"));
            setSidn(req.getParameter("ssn"));
            setDept(req.getParameter("dept"));
            setType(req.getParameter("type"));
            setPassword(req.getParameter("password"));
            new DBUser().updateEmployee(this);
    }

    public void setUserld(String userid) {
            this.userld = userid;
    }

    public void setEmail(String email) {
            if (email != null)
                this.email = email;
            else
                this.email = "";
    }

    public void setPassword(String password) {
            if (password != null)
                this.password = password;
    }
else
    this.password = "";
}

public String getPassword() {
    return password;
}

public void setStreet(String street) {
    if (street != null)
        this.street = street.toUpperCase().trim();
    else
        this.street = "";
}

public String getStreet() {
    return street;
}

public void setCity(String city) {
    if (this.city != null)
        this.city = city.toUpperCase().trim();
    else
        this.city = "";
}

public String getCity() {
    return city;
}

public void setState(String state) {
    if (this.state != null)
        this.state = state.toUpperCase().trim();
    else
        this.state = "";
}

public String getState() {
    return state;
}

public void setZip(String zip) {
    if (zip !="
        this.zip = zip;
    else
        this.zip = "";
}

public String getZip() {
    return zip;
}

public void setPhone(String phone) {
    if (phone != null)
        this.phone = phone;
    else
        this.phone = "";
}

public String getPhone() {
    return phone;
}

public void setSsn(String ssn) {
    if (ssn != null)
        this.ssn = ssn;
    else
        this.ssn = "";
}

public String getSsn() {
    return ssn;
}

public void setDept(String dept) {
    if (dept != null)
        this.dept = dept;
    else
        this.dept = "";
}

public String getDept() {
    return dept;
}

public void setType(String type) {
    if (type != null)
        this.type = type;
    else
        this.type = "";
}

public String getType() {
    return type;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getMidName() {
    return midName;
}

public void setMidName(String midName) {
    if (midName != null)
        this.midName = midName;
    else
        this.midName = "";
}
package pe.form;
import pe.utiLUASException;

public class NoSuchUserException extends UASException {
    public NoSuchUserException(String user) {
        super("" + user + " does not exist!");
        title = "No Such User!";
    }
}
public class PatientFilter {
    public PatientFilter() {
    }

    private String select1;
    private String select2;
    private String lastName;
    private String firstName;
    private String userid;
    private String dob;

    public ResultSet getPatientList(HttpServIetRequest req, HttpServletResponse res)
    throws UASException {
        if (req.getParameter("find") == null) return null;
        String constraints = getConstraints();
        return new DBPatients().getPatientList(constraints);
    }

    private String getConstraints() {
        StringBuffer constraints = new StringBuffer();
        String str = uu;
        if (lastName != null && !lastName.equals("") ) {
            str = "L_NAME=V" + lastName + "V";
            constraints.append(str);
        } else {
            str = "L_NAME LIKE VY%V";
            constraints.append(str);
        }
        if (firstName != null && !firstName.equals("") ) {
            str = "F_NAME=V" + firstName + "V";
            constraints.append(str);
        } else {
            str = "F_NAME LIKE VY%V";
            constraints.append(str);
        }
        if (userid != null && !userid.equals("") ) {
            str = "PATIENT_ID=Y" + userid + "Y";
            constraints.append(str);
        } else {
            str = "PATIENT_ID LIKE YV%V";
            constraints.append(str);
        }
        if (dob != null && !dob.equals("") ) {
            str = "dob=Y" + dob + "Y";
            constraints.append(str);
        } else {
            str = "dob LIKE YV%V";
            constraints.append(str);
        }
        return constraints.toString();
    }

    public String getSelect1() {
        return select1;
    }

    public void setSelect1(String select1) {
        this.select1 = select1;
    }

    public String getSelect2() {
        return select2;
    }

    public void setSelect2(String select2) {
        this.select2 = select2;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setUserid(String userid) {
        this.userid = userid;
    }

    public String getUserid() {
        return userid;
    }
}
public void setDob(String dob) {
    this.dob = dob;
}

public String getDob() {
    return dob;
}
package pe.form;
import java.lang.*;
import java.util.*;
import javax.servlet.http.*;
import pe.db.DBPatients;
import pe.util.UASException;
public class PatientInfoInput {
    private String email = "u";
    private String firstName = "u";
    private String midName = "u";
    private String lastName = "u";
    private String street = "u";
    private String city = "u";
    private String zip = "u";
    private String phone = "u";
    private String ssn = "u";
    private String gender = "u";
    private String userid = "u";
    private String height = "u";
    private String weight = "u";
    private String dob = "u";
    private String history = "u";

    /** Creates new PatientInfoInput */
    public PatientInfoInput() {
    }

    public void processRequest(HttpServletRequest req, HttpServletResponse res) throws Exception {
        if (req.getParameter("id") != null) {
            this.userid = req.getParameter("id");
            loadPatientInfo();
        } else if (req.getParameter("save") != null &&
            req.getParameter("create") != null) {
            createPatientInfo();
            req.getRequestDispatcher("submenu.jsp?id=" + userid + ";name=" + firstName + " + lastName").forward(req, res);
        } else if (req.getParameter("save") != null) {
            updatePatientInfo(req);
            req.getRequestDispatcher("submenu.jsp?id=" + userid + ";name=" + firstName + " + lastName").forward(req, res);
        } else if (req.getParameter("delete") != null) {
            new DBPatients().deletePatient(userid);
            req.getRequestDispatcher("submenu.jsp?id=" + userid + ";name=" + firstName + " + lastName").forward(req, res);
        }
    }

    private void loadPatientInfo() throws UASException {
        new DBPatients().loadPatient(this);
    }

    private void createPatientInfo() throws UASException {
        new DBPatients().createPatient(this);
    }

    private void updatePatientInfo(HttpServletRequest req) throws UASException {
        setEmail(req.getParameter("email"));
        setFirstName(req.getParameter("firstName"));
        setMidName(req.getParameter("midName"));
        setLastName(req.getParameter("lastName"));
        setStreet(req.getParameter("street"));
        setCity(req.getParameter("city"));
        setState(req.getParameter("state"));
        setRoom(req.getParameter("room"));
        setPhone(req.getParameter("phone"));
        setSsn(req.getParameter("ssn"));
        setGender(req.getParameter("gender"));
        setDob(req.getParameter("dob"));
        setHeight(req.getParameter("height"));
        setWeight(req.getParameter("weight"));
        setHistory(req.getParameter("history"));
        new DBPatients().updatePatient(this);
    }

    public void setUserid(String userid) {
        this.userid = userid;}
    public String getUserid() {
        return userid;}

    public void setEmail(String email) {
        if (email != null) {
            this.email = email;
        } else {
            this.email = "u";
        }
    }

    public String getPhone() {
        return phone;
    }

    public void setStreet(String street) {
        if (street != null) {
            this.street = street.toUpperCase().trim();
        } else {
            this.street = "u";
        }
    }
}

package pe.form;
import java.lang.*;
import java.util.*;
import javax.servlet.http.*;
import pe.db.DBPatients;
import pe.util.UASException;
public class PatientInfoInput {
    private String email = "u";
    private String firstName = "u";
    private String midName = "u";
    private String lastName = "u";
    private String street = "u";
    private String city = "u";
    private String zip = "u";
    private String phone = "u";
    private String ssn = "u";
    private String gender = "u";
    private String userid = "u";
    private String height = "u";
    private String weight = "u";
    private String dob = "u";
    private String history = "u";

    /** Creates new PatientInfoInput */
    public PatientInfoInput() {
    }

    public void processRequest(HttpServletRequest req, HttpServletResponse res) throws Exception {
        if (req.getParameter("id") != null) {
            this.userid = req.getParameter("id");
            loadPatientInfo();
        } else if (req.getParameter("save") != null &&
            req.getParameter("create") != null) {
            createPatientInfo();
            req.getRequestDispatcher("submenu.jsp?id=" + userid + ";name=" + firstName + " + lastName").forward(req, res);
        } else if (req.getParameter("save") != null) {
            updatePatientInfo(req);
            req.getRequestDispatcher("submenu.jsp?id=" + userid + ";name=" + firstName + " + lastName").forward(req, res);
        } else if (req.getParameter("delete") != null) {
            new DBPatients().deletePatient(userid);
            req.getRequestDispatcher("submenu.jsp?id=" + userid + ";name=" + firstName + " + lastName").forward(req, res);
        }
    }

    private void loadPatientInfo() throws UASException {
        new DBPatients().loadPatient(this);
    }

    private void createPatientInfo() throws UASException {
        new DBPatients().createPatient(this);
    }

    private void updatePatientInfo(HttpServletRequest req) throws UASException {
        setEmail(req.getParameter("email"));
        setFirstName(req.getParameter("firstName"));
        setMidName(req.getParameter("midName"));
        setLastName(req.getParameter("lastName"));
        setStreet(req.getParameter("street"));
        setCity(req.getParameter("city"));
        setState(req.getParameter("state"));
        setRoom(req.getParameter("room"));
        setPhone(req.getParameter("phone"));
        setSsn(req.getParameter("ssn"));
        setGender(req.getParameter("gender"));
        setDob(req.getParameter("dob"));
        setHeight(req.getParameter("height"));
        setWeight(req.getParameter("weight"));
        setHistory(req.getParameter("history"));
        new DBPatients().updatePatient(this);
    }

    public void setUserid(String userid) {
        this.userid = userid;}
    public String getUserid() {
        return userid;}

    public void setEmail(String email) {
        if (email != null) {
            this.email = email;
        } else {
            this.email = "u";
        }
    }

    public String getPhone() {
        return phone;
    }

    public void setStreet(String street) {
        if (street != null) {
            this.street = street.toUpperCase().trim();
        } else {
            this.street = "u";
        }
    }
}
else
   this.street = null;
}
public String getStreet() {
   return street;
}
public void setCity(String city) {
   if (this.city != null)
      this.city = city.toUpperCase().trim();
   else
      this.city = null;
}
public String getCity() {
   return city;
}
public void setState(String state) {
   if (this.state != null)
      this.state = state.toUpperCase().trim();
   else
      this.state = null;
}
public String getState() {
   return state;
}
public void setGender(String gender) {
   if (this.gender != null)
      this.gender = gender.toUpperCase().trim();
   else
      this.gender = null;
}
public String getGender() {
   return gender;
}
public void setZip(String zip) {
   if (zip != null)
      this.zip = zip;
   else
      this.zip = null;
}
public String getZip() {
   return zip;
}
public void setPhone(String phone) {
   if (phone != null)
      this.phone = phone;
   else
      this.phone = null;
}
public String getPhone() {
   return phone;
}
public void setDob(String dob) {
   if (dob != null)
      this.dob = dob;
   else
      this.dob = null;
}
public String getDob() {
   return dob;
}
public void setSsn(String ssn) {
   if (ssn != null)
      this.ssn = ssn;
   else
      this.ssn = null;
}
public String getSsn() {
   return ssn;
}
public void setHeight(String height) {
   if (height != null)
      this.height = height;
   else
      this.height = null;
}
public String getHeight() {
   return height;
}
public void setWeight(String weight) {
   if (weight != null)
      this.weight = weight;
   else
      this.weight = null;
}
public String getWeight() {
   return weight;
}
public void setHistory(String history) {
   if (history != null)
      this.history = history;
   else
      this.history = null;
}
public String getHistory() {
   return history;
}
public String getFirstName() {
   return firstName;
}
public void setFirstName(String firstName) {
   this.firstName = firstName;
}
public String getLastName() {
   return lastName;
}
public void setLastName(String lastName) {
   this.lastName = lastName;
}
public String getMidName() {
   return midName;
}
public void setMidName(String midName) {
   if (midName != null)
this.midName = midName;
else
    this.midName = '';
}
package pe.form;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;
import pe.db.DBPrescription;
import pe.util.UASException;
public class PrescriptionFilter {
    public PrescriptionFilter() {
    }
    private String prescriptionId;
    private String patientId;
    public ResultSet getPrescriptionStatus(HttpServletRequest req, HttpServletResponse res) throws UASException {
        if (req.getParameter("find") == null) {
            return null;
        }
        String constraints = getConstraints();
        return new DBPrescription().loadPrescriptionStatus(constraints);
    }
    private String getConstraints() {
        StringBuffer constraints = new StringBuffer();
        String str = "PRESCRIPTION_ID = " + prescriptionId + " ;
        if (! prescriptionId.equals("")) {
            if (constraints.length() == 0) {
                constraints.append(str);
            } else {
                constraints.append("and" + str);
            }
        } else {
            return constraints.toString();
        }
        public void setPrescriptionId(String prescriptionId) {
            this.prescriptionId = prescriptionId;
        }
        public String getPrescriptionId() {
            return prescriptionId;
        }
        public void setPatientId(String patientId) {
            this.patientId = patientId;
        }
        public String getPatientId() {
            return patientId;
        }
    }
}
/* PrescriptionlnfoInput.java  
Jarvis (Chia-Yu), Tsai  
*CSCI CSUSB  
*/

package pe.form;
import java.util.*;
import javax.servlet.http.*;
import pe.db.DBPrescription;
import pe.util.UASException;

public class PrescriptionlnfoInput  
{
private String prescriptionld = "";
private String rxName = "";
private String dose = "";
private String dispense = "";
private String usage = "";
private String substitute = "";
private String status = "";
private String refill = "";
private String doctorid = "";
private String doctorid = "";
private String patientld = "";
private String pharmacistid = "";
private String examld = "";
private String notes = "";

/** Creates new PerslnfoInput */
public PrescriptionlnfoInput()  
{  
}

public void processRequest(HttpServletRequest req, HttpServletResponse res)  
throws Exception  
{  
if (req.getParameter("id") != null)  
{  
this.prescriptionld = req.getParameter("id");
loadPrescription();  
}  
else if (req.getParameter("save") != null)  
{  
makePrescription();
req.getRequestDispatcher("submenu.jsp?prescriptionId=" + prescriptionld).forward(req, res);
}  
else if (req.getParameter("update") != null)  
{  
modifyPrescription(req);
req.getRequestDispatcher("submenu.jsp?prescriptionId=" + prescriptionld).forward(req, res);
}  
else if (req.getParameter("modify") != null)  
{  
updatePrescriptionStatus(req);
req.getRequestDispatcher("submenu.jsp?prescriptionId=" + prescriptionld).forward(req, res);
}  
else if (req.getParameter("delete") != null)  
{  
newDBPrescription().deletePrescription(prescriptionld);
req.getRequestDispatcher("submenu.jsp?prescriptionId=" + prescriptionld).forward(req, res);
}

private void loadPrescription() throws UASException  
{  
new DBPrescription().loadPrescription(this);
}

private void makePrescription() throws UASException  
{  
new DBPrescription().makePrescription(this);
}

private void modifyPrescription(HttpServletRequest req) throws UASException  
{  
setPrescriptionld(req.getParameter("prescriptionld"));
setRxName(req.getParameter("rxName"));
setDose(req.getParameter("dose"));
setDispense(req.getParameter("dispense"));
setUsage(req.getParameter("usage"));
setSubstitute(req.getParameter("substitute"));
setRefill(req.getParameter("refill"));
new DBPrescription().modifyPrescription(this);
}

private void updatePrescriptionStatus(HttpServletRequest req) throws UASException  
{  
setPrescriptionld(req.getParameter("prescriptionld"));
setPharmacistid(req.getParameter("pharmacistid"));
setStatus(req.getParameter("status"));
setRefill(req.getParameter("refill"));
new DBPrescription().updatePrescriptionStatus(this);
}

public void setDoctorId(String doctorid)  
{  
this.doctorid = doctorid;
}

public String getDoctorId()  
{  
return doctorid;
}

public void setExaId(String examld)  
{  
this.exaId = examld;
}

public String getExaId()  
{  
return examld;
}

public void setNotes(String notes)  
{  
this.notes = notes;
}

public String getNotes()  
{  
return notes;
}

public void setPatientId(String patientld)  
{  
this.patientld = patientld;
}

public String getPatientId()  
{  
return patientld;
}

public void setPharmacistId(String pharmacistid)  
{  
this.pharmacistid = pharmacistid;
}  

}  

119
public String getPharmacistId() {
    return pharmacistId;
}
public void setPrescriptionId(String prescriptionId) {
    this.prescriptionId = prescriptionId;
}
public String getPrescriptionId() {
    return prescriptionId;
}
public String getRxName(String rXName) {
    this.rXName = rXName;
}
public String getRxName() {
    return rXName;
}
public void setDose(String dose) {
    this.dose = dose;
}
public String getDose() {
    return dose;
}
public void setDispense(String dispense) {
    this.dispense = dispense;
}
public String getDispense() {
    return dispense;
}
public void setUsage(String usage) {
    this.usage = usage;
}
public String getUsage() {
    return usage;
}
public void setSubstitute(String substitute) {
    this.substitute = substitute;
}
public String getSubstitute() {
    return substitute;
}
public void setStatus(String status) {
    this.status = status;
}
public String getStatus() {
    return status;
}
public void setRefill(String refill) {
    this.refill = refill;
}
public String getRefill() {
    return refill;
}
public class RefillFilter {
    private String prescriptionid;
    private String patientld;

    public ResultSet findRefillPrescription(HttpServletRequest req, HttpServletResponse res)
            throws UASEException {
        if (req.getParameter("find") == null) {
            return null;
        }
        String constraints = getConstraints();
        return new DBPrescription().loadRefillPrescription(constraints);
    }

    private String getConstraints() {
        StringBuffer constraints = new StringBuffer();
        String str = "u;
        constraints.append("refill <> 0");
        if (prescriptionid != null && !prescriptionid.equals("")) {
            str = " PRESCRIPTION_ID =" + prescriptionid + ";
            if (constraints.length() == 0) constraints.append(str);
            else constraints.append(" and " + str);
        }
        if (patientld != null && !patientld.equals("")) {
            str = " PATIENT =" + patientld + ";
            if (constraints.length() == 0) constraints.append(str);
            else constraints.append(" and " + str);
        }
        return constraints.toString();
    }

    public void setPrescriptionid(String prescriptionid) {
        this.prescriptionid = prescriptionid;
    }

    public String getPrescriptionid() {
        return prescriptionid;
    }

    public void setPatientld(String patientld) {
        this.patientld = patientld;
    }

    public String getPatientld() {
        return patientld;
    }
}
package pe.util;
public class UASException extends Exception {
    protected String title = "Sorry";
    public UASException() {
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public UASException(String msg) {
        super(msg);
    }
    public UASException(String msg, String title) {
        super(msg);
        this.title = title;
    }
}
<html>
<body bgcolor="#FFFFFF">
<table border="0" cellspacing="0" borderpadding="0">
<tr>
<td align="center">

</td></tr>
</table>
</body>
</html>
<table>
<thead>
<tr>
<th>Name</th>
<th>First</th>
<th>Middle</th>
<th>Last</th>
<th>Admin</th>
<th>User</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
<td>Doe</td>
<td>Johnson</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Welcome: Administrator

User ID: Admin

<font color="#3366FF">Welcome: Administrator</font>

Last Name: Doe

First Name: John

Middle: Smith

SSN (123-45-6789): 

Filter: Filter property: lastName
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
<td>Value 4</td>
</tr>
</tbody>
</table>

For any question and suggestion, please contact adminstrator or (909)333-2727.
<table>
<thead>
<tr>
<th>Width</th>
<th>Height</th>
<th>Background Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td>18%</td>
<td>#F08080</td>
</tr>
<tr>
<td>15%</td>
<td>18%</td>
<td>#3366FF</td>
</tr>
<tr>
<td>15%</td>
<td>18%</td>
<td>#F08080</td>
</tr>
<tr>
<td>15%</td>
<td>18%</td>
<td>#3366FF</td>
</tr>
<tr>
<td>15%</td>
<td>18%</td>
<td>#F08080</td>
</tr>
<tr>
<td>15%</td>
<td>18%</td>
<td>#3366FF</td>
</tr>
</tbody>
</table>

*For any question and suggestion, please contact administrator@jarviscyt@hotmail.com or (909)333-2727.*

Welcome: Administrator Home

Administrator: Administrator

Administrator's last name: Administrator

For any question and suggestion, please contact administrator@jarviscyt@hotmail.com or (909)333-2727.
<table border="0" cellspacing="0" cellpadding="5" width="75%" align="center">
<tr align="center">
<td colspan="4" align="center" height="356" width="19%">
</td>
</tr>
<tr align="center">
<td align="center" height="18" width="51%" value="#FF0000" noshade="noshade">Welcome: Administrator</td>
<td align="center" height="18" width="51%" value="#FF0000" noshade="noshade">Employee Information</td>
</tr>
</table>

<form method="post" action="createAccount.jsp">

<INPUT TYPE="text" NAME="firstName" SIZE="23" value=""><input type="hidden" name="authBean" value=""><input type="hidden" name="personInfo" property="firstName" value=""><input type="hidden" name="personInfo" property="lastName" value=""><input type="hidden" name="personInfo" property="middleName" value=""></form>

</div align="right">
<table>
<thead>
<tr>
<th>State</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>Obstetrics &amp; Gynecology</td>
</tr>
<tr>
<td>AK</td>
<td>Emergency Medicine</td>
</tr>
<tr>
<td>AZ</td>
<td>Trauma Surgery</td>
</tr>
<tr>
<td>AR</td>
<td>Endocrinology</td>
</tr>
<tr>
<td>CA</td>
<td>Medical Physics</td>
</tr>
<tr>
<td>CO</td>
<td>Mental Health Center</td>
</tr>
<tr>
<td>CT</td>
<td>General Surgery</td>
</tr>
<tr>
<td>DE</td>
<td>Hand &amp; Leprosy Surgery</td>
</tr>
<tr>
<td>DC</td>
<td>General Surgery</td>
</tr>
<tr>
<td>FL</td>
<td>Emergency Department</td>
</tr>
<tr>
<td>GA</td>
<td>Endocrine</td>
</tr>
<tr>
<td>KY</td>
<td>Emergency Medicine</td>
</tr>
<tr>
<td>KS</td>
<td>Endocrinology</td>
</tr>
<tr>
<td>LA</td>
<td>Medical Surgery</td>
</tr>
<tr>
<td>MD</td>
<td>Mental Health Center</td>
</tr>
<tr>
<td>MA</td>
<td>Medical Physics</td>
</tr>
<tr>
<td>MI</td>
<td>General Surgery</td>
</tr>
<tr>
<td>MN</td>
<td>Emergency Department</td>
</tr>
<tr>
<td>MS</td>
<td>Medical Surgery</td>
</tr>
<tr>
<td>MO</td>
<td>General Surgery</td>
</tr>
<tr>
<td>MT</td>
<td>General Surgery</td>
</tr>
<tr>
<td>NE</td>
<td>Special Surgery</td>
</tr>
<tr>
<td>NH</td>
<td>General Surgery</td>
</tr>
<tr>
<td>NJ</td>
<td>General Surgery</td>
</tr>
<tr>
<td>NY</td>
<td>General Surgery</td>
</tr>
<tr>
<td>OH</td>
<td>Special Surgery</td>
</tr>
<tr>
<td>OK</td>
<td>Emergency Medicine</td>
</tr>
<tr>
<td>OR</td>
<td>General Surgery</td>
</tr>
<tr>
<td>PA</td>
<td>General Surgery</td>
</tr>
<tr>
<td>RI</td>
<td>General Surgery</td>
</tr>
<tr>
<td>SC</td>
<td>General Surgery</td>
</tr>
<tr>
<td>SD</td>
<td>General Surgery</td>
</tr>
<tr>
<td>TN</td>
<td>Emergency Department</td>
</tr>
<tr>
<td>TX</td>
<td>General Surgery</td>
</tr>
<tr>
<td>UT</td>
<td>General Surgery</td>
</tr>
<tr>
<td>VA</td>
<td>General Surgery</td>
</tr>
<tr>
<td>WI</td>
<td>General Surgery</td>
</tr>
<tr>
<td>WV</td>
<td>General Surgery</td>
</tr>
<tr>
<td>WY</td>
<td>General Surgery</td>
</tr>
</tbody>
</table>

**Contact Information:**

- **First Name:** [java:getProperty name="personInfo" property="firstName" size="63"]
- **Last Name:** [java:getProperty name="personInfo" property="lastName" size="63"]
- **User Name:** [java:getProperty name="personInfo" property="username" size="63"]
- **Password:** [java:getProperty name="personInfo" property="password" size="63"]
- **Street Address:** [java:getProperty name="personInfo" property="streetAddress" size="63"]
- **City:** [java:getProperty name="personInfo" property="cityName" size="63"]
- **State:** [java:getProperty name="personInfo" property="state" size="63"]
- **Phone:** [java:getProperty name="personInfo" property="phoneNumber" size="63"]
- **Email:** [java:getProperty name="personInfo" property="emailAddress" size="63"]
- **Social Security Number:** [java:getProperty name="personInfo" property="ssn" size="63"]
<SELECT><OPTION>MO</OPTION><OPTION>MT</OPTION><OPTION>NC</OPTION><OPTION>ND</OPTION><OPTION>NE</OPTION><OPTION>NH</OPTION><OPTION>NI</OPTION><OPTION>NM</OPTION><OPTION>NY</OPTION><OPTION>NV</OPTION><OPTION>OH</OPTION><OPTION>OK</OPTION><OPTION>OR</OPTION><OPTION>PA</OPTION><OPTION>RI</OPTION><OPTION>SC</OPTION><OPTION>SD</OPTION><OPTION>TN</OPTION><OPTION>TX</OPTION><OPTION>UT</OPTION><OPTION>VT</OPTION><OPTION>VA</OPTION><OPTION>WA</OPTION><OPTION>WV</OPTION><OPTION>WI</OPTION><OPTION>WY</OPTION></SELECT>

<font size="2">Zip</font> <INPUT TYPE="text" NAME="zip" SIZE="10" value="<jsp:getProperty name="personInfo" property="zip"/>">
</TD></TR>

<TR BGColor="#FFFFFF">
<TD HEIGHT="30" BORDERCOLOR="#000000"><font size="2">Phone number:</font>
<INPUT TYPE="text" NAME="phone" SIZE="20" value="<jsp:getProperty name="personInfo" property="phone"/>">
</TD></TR>

<TR ALIGN="CENTER" TD >% if (create) {%>
<input type="hidden" name="create" value="true">% } %>
<INPUT TYPE="submit" NAME="save" VALUE="% if (create) "Save": Update":%>
<INPUT TYPE="reset" NAME="cancel" VALUE="reset">% if (create) {%>
<input type="submit" name="delete" value="delete"><%}
</TD></TR>

<br align="center" height="19">
<font size="1">For any question and suggestion, please contact <a href="mailto:jarviscyt@hotmail.com">administrator</a> or (909)333-2727</font>
<td></td>
</table></td></tr></table>

<!— #EndEditable —>
<p>&nbsp;</p>
</td>
</tr></table></td>
</table></td>
</tr></table></td>
</table></td>
</tr></table>

131
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>Data 1</td>
<td>Data 2</td>
<td>Data 3</td>
</tr>
<tr>
<td>Data 4</td>
<td>Data 5</td>
<td>Data 6</td>
</tr>
<tr>
<td>Data 7</td>
<td>Data 8</td>
<td>Data 9</td>
</tr>
</tbody>
</table>

Additional notes:

- The table contains three columns and multiple rows of data.
- The table is aligned to the center of the page.
<table>
<thead>
<tr>
<th>TD BGCOLOR=&quot;#FFFFFF&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (rs != null) {}</td>
</tr>
<tr>
<td>&lt;TABLE cellSpacing=3 cellPadding=1 border=1 align=center</td>
</tr>
<tr>
<td>&lt;TR align=middle BGCOLOR=&quot;#3366FF&quot; &gt;</td>
</tr>
<tr>
<td>&lt;TD width=&quot;49%&quot; BORDERCOLOR=&quot;#000000&quot; HEIGHT=&quot;69&quot; &gt;</td>
</tr>
<tr>
<td>&lt;%= if (rs1 == null) {} %&gt;</td>
</tr>
<tr>
<td>&lt;TABLE cellspacing=3 cellpadding=1 border=1 align=center</td>
</tr>
<tr>
<td>&lt;TR align=middle BGCOLOR=&quot;#3366FF&quot; &gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;tD&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Patient&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Rx Name&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Dose&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Disp&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Usage&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Sub&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Status&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Refill&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;TD   &lt;font color=&quot;#FFFFFF&quot; size=&quot;2&quot;&gt;Modify Status&lt;/font&gt; &lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;/TR&gt;</td>
</tr>
<tr>
<td>&lt;/TABLE&gt;</td>
</tr>
<tr>
<td>&lt;/TD&gt;</td>
</tr>
<tr>
<td>&lt;/TR&gt;</td>
</tr>
<tr>
<td>&lt;/TABLE&gt;</td>
</tr>
<tr>
<td>&lt;/td&gt;</td>
</tr>
<tr>
<td>&lt;/tr&gt;</td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
</tr>
<tr>
<td>&lt;/td&gt;</td>
</tr>
<tr>
<td>&lt;/tr&gt;</td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
</tr>
<tr>
<td>&lt;/body&gt;</td>
</tr>
<tr>
<td>&lt;/html&gt;</td>
</tr>
</tbody>
</table>
<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>

<tr><td><div align="right"><div value="like" name='filter'>filter.getSelect2().equals(name-'filter')&nbsp;&nbsp;&nbsp;</div></div></td></tr>
<input type="text" name="dob" size="11" value=""/>

<form>
  &lt;tr align="center"&gt;&lt;td&gt;&lt;input type="submit" name="find" value="find"&gt;&lt;/td&gt;&lt;/tr&gt;
</form>

<tr bgcolor="#FFFFFF" align="center">
  &lt;td width="49%" bordercolor="#000000" height="21"&gt;
  &lt;table width="98%" border=1 cellspacing=3 cellpadding=5 align=center&gt;
    &lt;tr bgcolor="#3366FF" align=center&gt;
      &lt;td&gt;&lt;b&gt;&lt;font size=-1 color=#FFFFFF&gt;Patient ID&lt;/font&gt;&lt;/b&gt;&lt;/td&gt;
      &lt;td&gt;&lt;b&gt;&lt;font size=-1 color=#FFFFFF&gt;Last Name&lt;/font&gt;&lt;/b&gt;&lt;/td&gt;
      &lt;td&gt;&lt;font size=-1 color=#FFFFFF&gt;Middle&lt;/font&gt;&lt;/td&gt;
      &lt;td&gt;&lt;font size=-1 color=#FFFFFF&gt;First Name&lt;/font&gt;&lt;/td&gt;
      &lt;td&gt;&lt;font size=-1 color=#FFFFFF&gt;View Information&lt;/font&gt;&lt;/td&gt;
    &lt;/tr&gt;
    &lt;tr align=center&gt;
      &lt;td&gt;&lt;font size=2&gt;&lt;%=rs.getString(1)%&gt;&lt;/font&gt;&lt;/td&gt;
      &lt;td&gt;&lt;font size=2&gt;&lt;%=rs.getString(2)%&gt;&lt;/font&gt;&lt;/td&gt;
      &lt;td&gt;&lt;font size=2&gt;&lt;%=rs.getString(3)%&gt;&lt;/font&gt;&lt;/td&gt;
      &lt;td&gt;&lt;font size=2&gt;&lt;%=rs.getString(4)%&gt;&lt;/font&gt;&lt;/td&gt;
      &lt;td&gt;&lt;form method="post" action=""&lt;%=request.getContextPath()%>+authBean.getMenuPage()%&gt;"&gt;&lt;input type="submit" value="View Information"&gt;&lt;/form&gt;&lt;/td&gt;
    &lt;/tr&gt;
  &lt;/table&gt;
&lt;/td&gt;&lt;/tr&gt;
&lt;/table&gt;&lt;/tr&gt;
&lt;tr valign=bottom&gt;
 &lt;td align=center height=166&gt;
 &lt;hr width="100%" color="#0085B0" size=1 noshade&gt;
 &lt;font size=1&gt;For any question and suggestion, please contact &lt;a href="mailto:jarviscyt@hotmail.com">adminstrator</a> or (909)333-2727&lt;/font&gt;&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;
&lt;table&gt;
  &lt;tr&gt;
    &lt;td align=center height=166&gt;
    &lt;hr width="100%" color="#000000" size=1 noshade&gt;
    &lt;font size=1&gt;For any question and suggestion, please contact &lt;a href="mailto:jarviscyt@hotmail.com">adminstrator</a> or (909)333-2727&lt;/font&gt;&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;
&lt;/body&gt;
&lt;/html&gt;
<table>
<thead>
<tr>
<th>Pharmacist</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>Administrator</td>
</tr>
</tbody>
</table>

For any question and suggestion, please contact info@pharmacist.com or (909)333-2727.
<table>
<thead>
<tr>
<th>Pharmacist</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="../iogout.jsp?logout=true">Logout</a></td>
<td></td>
</tr>
</tbody>
</table>

### Patrol Report

- **Date:** 07/15/2021
- **Time:** 10:00 AM
- **Location:** Pharmacy

**Incident Brief:**

- **Description:** Patient reported experiencing side effects from medication.
- **Action:** Prescribe additional medication.

---

**Notes:**

- **Action Taken:** Prescribed medication to be taken immediately.
- **Follow-up:** Patient to report back in 24 hours.

---

**Signatures:**

- **Dr. Smith**: Medical Director
- **Pharmacist**: Samuel
- **Patient**: John Doe
| Pharmacist ID: | <input type="text" name="pharmacistld" size="15" value=""><jsp:getProperty name="personlnfo" property="pharmacistld7"/>
| Prescription ID: | <jsp:getProperty name="personlnfo" property="prescriptionId7"/>
| Medication Name: | <jsp:getProperty name="personlnfo" property="rxName7"/>
| Dose: | <jsp:getProperty name="personlnfo" property="dose7"/>
| Dispense: | <jsp:getProperty name="personlnfo" property="dispense7"/>
| Usage: | <jsp:getProperty name="personlnfo" property="usage7"/>
| Substitute: | <jsp:getProperty name="personlnfo" property="substitute7"/>
| Status: | <select name="status">
  <option>Received</option>
  <option>Processing</option>
  <option>Completed</option>
  <option>Picked-up</option>
  <option>Refill Processing</option>
  <option>Refill Completed</option>
</select>
| Refill: | <input type="text" name="refill" size="15" value=""><jsp:getProperty name="personlnfo" property="refill7"/>

For any question and suggestion, please contact <a href="mailto:jarviscyt@hotmail.com">adminstrator</a> or (909)333-2727.
<table width="100%" border="0" cellspacing="0" cellpadding="0">  
  <tr>    
    <td align="center">      
      <table width="100%" border="0" cellspacing="0" cellpadding="10" align="center" bordercolor="#FFFF00">        
        <tbody>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
          <tr bgcolor="#3366FF">            
            <td width="15" height="18" bgcolor="#3366FF">  
              <div align="right"></div>            
          </tr>          
        </tbody>      
    </table>  
  </td>  
</tr></table>
<table>
<thead>
<tr>
<th>ID</th>
<th>Rx Name</th>
<th>Dose</th>
<th>Treatment</th>
<th>Indication</th>
<th>Available</th>
</tr>
</thead>
</table>

For any question and suggestion, please contact <a href="mailto:jarviscy@hotmail.com">administrator</a> or (909)333-2727.
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Gender</th>
<th>Race</th>
<th>Blood Type</th>
<th>Weight</th>
<th>Height</th>
<th>Blood Pressure</th>
<th>Meds</th>
<th>Other Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Patient A</td>
<td>32</td>
<td>Male</td>
<td>White</td>
<td>B</td>
<td>150</td>
<td>6.0</td>
<td>120/80</td>
<td>AL</td>
<td>None</td>
</tr>
<tr>
<td>102</td>
<td>Patient B</td>
<td>45</td>
<td>Female</td>
<td>Black</td>
<td>O</td>
<td>165</td>
<td>5.5</td>
<td>130/90</td>
<td>B/P</td>
<td>None</td>
</tr>
<tr>
<td>103</td>
<td>Patient C</td>
<td>28</td>
<td>Male</td>
<td>Asian</td>
<td>A</td>
<td>170</td>
<td>6.2</td>
<td>115/75</td>
<td>C</td>
<td>None</td>
</tr>
</tbody>
</table>

*Note: For Patient D, the blood pressure is 140/90. Other information is not available.*
| **Patient Name:** | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n... |
<table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr align="center">
    <td bgcolor="#ffffff" width="10" border="0" cellspacing="2" cellpadding="0">
      <div>
        <div>
          <div>
            <div>
              <div>
                <div>
                  <div>
                    <div>
                      <div>
                        <div>
                          <div>
                            <div>
                              <div>
                                <div>
                                  <div>
                                    <div>
                                      <div>
                                        <div>
                                          <div>
                                            <div>
                                              <div>
                                                <div>
                                                  <div>
                                                    <div>
                                                      <div>
                                                        <div>
                                                          <div>
                                                            <div>
                                                              <div>
                                                                <div>
                                                                  <div>
                                                                    <div>
                                                                      <div>
                                                                        <div>
                                                                          <div>
                                                                            <div>
                                                                              <div>
                                                                                <div>
                                                                                  <div>
                                                                                      <div>
                                                                                        <div>
                                                                                      </div>
                                                                                  </div>
                                                                                </div>
                                                                              </div>
                                                                            </div>
                                                                          </div>
                                                                    </div>
                                                                  </div>
                                                                </div>
                                                              </div>
                                                            </div>
                                                          </div>
                                                        </div>
                                                      </div>
                                                    </div>
                                                  </div>
                                                </div>
                                              </div>
                                            </div>
                                          </div>
                                        </div>
                                      </div>
                                    </div>
                                  </div>
                                </div>
                              </div>
                            </div>
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </td>
  </tr>
</table>
<table width="100%" border="0" cellspacing="0" cellpadding="0" align="center" width="60%">
<tr><td bgcolor="#666666" align="center" width="100%" border="1" noshade>
<table>
<thead>
<tr>
<th>Name</th>
<th>Prescription List</th>
<th>Status</th>
<th>Rx Name</th>
<th>Close</th>
<th>Disp</th>
<th>Usage</th>
<th>Subfect</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
</table>

<div align="right"></div>
</body>
</html>
<table>
<thead>
<tr>
<th>PRESCRIPTION_ID</th>
<th>PATIENT</th>
<th>RX_NAME</th>
<th>DOSE</th>
<th>SUBSTITUTE</th>
<th>STATUS</th>
<th>REFILL</th>
<th>PHARMACIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRESCRIPTION_ID</td>
<td>PATIENT</td>
<td>RX_NAME</td>
<td>DOSE</td>
<td>SUBSTITUTE</td>
<td>STATUS</td>
<td>REFILL</td>
<td>PHARMACIST</td>
</tr>
</tbody>
</table>

For any question and suggestion, please contact <a href="mailto:jarviscyt@hotmail.com">adminstrator</a> or (909)333-2727.
<table>
<thead>
<tr>
<th>No.</th>
<th>First Name</th>
<th>Last Name</th>
<th>Phone</th>
<th>E-mail</th>
<th>Gender</th>
<th>Date of Birth</th>
<th>Prescriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Smith</td>
<td>555-1234</td>
<td><a href="mailto:john.smith@email.com">john.smith@email.com</a></td>
<td>Male</td>
<td>1990-01-01</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>Doe</td>
<td>555-5678</td>
<td><a href="mailto:jane.doe@email.com">jane.doe@email.com</a></td>
<td>Female</td>
<td>1985-02-15</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Michael</td>
<td>Johnson</td>
<td>555-9087</td>
<td><a href="mailto:michael.johnson@email.com">michael.johnson@email.com</a></td>
<td>Male</td>
<td>1992-03-20</td>
<td>4</td>
</tr>
</tbody>
</table>

*Note: The table above represents a list of patients with their personal information, including their first and last name, contact details, and the number of prescriptions they have been prescribed.*
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Patient ID</td>
</tr>
<tr>
<td>Name:</td>
<td>Rx Name</td>
</tr>
<tr>
<td>Name:</td>
<td>Dose</td>
</tr>
<tr>
<td>Name:</td>
<td>Usage</td>
</tr>
<tr>
<td>Name:</td>
<td>Notes</td>
</tr>
<tr>
<td>Name:</td>
<td>Refill</td>
</tr>
<tr>
<td>Name:</td>
<td>Substitute</td>
</tr>
</tbody>
</table>

For any question and suggestion, please contact administrator@nmail.com or (800)333-2727.
<table>
<thead>
<tr>
<th>Prescription</th>
<th>Doctor Name</th>
<th>Email</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prescription 1</td>
<td>John Smith</td>
<td><a href="mailto:john.smith@example.com">john.smith@example.com</a></td>
<td>123-456-7890</td>
</tr>
<tr>
<td>Prescription 2</td>
<td>Jane Doe</td>
<td><a href="mailto:jane.doe@example.com">jane.doe@example.com</a></td>
<td>098-765-4321</td>
</tr>
</tbody>
</table>

- Click on the Edit button to edit the prescription.
<table border="0" cellspacing="0" cellpadding="6" align="center" height="377">

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| <td height="22" font-size="2"><input type="hidden" name="patientId" Size="35" value="jsp:getProperty name="personInfo" property="patientId"/>
</td> |   |
| <td height="22" font-size="2"><input type="hidden" Name="prescriptionId" size="30" value="jsp:getProperty name="personInfo" property="prescriptionId"/>
</td> |   |
| <td height="22">Rx Name: <input type="text" NAME="rxName" Size="35" value="jsp:getProperty name="personInfo" property="rxName"/>
</td> |   |
| <td height="22">Dose: <input type="text" NAME="dose" SIZE="15" VALUE="jsp:getProperty name="personInfo" property="dose"/>
</td> |   |
| <td height="22">Dispense: <select NAME="dispense">
  <option>q.s. for 30 days</option>
  <option>q.s. for 60 days</option>
  <option>q.s. for 12 months</option>
</select></td> |   |
| <td height="22">Usage: <input type="text" NAME="usage" SIZE="40" value="jsp:getProperty name="personInfo" property="usage"/>
</td> |   |
| <td height="22">Substitute: <select NAME="substitute">
  <option>Y</option>
  <option>N</option>
</select></td> |   |
| <td height="22">Refill: <select NAME="refill">
  <option></option>
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
  <option>6</option>
  <option>7</option>
  <option>8</option>
  <option>9</option>
</select></td> |   |


</table>

For any question and suggestion, please contact <a href="mailto:jarviscyt@hotmail.com">adminstrator</a> or (909)333-2727

164
<table>
<thead>
<tr>
<th>Rx Name</th>
<th>Dose</th>
<th>Treatment</th>
<th>Availability</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For any question and suggestion, please contact administrator or (909)333-2727.
<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>ID</th>
<th>Patient Name</th>
<th>Appointment Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>19:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>19:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>21:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>21:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>22:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>22:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Nurse</td>
<td>494</td>
<td>15</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Welcome:

- [Change Password](#)
<table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr align="center">
    <td align="center">
      <table width="100%" border="0" cellspacing="0" cellpadding="0">
        <tr bgcolor="#3366FF">
          <td width="15" height="18" bgcolor="#3366FF" align="center">
            <font face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#3366FF"><a href="nurse/inputPatientInfo.jsp?new=true">New Patient</a></font>
          </td>
          <td width="18" height="18" bgcolor="#3366FF" align="center">
            <font face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#3366FF"><a href="nurse/statusList.jsp">Status</a></font>
          </td>
          <td width="18" height="18" bgcolor="#3366FF" align="center">
            <font face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#3366FF"><a href="nurse/list.jsp">Existing Patient</a></font>
          </td>
          <td width="18" height="18" bgcolor="#3366FF" align="center">
            <font face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#3366FF"><a href="nurse/nurse.jsp">Home</a></font>
          </td>
          <td width="18" height="18" bgcolor="#3366FF" align="center">
            <font face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#3366FF"><a href="nurse/inputPrescription.jsp">Prescription</a></font>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
<table>
<thead>
<tr>
<th>Nurse</th>
<th>Change Password</th>
<th>Logout</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>First Name</td>
<td>&lt;jsp:getProperty name=&quot;personInfo&quot; property=&quot;firstName&quot; /&gt;</td>
<td></td>
</tr>
<tr>
<td>Last Name</td>
<td>&lt;jsp:getProperty name=&quot;personInfo&quot; property=&quot;lastName&quot; /&gt;</td>
<td></td>
</tr>
<tr>
<td>Gender</td>
<td>&lt;OPTION value=&quot;M&quot;&gt;M&lt;/OPTION&gt;</td>
<td>&lt;OPTION value=&quot;F&quot;&gt;F&lt;/OPTION&gt;</td>
</tr>
<tr>
<td>DOB</td>
<td>&lt;input type=&quot;text&quot; name=&quot;dob&quot; value='&lt;jsp:getProperty name=&quot;personInfo&quot; property=&quot;dob&quot; /&gt;' /&gt;</td>
<td></td>
</tr>
<tr>
<td>SSN</td>
<td>&lt;input type=&quot;password&quot; name=&quot;ssn&quot; value='&lt;jsp:getProperty name=&quot;personInfo&quot; property=&quot;ssn&quot; /&gt;' /&gt;</td>
<td></td>
</tr>
<tr>
<td>Height</td>
<td>&lt;input type=&quot;text&quot; name=&quot;height&quot; value='&lt;jsp:getProperty name=&quot;personInfo&quot; property=&quot;height&quot; /&gt;' /&gt;</td>
<td></td>
</tr>
</tbody>
</table>

177
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight (pounds)</td>
<td>&lt;input type=&quot;text&quot; name=&quot;weight&quot; size=&quot;18&quot; value=&quot;58&quot; /&gt;</td>
</tr>
<tr>
<td>History</td>
<td>textarea rows=5 cols=55</td>
</tr>
<tr>
<td>Street</td>
<td>&lt;input type=&quot;text&quot; name=&quot;street&quot; size=&quot;55&quot; value=&quot;&quot; /&gt;</td>
</tr>
<tr>
<td>City</td>
<td>&lt;input type=&quot;text&quot; name=&quot;city&quot; size=&quot;27&quot; value=&quot;&quot; /&gt;</td>
</tr>
<tr>
<td>State</td>
<td>&lt;select name=&quot;state&quot;&gt;</td>
</tr>
<tr>
<td>Zip</td>
<td>&lt;input type=&quot;text&quot; name=&quot;zip&quot; size=&quot;10&quot; value=&quot;&quot; /&gt;</td>
</tr>
<tr>
<td>Phone</td>
<td>&lt;input type=&quot;text&quot; name=&quot;phone&quot; size=&quot;21&quot; value=&quot;&quot; /&gt;</td>
</tr>
</tbody>
</table>
<tr>
<%}
</tr>
</table>

```
<%@ page import="java.util.*, pe.db.*, pe.auth.AuthBean" %>
 <%@ page contentType="text/html" %>
<%@ page isErrorPage="true" %>
<%@ page import="java.sql.ResultSet, pe.db.*", page errorPage="errorPage.jsp" %>
<% pe.auth.AuthBean authBean = new pe.auth.AuthBean(); %>
<% pe.db.Connection conn = connFactory.getConnection(); %>
<% pe.db.Statement stmt = conn.createStatement(); %>
<% pe.db.ResultSet rs = stmt.executeQuery(query); %>
<% ResultSet rs = filter.getAppointmentList(request, response); %>
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
<th>Column 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 1</td>
<td>Data 2</td>
<td>Data 3</td>
<td>Data 4</td>
<td>Data 5</td>
<td>Data 6</td>
</tr>
<tr>
<td>Data 7</td>
<td>Data 8</td>
<td>Data 9</td>
<td>Data 10</td>
<td>Data 11</td>
<td>Data 12</td>
</tr>
</tbody>
</table>

**Notes:**
- Table width: 100%
- Border: 0
- Cellspacing: 0
- Cellpadding: 0
- Align: Center
- Height: 433
```html
<TR ALIGN="CENTER"><TD HEIGHT="22">Doctor:<font size="2"></font><SELECT NAME="doctorId">
<option>/ Option>
<% if (rs.next()) {
%>
<% temp = rs.getString("ARCH_NAME"); %>
<% if (temp.equals("16:30")) temp = "16:00 PM";
%>
<% if (temp.equals("13:30")) temp = "1:30 PM";
%>
<% if (temp.equals("10:30")) temp = "10:30 AM";
%>
<% if (temp.equals("9:30")) temp = "9:30 AM";
%>
<% if (temp.equals("8:30")) temp = "8:30 AM";
%>
<% else if (temp.equals("8:00")) temp = "8:00 AM";
%>
<% } }
</SELECT>
</TD></TR>

<TR ALIGN="CENTER"><TD HEIGHT="25">Appointment Day:<font size="2"></font><INPUT TYPE="text" Name="dt" VALUE=""%
<% if (rs.getString("ARCH_NAME").equals("7:00 PM")); %>
<% } }
</TD></TR>

<TR ALIGN="CENTER"><TD HEIGHT="36"><INPUT TYPE="submit" NAME="find" VALUE="Find"
<% if (rs.getString("ARCH_NAME").equals("6:00 PM")); %>
<% } }
</TD><TR COLOR="#FF0000">

<% if (rs.next()) {
%>
<% temp = rs.getString("ARCH_NAME"); %>
<% if (temp.equals("16:30")) temp = "16:00 PM";
%>
<% if (temp.equals("13:30")) temp = "1:30 PM";
%>
<% if (temp.equals("10:30")) temp = "10:30 AM";
%>
<% if (temp.equals("9:30")) temp = "9:30 AM";
%>
<% if (temp.equals("8:30")) temp = "8:30 AM";
%>
<% else if (temp.equals("8:00")) temp = "8:00 AM";
%>
<% } }
</TD></TR>

<% if (rs.getString("ARCH_NAME").equals("7:00 PM")); %>
<% } }
</TD>
```
temp = "7:30 PM";
else if(rs.getString("APPOINTMENT_TIME").equals("20:00"))
temp = "8:00 PM";
else if(rs.getString("APPOINTMENT_TIME").equals("20:30"))
temp = "8:30 PM";
else if(rs.getString("APPOINTMENT_TIME").equals("21:00"))
temp = "9:00 PM";
else if(rs.getString("APPOINTMENT_TIME").equals("21:30"))
temp = "9:30 PM";
else if(rs.getString("APPOINTMENT_TIME").equals("22:00"))
temp = "10:00 PM";
else if(rs.getString("APPOINTMENT_TIME").equals("22:30"))
temp = "10:30 PM";
else if(rs.getString("APPOINTMENT_TIME").equals("23:00"))
temp = "11:00 PM";
else if(rs.getString("APPOINTMENT_TIME").equals("23:30"))
temp = "11:30 PM";

%>
<TD><font size=2><%=temp%>&nbsp;</font></TD>
<TD><font size=2><%=rs.getString("APPOINTMENT_DAY")%>&nbsp;</font></TD>
<TD><font size=2><%=rs.getString("DOCTOR")%>&nbsp;</font></TD>
<TD><font size=2><%=rs.getString("PATIENT")%>&nbsp;</font></TD>
</TR>
</TABLE>
</TD></TR>
</TABLE>
</form></td></tr>
<tr valign=bottom>
<td align='center''height="121">
<hr width="100%" color="#0085B0" size="1" noshade>
For any question and suggestion, please contact <a href=mailto:jarviscyt@hotmail.com>administrator</a> or (909)333-2727</font></td>
</tr>
</table>
</td>
</tr>
</table></script>
<p>&nbsp;</p>
</table>
</body>
</html>
<table>
<thead>
<tr>
<th>Status</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>1</td>
</tr>
<tr>
<td>Existing</td>
<td>2</td>
</tr>
<tr>
<td>Prescription</td>
<td>1</td>
</tr>
</tbody>
</table>

<a href="/nurse/changePassword.jsp">Change Password</a>
<table>
<thead>
<tr>
<th>Prescription ID</th>
<th>Nurse</th>
<th>Order Status</th>
<th>Dispense Date</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456</td>
<td>John</td>
<td>Prescribed</td>
<td>2023-01-01</td>
<td>None</td>
</tr>
<tr>
<td>789012</td>
<td>Jane</td>
<td>Issued</td>
<td>2023-02-02</td>
<td>Complete</td>
</tr>
<tr>
<td>345678</td>
<td>Mark</td>
<td>Pending</td>
<td>2023-03-03</td>
<td>In Process</td>
</tr>
<tr>
<td>901234</td>
<td>Lisa</td>
<td>-</td>
<td>2023-04-04</td>
<td>-</td>
</tr>
</tbody>
</table>

*Table shows summary of prescription information.*
<table>
<thead>
<tr>
<th>Patient ID:</th>
<th>&lt;input type=&quot;text&quot; name=&quot;patientid&quot; size=&quot;15&quot; value=&quot;&quot;/&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;input type=&quot;submit&quot; name=&quot;find&quot; value=&quot;find&quot;/&gt;</td>
<td>&lt;input type=&quot;submit&quot; name=&quot;cancel&quot; value=&quot;cancel&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;table width=&quot;100%&quot; bordercolor=&quot;#000000&quot; height=&quot;89&quot; align=&quot;center&quot;&gt;</td>
<td>&lt;/table&gt;</td>
</tr>
<tr>
<td>&lt;tr bgcolor=&quot;#3366FF&quot; align=&quot;center&quot;&gt;</td>
<td>&lt;td&gt;Prescription ID&lt;/td&gt; &lt;td&gt;Prescription Status&lt;/td&gt;</td>
</tr>
<tr>
<td>&lt;/tr&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;tr align=&quot;center&quot;&gt;</td>
<td>&lt;td colspan=&quot;2&quot;&gt;For any question and suggestion, please contact &lt;a href=&quot;mailto:jarviscyt@hotmail.com&quot;&gt;administrator&lt;/a&gt; or (909)333-2727&lt;/td&gt;</td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/td&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/tr&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/table&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/body&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/html&gt;</td>
<td></td>
</tr>
<tr>
<td>189</td>
<td></td>
</tr>
</tbody>
</table>
REFERENCES


