Sequence Similarity Search portal

Arokiya Louis Monica Joseph
SEQUENCE SIMILARITY SEARCH PORTAL

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
Arokiya Louis Monica Joseph

March 2007
SEQUENCE SIMILARITY SEARCH PORTAL

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

by
Arokiya Louis Monica Joseph
March 2007

Approved by:

Dr. Keith Schubert, Chair, Computer Science

Dr. Anthony Metcalf, Co-Chair, Biology

Dr. Arturo Concepcion, Computer Science

Dr. Ernesto Gomez, Computer Science

March 15, 2007
ABSTRACT

Sequence Similarity Search Portal serves as a single platform for sequence similarity searching. Sequence similarity searching against molecular sequence databases is commonly used by molecular biologist to clarify biochemical and physiological features of newly discovered genes and proteins. The Sequence Similarity Search portal currently allows the molecular biologists to search the two most popular molecular sequence databases NCBI and EMBL-EBI database and also to store their results for future references and share their results if needed.

The portal is also designed as a teaching aid and to be used as an extension to the wet lab. The molecular biology students can submit their search results and reports online. Students can also explore more about these two programs in particular by using Smart Sequence Similarity Search System (S4) that is interfaced with the portal. The S4 system is an expert system developed by Ms. Zhuo Chen as part of her Master's Degree program here at California State University San Bernardino that, guides the novice researchers in selecting the appropriate database and its search parameters for efficient sequence similarity searches.
The portal is developed using Servlet and JSP technology. The NCBI and EMBL-EBI databases are accessed through the NCBI QBLAST service and WSFasta web service offered by the NCBI server and EBI server respectively. The persistent data of the system are supported by MySQL server. The web portal can be accessed from anywhere through internet.
ACKNOWLEDGEMENTS

I take this opportunity to thank my advisor Dr. Schubert for guiding me in this project. Many a times, I had been awestruck by his ideas, suggestions and guidance. I have learnt to approach and analyze the problem as I call it: The Dr. Schubert’s way. I also thank Dr. Metcalf my co-advisor for helping me with the challenging domain research that lies behind the project.

My special thanks to Dr. Concepcion, who had guided me throughout my three years at Cal State. He inspired me to write academic papers, present my ideas and always demanded more from me. I also thank him for accepting to be my committee member and for helping me understand Zhuo’s S4 project to start with. I also thank Dr. Gomez, for his many valuable suggestions along the way.

I also take this opportunity to thank Dr. Mendoza, my graduate advisor for her support, guidance and of course for all those little tips that made my life so much better and Ms. Monica Latimer, for helping me every time and solving all those unsolvable administrative problems.

Last and most important, I thank my husband Antony for his unconditional moral & financial support and encouragement to pursue my interests, for listening to my
complaints and frustrations and yes, also for all those feedbacks and suggestions on the project, though I made a big fuss about it always.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>List of Program Listings</td>
<td>xiii</td>
</tr>
<tr>
<td>Chapter One: Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Introduction to Similarity Search</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Algorithms for Similarity Search</td>
<td>3</td>
</tr>
<tr>
<td>1.1.3 Web Tools for Similarity Search</td>
<td>5</td>
</tr>
<tr>
<td>1.1.4 Web Services</td>
<td>6</td>
</tr>
<tr>
<td>1.1.5 Smart Sequence Similarity Search (S^4) System</td>
<td>7</td>
</tr>
<tr>
<td>1.2 Purpose of the Project</td>
<td>9</td>
</tr>
<tr>
<td>1.3 Organization of this Documentation</td>
<td>10</td>
</tr>
<tr>
<td>Chapter Two: Software Requirement Specification</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1 Purpose</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2 Scope</td>
<td>11</td>
</tr>
<tr>
<td>2.1.3 Definitions, Acronyms, and Abbreviations</td>
<td>14</td>
</tr>
<tr>
<td>2.1.4 Overview</td>
<td>21</td>
</tr>
<tr>
<td>2.2 Overall Description</td>
<td>21</td>
</tr>
</tbody>
</table>
7.2 Future Direction ........................................ 103

APPENDIX A: LIST OF DATABASE TABLES ...................... 105

BIBLIOGRAPHY .......................................................... 111
LIST OF TABLES

Table 4.1. Axis Generated Java Classes .................... 65
Table 5.1. Report of Unit Test ............................... 74
Table 5.2. Report of Integration Test ...................... 77
Table 5.3. Report of System Test ......................... 79
LIST OF FIGURES

Figure 2.1. The Deployment Diagram ......................... 24
Figure 2.2. The Use Case Diagram .............................. 29
Figure 3.1. Web Service: Behind the Scene .................... 40
Figure 3.2. Message Flow Diagram-EBI Web Services .......... 44
Figure 4.1. The System Design ................................. 52
Figure 4.2. Request/Response Flow-JSP Page .................. 58
Figure 4.3. csusb.s4 Package Diagram ......................... 59
Figure 4.4. csusb.s4.controller Sub-Package Diagram ...... 61
Figure 4.5. csusb.s4.blast Sub-Package Diagram ............. 63
Figure 4.6. csusb.s4.fasta Sub-Package Diagram ............ 66
Figure 4.7. uk.ac.ebi.www Sub-Package Diagram ............. 67
Figure 6.1. The Login Page ..................................... 87
Figure 6.2. The Search Page .................................... 89
Figure 6.3. The Result Page .................................... 91
Figure 6.4. The Profile Page ................................... 93
Figure 6.5. The Group Page .................................... 95
Figure 6.6. The Group Page-Create/Delete Groups .......... 96
Figure 6.7. The Group Page-Add/Delete Users ................. 97
Figure 6.8. The Group Page-Share Results .................... 99
Figure 6.9. The S^4 Page ....................................... 100
LIST OF PROGRAM LISTINGS

Listing 4.1 Pseudocode for NCBI QBLAST Invocation .......... 70

Listing 4.2 Pseudocode for WSFasta
  Web Service Invocation ................................ 72
CHAPTER ONE

INTRODUCTION

1.1 Background

As we are entering the genomic era, high throughput technologies, such as large-scale sequencing, gene expression profiling, single-nucleotide polymorphism (SNP) discovery, and proteomics, become more and more developed and commonly used in laboratories. One of the net results of the advance of these technologies is the discovery of thousands of novel DNAs and proteins every month.

DNA is the basic carrier molecule of the genetic code of most organisms. An organism’s total DNA complement is called its genome. DNA is usually represented by its sequence of nucleotide bases consisting of Adenosine (A), Thymine (T), Cytosine (C), and Guanosine (G). The genetic information encoded in DNA can be transcribed into mRNA, which is then translated into proteins. Protein can also be represented by a sequence of characters, which are the abbreviations of twenty different types of amino acid residues, protein’s basic building block. Protein must form stable tertiary structures to be functional. Hence, they can be represented by a sequence of characters.
These DNA and protein sequences are usually deposited and stored in the public databases and can be retrieved by the researchers. It should be noted that it is not the sequence itself, but the function of the sequence that is the most interesting. The functions of many sequences, especially from model systems such as yeast, bacteria, mouse, and human, have been determined experimentally. However, the majority of sequences in sequence databases do not have known functions, and hence function prediction is in high demand. Consequently, sequence similarity searches against databases have become a mainstay of bioinformatics, partially because of the maturity of sequence alignment algorithms and the availability of high quality similarity search tools on the Internet.

1.1.1 Introduction to Similarity Search

The 'Smart Sequence Similarity Search System B[1]' explains, Similarity searching as the application of knowledge gained from previous experiments to the problem of discovering the biochemistry and physiology of a newly discovered gene or its protein product. In practice, the sequence of interest is compared to every sequence in a
sequence database, and the similar ones are identified. If a query sequence is similar to a database sequence of known function, structure, or biochemical activity, the query sequence is predicted to have the same function, structure, or biochemical activity. The strength of these predictions depends on the quality of the alignment between the sequences. Sequence alignment is to put two sequences together to find common patterns within sequences. The purpose of making alignments is to discover whether or not sequences are homologous or derived from a common ancestor gene, which implies similar function. If an alignment can be found that would rarely be observed between random sequences, the sequences are predicted to be related with a high degree of confidence.

1.1.2 Algorithms for Similarity Search

There are three major algorithms widely used in sequence similarity searching, Smith-Waterman [B2], FASTA (pronounced FAST-Aye)[B3], and BLAST (Basic Local Alignment Search Tools) [B4]. The different algorithms add different restrictions to the simple model of sequence evolution on which similarity searching is based.
The 'Smart Sequence Similarity Search System' [B1] describes in details the comparison of the three algorithm which is as follows:

Smith-Waterman is the most rigorous algorithm and does not place any heuristic restrictions on the evolutionary model [B2]. It is mathematically rigorous, and guaranteed to find the best scoring alignment between the pair of sequences being compared [B2]. FASTA stands for FAST-ALL, reflecting the fact that it can be used for a fast protein comparison or a fast nucleotide comparison. The high speed of this program is achieved by using the observed pattern of word hits to identify potential matches before attempting the more time consuming optimized search. Not every word hit is investigated, instead the program initially looks for segments containing several nearby hits [B5]. The BLAST programs are a set of sequence comparison algorithms used to search sequence databases for optimal local alignments to a query. The BLAST algorithm uses a word based heuristic similar to that of FASTA [B4]. The BLAST programs improved the overall speed of searches while retaining good sensitivity by breaking the query and database sequences into fragments ("words"), and initially seeking matches
between fragments. Word hits are then extended in either direction in an attempt to generate an alignment[B6].

Both BLAST and FASTA place additional restrictions on the alignments that they report in order to speed up their operation. The actual pattern of evolutionary changes between the query sequence and any homologues in the database can be incompatible with the heuristic restrictions imposed by either BLAST or FASTA. Alternatively the additional selectivity that results from these restrictions can sometimes be an advantage. Because of the mathematical rigor and lack of restrictions, the Smith-Waterman algorithm is more sensitive than either BLAST or FASTA [B7]. This additional sensitivity comes at the price of being a very much slower way to search a sequence database than are either BLAST or FASTA [B7].

1.1.3 Web Tools for Similarity Search

BLAST programs are available interactively through a public database server at the National Center for Biotechnology Information (NCBI) (http://www.ncbi.nlm.nih.gov). FASTA has recently become available interactively at the European Bioinformatics
Institute (EBI) (http://www.ebi.ac.uk/fasta33/). Results can also be sent to users by email from FASTA server. Because the Smith-Waterman algorithm is slow, there are very limited implementations of this algorithm for large scale similarity searches. However, there are efforts to better implement the Smith-Waterman algorithm [B8]. Right now, several servers provide similarity search services based on Smith-Waterman algorithm by emailing the results to the user, but interactive online tools is still not available. The largest Smith-Waterman based similarity search server with the most functionality is provided by DNA Data Bank of Japan (DDBJ). The program is called S&W SEARCH (http://www.ddbj.nig.ac.jp/E-mail/homology.html).

1.1.4 Web Services

According to W3C Group Working Notes[B15], a Web Service is defined as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed
using HTTP with an XML serialization in conjunction with other Web-related standards. Web services can perform encapsulated business functions, ranging from simple request-reply to full business process interactions. The project utilizes a simple request-reply business function offered by the NCBI and EMBL-EBI database servers. The web services architecture and implementation is discussed in detail in chapter 3.

1.1.5 Smart Sequence Similarity Search (S^4) System

An efficient sequence similarity search relies on the choice of tools and their associated subprograms and numerous parameter settings. This could be very challenging for similarity search users, especially those at the beginner level. To assist researchers in selecting optimal search programs and parameter settings for efficient sequence similarity searches, Ms. Zhuo Chen as part of her Master program in Computer Science have developed a Web-based expert system, Smart Sequence Similarity Search (S^4) system.

This system is implemented in Java and Jess scripts, and uses the Jess Expert System as its reasoning core. The
expert knowledge provided for a sequence similarity search is represented in the form of decision tree and stored in a XML file. The system also provides interfaces for expert users to improve this knowledge by extending the decision tree. With its capability to continuously improve sequence similarity searches through a decision tree, the Web-based expert system provides a solid advising tool for researchers interested in efficient sequence similarity searches.

The Sequence Similarity Search Portal provides interfaces to the $S^4$ system to allow the novice users to take advantage of its expert system and select appropriate molecular databases and its search parameters. The chosen molecular database and search parameters provided by the $S^4$ system are grasped by the portal and sent to the appropriate server. The portal then retrieves the results from the database and reveals to the user in a more comprehensible format. It also allows the user to save the search parameters and query sequences.
1.2 Purpose of the Project

Web Services technology brings into the bioinformatics community a new development concept in which users can access all data and applications hosted in the main research centers (EBI, DDBJ, KEGG, NCBI, etc) as if they were installed in their local machines, providing seamless integration between disparate services. The project takes one step in that direction to integrate the sequence similarity searching at EBI and NCBI by using web services. The project is also intended to allow the molecular biologists to save their searches and acts a log book for their sequence similarity searches. It also allows the molecular biologist to share their sequences and results with other researchers thereby enabling intellectual discussion on interested sequences.

The project also integrates the expert system developed by Ms. Zhuo Chen to provide expert knowledge to help users choose the best search algorithm with the most reasonable parameter settings, so that better search results can be obtained. The choice of algorithms and their specific parameters could become very challenging for similarity search users, especially for those who are at the beginner level. Thus the integration of the expert
system allows the users to use the expert system if they are unsure are not familiar with sequence similarity search.

1.3 Organization of this Documentation

The remaining sections of this documentation will be organized as follows: Chapter 2 describes the software requirements specification. Chapter 3 describes web services its architecture and implementation in the project. Chapter 4 provides a description of the system architecture and detailed design. Chapter 5 is the system test and Chapter 6 is the maintenance and users manual. Finally, Chapter 7 concludes the project and lists suggestions for future developments.
CHAPTER TWO
SOFTWARE REQUIREMENT SPECIFICATION

2.1 Introduction

2.1.1 Purpose

This Software Requirements Specification (SRS) documents the agreements concerning the purpose, characteristics and specific requirements of the proposed Sequence Similarity Search Portal. It will be developed by Arokiya Joseph as the requirements for Master Degree in Computer Science of the Department of Computer Science, California State University, San Bernardino, CA. It also interfaces the Smart Sequence Similarity Search System developed by Ms. Zhuo Chen as the requirements for Master degree in computer science of the Department of Computer Science, California State University, San Bernardino, CA.

2.1.2 Scope

This document specifies the software requirements for functions of the Sequence Similarity Search Portal to be used at CSUSB. It is one of the projects related to
bioinformatics being developed at CSUSB in order to provide a comprehensive, user friendly and personalized environment for sequence analysis to molecular biologists.

Sequence similarity searching is commonly used to help clarify the biochemical and physiological features of newly discovered genes or proteins. Sequence similarity searching is conducted on sequence databases, which is a large collection of DNA, protein, or other sequences stored on a computer. Some of the popular sequence databases include European Bioinformatics Institute databases, NCBI completely sequenced genomes and Stanford Saccharomyces Genome Database. There are three major algorithms widely used in sequence similarity searching on the sequence databases. They are Smith-Waterman, FASTA, and BLAST (Basic Local Alignment Search Tools) algorithm. BLAST algorithm is available interactively at the National Center for Biotechnology Information (NCBI) (http://www.ncbi.nlm.nih.gov). FASTA algorithm has recently become available interactively at the European Bioinformatics Institute (EBI) (http://www.ebi.ac.uk/fasta33/). The project aims to utilize the web services offered by the servers at NCBI and EBI and integrate the sequence similarity search on
above mentioned servers. As explained in [B16], Web Services technology enables scientists to access biological data and analyze applications as if they were installed on their laboratory computers. Similarly, it enables programmers to build complex applications without the need to install and maintain the databases and analysis tools and without having to take on the financial overheads that accompany these. Moreover, Web Services provide easier integration and interoperability between bioinformatics applications and the data they require.

The following features will be implemented in the project:

The user should be able to enter a query sequence for the sequence similarity search and choose either BLAST or FASTA algorithm and set the parameters suitably and retrieve the results from appropriate server. If the user is not familiar with sequence similarity search, he should be directed to the S4 expert system to decide the algorithm and its parameter values for sequence analysis.

The expert system decides the parameters based on a sequence of questions to be answered by the user (molecular biologist).
The user should be able to view the results of the sequence analysis and save the sequence and parameter settings if needed. The saved sequence and its corresponding algorithm and parameters are together referred as profiles.

The user can view or delete his saved profiles. The user can also get new results for the saved profile.

The user can create new groups and share his saved profiles with other researchers in the group. He should be able to add and delete users of the group.

Though there is many other sequence databases developed and being developed, this project does not aim to connect to the rest of the databases through web services, since NCBI and EMBL-EBI are the most sought after comprehensive databases. This project also does not aim to modify the expert system developed by Ms. Zhuo Chen.

2.1.3 Definitions, Acronyms, and Abbreviations

Amino Acid - Amino acids are structural molecular of proteins. Each molecule contains both an amino group and a carboxyl group.
**Amino Acid Sequence** - The sequence or order of linkage of amino acids in a given polypeptide chain or protein. This is ultimately determined by the genetic code.

**Bioinformatics** - A rapidly developing branch of biology that uses techniques and concepts from informatics, statistics, mathematics, chemistry, biochemistry, physics, and linguistics. It derives knowledge from computer analysis of biological data. These can consist of the information stored in the genetic code, but also experimental results from various sources, patient statistics, and scientific literature. Research in bioinformatics includes method development for storage, retrieval, and analysis of the data.

**BLAST** - Basic Local Alignment Search Tool. It is a set of similarity search programs designed to explore all of the available sequence databases regardless of whether the query is protein or DNA.

**CSS** - Cascade Style Sheet

**CSUSB** - California State University, San Bernardino

**DNA** - Deoxyribonucleic acid. A polymer of covalently linked deoxyribonucleotides serving as the primary genetic material of most biological organisms. DNA is
usually a double stranded helix of two polynucleotide chains linked by hydrogen bonds.

**DECISION TREE** - Decision trees possess (typically) a single root, a set of branches from that root, interior nodes that also have branches and terminal nodes which present the classification information. Each node within the tree represents a decision point that determines which subsequent branch is followed.

**EXPERT SYSTEM** - An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant expertise for their solutions.

**FASTA** - FASTA (pronounced FAST-Aye) stands for FAST-All, is to be used for a fast protein or nucleotide comparison.

**Gene** - Region of DNA that controls a discrete hereditary characteristic, usually corresponding to a single polypeptide, protein or RNA. It includes regions preceding and following the coding region as well as intervening sequences (intron) between individual coding segments (exon).

**Genome** - The total gene complement, comprising the genetic information of the entire organism.
HTML - Hyper Text Markup Language

HTTP - Hyper Text Transfer Protocol. The client/server protocol that defines how messages are formatted and transmitted on the World Wide Web

IEEE - Institute of Electrical and Electronics Engineers

Java - An object oriented language developed by Sun Microsystems. Java programs are capable of running on most popular computer platforms without the need for recompilation.

JSP - Java Server Page. An extension to the Java servlet technology from Sun that provides a simple programming vehicle for displaying dynamic content on a Web page.

Java Servlet - A Java application that runs in a Web server or application server and provides server-side processing, typically to access a database or perform e-commerce processing.

JavaScript - A scripting language that is widely supported in Web browsers and other Web tools. It adds interactive functions to HTML pages, which are otherwise static.
**JDBC** - Java DataBase Connectivity. A programming interface that lets Java applications access a database via the SQL language.

**JESS** - JESS is an expert system shell and scripting language written entirely in Sun Microsystem's Java language. It supports the development of rule-based expert systems which can be tightly coupled to code written in the powerful, portable Java language.

**JRE** - Java Runtime Environment

**JDK** - Java Development Kit

**Nucleotide** - It is the structural components of nucleic acids, including DNA and RNA. They consist of a pentose sugar, a phosphate and a nitrogenous base - Adenine, Cytosine, Guanine, Thymine (for DNA) or Uracil (for RNA).

**Nucleotide Sequence** - It is the sequence or order of linkage of nucleotides in a given polynucleotide chain or nucleic acid. They may include both coding and non-coding sequences in DNA and RNA.

**OS** - Operating System

**Protein** - It is a complex molecule consisting of a particular sequence of amino acids (peptides) that are joined to form a protein (polypeptides).
Proteomics - It is the study and analysis of protein structure and function. It is becoming quite an important science with the mapping of several genomes, including the human one, and the discovery of new proteins.

RNA - Ribonucleic acid. It is a polymer formed from covalently linked ribonucleotide monomers.

Rule-based Expert System - An expert system within which the knowledge base contains the domain knowledge needed to solve problems coded in the form of rules.

Sequence Alignment - The procedure of comparing two or more protein and nucleic acid sequences by looking for a series of individual characters or character patterns that is in the same order in the sequences.

SRS - Software Requirements Specification

S4 - Smart Sequence Similarity Search

Smith-Waterman algorithm - It uses dynamic programming to find optimal local alignments between sequences.

SOAP - Simple Object Access Protocol. It is a standard for exchanging XML-based messages over a computer network, normally using HTTP. SOAP forms the foundation layer of the web services stack, providing a basic messaging framework.
UDDI - Universal Description, Discovery and Integration.
It is a platform independent, open framework for
describing services, discovering businesses, services
using the internet.

URL - Universal Resource Locator

Web Service - A web service is a collection of protocols
and standards used for exchanging data between
applications or systems. Software applications
written in various programming languages and running
on various platforms can use web services to exchange
data over computer networks like the Internet in a
manner similar to inter-process communication on a
single computer.

WSDL - Web Services Description Language. It is an XML -
based language used to describe the services a
business offers and to provide a way for individuals
and other businesses to access those services
electronically.

XML - Extensible Markup Language. It is a W3C initiative
that allows information and services to be encoded
with meaningful structure and semantics that computer
and humans can understand. XML is great for
information exchange, and can easily be extended to include user-specified and industry-specified tags.

**XHTML - Extensible HTML.** It is an extension of HTML as an application of the XML language. XHTML enables HTML to be extended with proprietary tags.

### 2.1.4 Overview

Section 2.2 follows the guidelines of IEEE Std. 830-1998 IEEE Recommended Practice of Software Requirements Specifications [B12]. This section provides product perspective, a summary of product functions, a description of the characteristics of the expected users, and a list of assumptions and dependencies.

Section 2.3 presents the specific requirements for this system. They are organized by mode, following the SRS Section 3 template shown in IEEE Std 830-1998, Annex A, Paragraph A.3 [B18].

### 2.2 Overall Description

Currently, the widely used programs on sequence similarity search are BLAST, FASTA and Smith-Waterman (SW) algorithm based programs. The project integrates the BLAST
and FASTA similarity searching as one interface, for the convenience of molecular biologist.

2.2.1 Product Perspective

The Sequence Similarity Search Portal will be a Web-based system and is connected to the external web servers of NCBI and EMBL-EBI using web services. The user interfaces will be built to be viewed on web browsers. The system hardware interface requirement is that it must run on the existing web servers. The software interface is that it must support current versions of Netscape & Internet Explorer. The communications interface requires support for Hyper Text Transfer Protocol (HTTP).

The system will be operated 24 hours per day, 7 days per week. All actions are user-initiated. No separate backup and recovery or maintenance functions are required as that is handled by system administration on the hosting server machine.

2.2.1.1 System Interfaces. The system is three-tier architecture.

The first tier is the presentation tier that displays the user interface in a web browser via HTML.
The second tier is the web server that uses Java Servlet to handle requests from the client and response to the client after logic processing by the system application. The HTTP server is provided by Apache Tomcat.

Upon the user determining the algorithm and settings, system web server will communicate with one of the remote sequence similarity servers, NCBI or EMBL-EBI via SOAP messages. The similarity search results are retrieved using HTTP response. If the user wishes to save the profiles or create groups and share his results, the corresponding data is stored in the database supported by MySQL database server.
Figure 2.1. The Deployment Diagram
A user task will be conducted in the following steps:

- User initiates a HTML user interface through web browser;
- Request is transmitted to web server via HTTP protocol;
- Web server responds to the request and execute a Servlet or a Servlet compiled from a JSP page;
- If the user request requires retrieving results of the similarity searching algorithm the Servlet sends a SOAP message to the appropriate database server chosen by the expert system and retrieves the results from the server.
- If the user request requires a database action like saving a profile or sharing it with a group, the servlet connects to the database using JDBC and performs appropriate database action.
- If the user request corresponds to using the expert system, the servlet spawns and communicates with a separate thread running JESS expert system on the server; Jess expert system retrieves knowledge from XML, if necessary, and
determines the information to be used by Servlet by proper reasoning.

- A custom HTML document containing information suitable response to the user request is sent back to the user via HTTP protocol;
- HTML page is displayed by user’s web browser.

### 2.2.1.2 User Interfaces

User interfaces for the Sequence Similarity Search Portal will be designed in HTML page. The contents are generated dynamically by Servlet and presented as HTML output in response to the user’s requests. The following features will be incorporated to produce a more descriptive representation of the interface:

**Search Tab:** The general tab lets the user enter the name for the query, a query sequence and the option to select BLAST, FASTA search using user determined parameters or S4 defined algorithm and parameters. Depending on users selection the general tab would display suitable parameter list for sequence similarity search.

**Result Tab:** The result tab displays the result list retrieved from the database servers through web services
to the user and allows the user to save its profile for future references.

Profile Tab: The profiles tab displays the previous profiles that have been saved and allows the user to retrieve results from the profile or delete the profiles.

S4 Tab: The S4 tab allows the user to utilize the S4 Expert System to determine suitable tools and parameters.

Group Tab. The group tab lets the user to create new groups with the already registered users to share his profiles.

2.2.1.3 Hardware Interfaces. All hardware interfaces will be provided by the operating system. This system will not implement any hardware interface.

2.2.1.4 Software Interfaces. Software interfaces are provided in Java 1.4 APIs or higher, JSP 2.0 APIs and JESS 6.1 APIs or higher. In addition, user needs a Java Script compatible web browser, (Netscape 4.0 or higher, Internet Explorer 4.0 or higher). Web server needs to implement Java Servlet API and JSP API, XML API, and provide HTTP service.

2.2.1.5 Communications Interfaces. The communication interface uses HTTP for general information. Communication between threads uses Java piped input and output stream.
2.2.1.6 Memory Constraints. There is no specific memory requirement for client computer. Although there is no explicit memory requirement for Web server, enough memory is required to guarantee acceptable service speed because of the large size memory usage of Jess expert system. 128 Mega bytes or higher are recommended.

2.2.1.7 Operations. User shall access the system through the World Wide Web. The session can be expired if the user is inactive for a period of time, and subsequently accessing any pages might be redirected to the initial page.

2.2.2 Product Functions

Figure 2.2 shows Use Case diagram that graphically depicts users and principal functions of this system. The functions are further described below. and the actors in the diagram are described in section 2.3.
Figure 2.2. The Use Case Diagram
Sequence Similarity Using user determined Settings - Users will be asked to input a sequence initially, and choose appropriate algorithm and parameters and they can either access the NCBI or EMBL-EBI sequence database server using the default parameters.

Sequence Similarity Search Using S4 Expert System - Users will be asked to input a sequence and the S4 expert system decides the optimal algorithm and its parameters for their sequence similarity search.

View the Results of Similarity Search - Users can view the similarity search results.

Save the Profile - All users can save the profiles of his result if needed for future references.

Manage Profiles - The users can view or delete his profiles.

Create and Manage groups - The users can create and manage groups to share the saved profiles.

Sequence Similarity Search using saved profiles - The users can retrieve results of their previous searches from their saved profiles.

Extend the Decision Tree - The experts can extend the decision tree based on their knowledge of parameter values if they find it to be appropriate.
2.2.2.1 User Characteristics. Users of the system are molecular biologists are molecular biology students who would like to do sequence similarity search. All the users are assumed to know how to use web browser and speak English. They should also be able to follow the manual written in plain English.

2.2.2.2 Assumptions and Dependencies. These requirements assume there are no applicable hardware limitations. It is also assumed that system administration and maintenance issues will be dealt within the host system.

2.3 Specific Requirements

This section contains the software requirements to a level of sophistication that would enable designers to design the portal in conformance with the requirements of this Specification Requirement Specification document. This level of sophistication will also enable testers to generate tests for the system, to verify whether it meets the requirements.
2.3.1 External Interfaces

The external interfaces to the system are the remote database servers connected to the portal, the database servers of NCBI and EMBL-EBI. These external interfaces are accessed by the web services offered by them. Web Services is an integration technology. As explained by A. Labarga and et al.[B16], it can be stated that to ensure software from various sources work well together, web services is built on open standards such as Simple Object Access Protocol, a messaging protocol for transporting information; Web Services Description Language, a standard method of describing Web Services and their capabilities. For the transport layer itself, Web Services utilize most of the commonly available network protocols, especially Hypertext Transfer Protocol.

2.3.1.1 NCBI QBLAST. According to B[20] the URLAPI is a standardized application program interface (API) for accessing the NCBI QBLAST system. It uses direct HTTP-encoded requests to NCBI web server. Any HTTP browser or application capable to make HTTP requests can be utilized to use URLAPI.
2.3.1.2 EMBL-EBI Web Service. The EBI provides Web Services for sequence similarity tools (Fasta, WUBlast, NCBIBlast, MPsrch and ScanPS); protein analysis (InterProScan); multiple alignment (ClustalW and Tcoffee); and the European Molecular Biology Open Source Software Suite (EMBOSS). These are Web Services servers that provide the same or even more advanced functionality than the traditional browser-based services found at http://www.ebi.ac.uk/services. These services are implemented on a Perl-based, SOAP server and are tightly integrated with EBI hardware and middleware infrastructure. All web services have been developed using a common framework, and offer a similar interface to the user.

2.3.2 Performance Requirements

The portal will support approximately all researchers in CSUSB to use simultaneously. The response time to view any page should be less than ten seconds. The response time for seeing any result after submitting an input usually should not exceed twice the length than viewing a page.
2.3.3 Software System Attributes

2.3.3.1 Reliability. All contents and logs shall be generated dynamically and automatically so no human interference is needed. The sever shall be up twenty-four hours a day and seven days a week, with exception that periodical system maintenance needs to be conducted depending on the reliability of the web sever. The system should handle network packet loss smoothly. The system should not save inconsistent data or incomplete data into the system.

2.3.3.2 Security. The web-server used to execute the portal is open to public. No security issue is considered in this thread.

2.3.3.3 Maintainability. The system consists of java classes, JESS expert system and JSP pages. They will be put under different directories with hierarchy. Other files including java source code and documents will be put in separate directories as well. Java classes are organized using packages. This structure will aid in maintaining all modules organized and therefore maximizing maintenance facility.
2.3.3.4 Portability. The Server-Implementation of the portal shall be 100% portable since it will be written in Java, a proven portable language. The only determinant of how easily it is ported from one architecture to another is having the latest version of the Java Virtual Machine and MySQL server installed on the web-server machine. The Client portion of the portal will also be 100% portable since the system will be in presented using dynamic HTML pages and style sheet, which is supported by most up-to-date web browsers.
3.1 Web Services: Introduction

A Web Service is programmable application logic accessible using standard Internet protocols. Web Services combine the best aspects of component-based development and the Web. Like components, Web Services represent black-box functionality that can be reused without worrying about how the service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-specific protocols, such as DCOM, RMI, or IIOP. Instead, Web Services are accessed via ubiquitous Web protocols (ex: HTTP) and data formats (ex: XML). The advantages of using web services as described by Vivek Devarajan[B17] can be listed as follows:

First, web services are reusable software components. Web services continue the long ascension of object-oriented design in software development. Rather than requiring programmers to write one start-to-finish set of instructions after another, the component-based model allows developers to reuse the building blocks of code created by others to assemble and extend them in new ways.
Second, these software components are loosely coupled. Traditional application design depends upon a tight interconnection of all subsidiary elements. The complexity of these connections requires that developers thoroughly understand and have control over both ends of the connection; moreover, once established, it is exceedingly difficult to extract one element and replace it with another. Loosely coupled systems, on the other hand, require a much simpler level of coordination and allow for more flexible reconfiguration.

Third, web services semantically encapsulate discrete functionality. A web service is a self-contained "applet" that performs a single task. The component describes its own inputs and outputs in a way that other software can determine what it does, how to invoke its functionality, and what result to expect in return.

Fourth, web services can be accessed programmatically. Unlike web sites and desktop applications, web services are not designed for direct human interaction, and they do not have a graphical user interface. Rather, web services operate at the code level; they are called by and exchange data with other software.
Web services certainly will be incorporated into software designed for human interaction, however.

Finally, web services are distributed over the Internet. Web services make use of existing, ubiquitous transport protocols like HTTP. By piggybacking on the same, well-understood transport as web content, web services leverage existing infrastructure and can comply with current corporate firewall policies.

3.2 Web Services: Behind the Scene

As described in W3C working group notes B[15], the basic requirements for a network node to play the role of web service requestor or provider are the ability to build, parse a SOAP message, or both, and the ability to communicate over a network (receive, send messages, or both). SOAP is a simple and lightweight XML-based mechanism for exchanging structured data between network applications. SOAP consists of three parts: an envelope that defines a framework for describing what is in a message, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls (RPCs) and responses. SOAP can be used in combination with or re-enveloped by a
variety of network protocols such as HTTP, SMTP, FTP, RMI over IIOP or MQ.

Typically, a SOAP server running in a Web application server performs these functions. Alternatively, a programming language-specific runtime library can be used that encapsulates these functions within an API. Application integration with SOAP can be achieved by using four basic steps (See Fig 3.1):
In Figure 3.1, at (1) a service requestor's application creates a SOAP message. This SOAP message is the request that invokes the Web service operation provided by the service provider. The XML document in the body of the message can be a SOAP RPC request or a document-centric message as indicated in the service description. The service requestor presents this message together with the network address of the service provider to the SOAP infrastructure (for example, a SOAP client runtime).
SOAP client runtime interacts with an underlying network protocol (for example HTTP) to send the SOAP message out over the network.

At (2) the network infrastructure delivers the message to the service provider's SOAP runtime (for example a SOAP server). The SOAP server routes the request message to the service provider's Web service. The SOAP runtime is responsible for converting the XML message into programming language-specific objects if required by the application. This conversion is governed by the encoding schemes found within the message.

The Web service is responsible for processing the request message and formulating a response. The response is also a SOAP message. At (3) the response SOAP message is presented to the SOAP runtime with the service requestor as the destination. In the case of synchronous request/response over HTTP, the underlying request/response nature of the networking protocol is used to implement the request/response nature of the messaging. The SOAP runtime sends the SOAP message response to the service requestor over the network.

At (4) the response message is received by the networking infrastructure on the service requestor's node.
The message is routed through the SOAP infrastructure; potentially converting the XML message into objects in a target programming language. The response message is then presented to the application.

While it is important to understand this foundation, most Web service developers will not have to deal with this infrastructure directly. Most Web Services will use optimized programming language-specific bindings generated from WSDL. WSDL contains sufficient information to describe to the service requestor how to invoke and interact with the Web service. To deploy web services and host the web services it is necessary to have a central organization for registering, finding and using Web Services UDDI provides a mechanism for holding descriptions of Web Services.

3.3 EBI Web Service

The EBI web services aims to provide programmatic access to the various databases and retrieval and analysis services EBI provides. The EBI currently provides Web Services for sequence similarity tools (Fasta, WUBlast, NCBIBlast, MPsrch and ScanPS); protein analysis (InterProScan); multiple alignment (ClustalW and Tcoffee);
and the European Molecular Biology Open Source Software Suite (EMBOSS). These are Web Services servers that provide the same or even more advanced functionality than the traditional browser-based services found at http://www.ebi.ac.uk/services. These services are implemented on a Perl-based, SOAP::Lite (http://www.soaplite.com) server and are tightly integrated with EBI hardware and middleware infrastructure.

All EBI web services have been developed using a common framework, and offer a similar interface to the user. They provide three basic methods: runApp, checkStatus, and getResults. The runApp method (where App is the name of the application, i.e. runFasta, runClustalW etc) is used to submit a job to the EBI job dispatcher. This method accepts two inputs: an InputParams structure with the options to be passed to the application, and a string array with the sequences. The job can be submitted in two modes: synchronous and asynchronous. In both cases, the server returns a job identifier which can be used to retrieve the results (see Figure 3.2).
Figure 3.2. Message Flow Diagram—EBI Web Services (Source: http://www.ebi.ac.uk/Tools/webservices)

Synchronous mode: This mode is equivalent to a user running a command on a console or terminal and waiting for it to complete. This requires the client to be constantly connected to the server. This mode is suitable for database searches that can be executed in up to 5 minutes.

Asynchronous mode: In this mode, the user submits a job and receives a job identifier in return. This is the same as running a UNIX command in the background and obtaining a job id. The user can query the status (using the checkStatus method) of an asynchronous mode job and
receive the following four states in response: JOB RUNNING (i.e. the job is currently being processed), JOB PENDING (i.e. the job is in a queue waiting processing), JOB NOT FOUND (i.e. the job id is no longer available; job results are deleted after 24 h) and JOB FAILED (i.e. the job failed or no results where found). DONE is returned when the job has finished, and the results can then be retrieved. Once the job has been completed, the user can retrieve a list of the produced results using the getResults method. This generates a list of results identifiers and the type of file (text, image, etc). The method poll is then used to retrieve the desired result.

3.4 NCBI QBLAST

The NCBI Web service is a web program that enables developers to access Entrez Utilities via the Simple Object Access Protocol (SOAP). Programmers may write software applications that access the E-Utilities using any SOAP development tool. However the project requirements did not extend beyond using URLAPI of the NCBI QBLAST and hence did not use the web services provided by NCBI. The NCBI QBLAST tutorial [B20] explains the process of NCBI QBLAST access as follows
The URLAPI is a standardized application program interface for accessing the NCBI QBLAST system. It uses direct HTTP-encoded requests to NCBI web server. These encoded requests should be directed to the NCBI cgi-bin program: "http://www.ncbi.nlm.nih.gov/blast/Blast.cgi". Any HTTP browser or application capable to make HTTP requests can be utilized to use URLAPI.

Searching the NCBI QBLAST consists of two major steps. The first step is called "Put", and it puts the query sequence with the appropriate search parameters into the QBLAST system. The second step is called "Get" and it formats the results with specified format parameters. The following is an example how can this be done by using URLAPI:

Let's say we want to perform a nucleotide search of query with gi=555 against the 'nr' database. In addition we need to enable low-complexity filtering, set the expect value = 10, request HTML format of the output, use NCBI GI numbers in the output page, and show only the first 10 hits. In the URLAPI the first step will look like:

&DATABASE=nr&HITLIST_SIZE=10&FILTER=L
&EXPECT=10&FORMAT_TYPE=HTML&PROGRAM=blastn&CLIENT=web
&SERVICE=plain&NCBI_GI=on&PAGE=Nucleotides
&CMD=Put"
In the url-encoded format the '?' means the start of a list of parameters, which is followed by a list of name-pairs separated with '&'. Note, that we set 'CMD=Put' which means that we want to put this new search into QBLAST. It is also possible to specify a query sequence in FASTA format like 'QUERY=acgtacgt'.

The output of the 'Put' command will be a valid HTML page, the contents of which may be ignored except the following important section:

```html
<!--QBlastInfoBegin
RID = 954517067-8610-1647
RTOE = 207
QBlastInfoEnd
-->

This portion of the output is special and contains the Request Identifier (RID) and the estimated Request Time of Execution in seconds (RTOE) for the search. The RID is different for every search, and is a mandatory parameter for the next step which is formatting the BLAST results.

The simplest way to get results for a given RID using the default format parameter is to use the following URL:

RID=954517013-7639-11119&CMD=Get"

47
If the BLAST search is not yet complete, this will produce output in the following format with Status equal to "WAITING":

    <!--QBlastInfoBegin
    Status=WAITING
    QBlastInfoEnd
    -->

If the results are completed, the output will show the formatted results with status information like the following, with Status=READY:

    <!--QBlastInfoBegin
    Status=READY
    QBlastInfoEnd --><p>
    ... <formatted output here>

If search result has the Status of WAITING, it is advisable to wait for few seconds before trying to get results again.
CHAPTER FOUR
DESIGN OF THE PORTAL

4.1 System Design

The Sequence Similarity Search Portal is a web based system and utilizes the web services offered by NCBI and EBI servers.

4.1.1 System Overview

The Sequence Similarity Search Portal is intended to integrate the sequence similarity search tool available at NCBI and EMBL-EBI servers. It is implemented using WSFasta web service offered by the EBI server and NCBI QBLAST service offered by NCBI server. The portal is an online application and can be accessed from any part of the world through the internet. The system design follows the MVC design pattern for web applications.

The goal of the MVC design pattern as described by Sarah Wheeler [B18] is to separate the application object (model) from the way it is represented to the user (view) from the way in which the user controls it (controller). The model object knows about all the data that need to be
displayed. It also knows about all the operations that can be applied to transform that object. However, it knows nothing whatever about the GUI, the manner in which the data are to be displayed, nor the GUI actions that are used to manipulate the data. The data are accessed and manipulated through methods that are independent of the GUI. The view object refers to the model. It uses the query methods of the model to obtain data from the model and then displays the information. The controller object knows about the physical means by which users manipulate data within the model. In a GUI for example, the controller object would receive mouse clicks or keyboard input which it would translate into the manipulator method which the model understands.

The view component of the portal is presented by JSP pages that are generated dynamically based on the Http request/response object. The controller component comprises of SearchController, ResultController, ProfileController, ExpertController and GroupController to manipulate the data presented to the model. The model component by itself consists of three separate models since they are independent in function. They are NCBI QBLAST invoke model, Fasta Web Service invoke model and
the local database (See Fig 3.1). The detailed functionalities of each component are explained in the following sections.
4.1.2 The Controller

The controller component probes the HTTP request/response object sent by the view component and invokes appropriate methods on the model. The SearchController and ExpertController invokes methods on the NCBI QBLAST invoke model and Fasta web service invoke.
model. The ProfileController, GroupController and ResultController are responsible for database operations on the local database.

4.1.3 The NCBI QBLAST Invoke Model

The NCBI QBLAST invoke component is responsible for accepting the parameters from the corresponding controller and builds a suitable scenario (like constructing the URL, establishing connection to NCBI server and so on) to invoke the NCBI QBLAST service offered by NCBI server. It also receives the XML result from the NCBI QBLAST system, formats them appropriately and sends it to the corresponding view component.

4.1.4 The FASTA Web Service Invoke Model

The fasta web service invoke component receives the parameter sent by the controller and constructs parameter objects to be sent to the EBI server. It invokes the WSFasta web service using suitable functions described in the WSDL of the WSFasta web service. Finally, it receives the XML result, processes it to acceptable format and forwards it to the corresponding view component.
4.1.5 The Database

The local database is used to store persistent data. The portal allows the user to save the profiles of their search and hence the user can maintain a history of their searches. It also allows the user to create groups to share his searches with other interested researchers. To maintain this information the following database tables are created.

USER: Stores information about username and password

PROFILE: stores information about profiles created by the user.

BLAST: stores information on blast parameters chosen for saved search

FASTA: stores information on fasta parameters chosen for saved search

GROUP: stores information on groups created by the user

GROUPMEMBERS: stores information on the group members for every group

SHAREPROFILE: stores information on profiles that are shared by corresponding group members
The attributes and constraints of every table are described in detail in Appendix A.

4.2 System Architecture

4.2.1 Overview

The web portal is designed around Java servlets, JSP and a home-grown persistence layer. The servlets and JSP pages help separate presentation from content and bring the Write Once, Run Anywhere paradigm to web applications. The embedded Java code (scriptlets) in sections of HTML documents can result in complex applications that are not efficient, and difficult to reuse, enhance, and maintain. Hence Java Servlet technology and JavaServer Pages are used to build quality, reusable, and easily maintainable web applications.

4.2.1.1 Overview of Servlet. As described by Qusay H. Mahmoud [B19], similar to Common Gateway Interface (CGI) scripts, servlets support a request and response programming model. When a client sends a request to the server, the server sends the request to the servlet. The servlet then constructs a response that the server sends
back to the client. Unlike CGI scripts, however, servlets run within the same process as the HTTP server. When a client request is made, the service method is called and passed a request and response object. The servlet first determines whether the request is a GET or POST operation. It then calls one of the following methods: doGet or doPost. The doGet method is called if the request is GET, and doPost is called if the request is POST. Both doGet and doPost take request (HttpServletRequest) and response (HttpServletResponse).

In the simplest terms, servlets are Java classes that can generate dynamic HTML content using print statements. What is important to note about servlets, however, is that they run in a container, and the APIs provide session and object life-cycle management. Consequently, when we use servlets, we gain all the benefits from the Java platform, which include the sandbox (security), database access API via JDBC, and cross-platform portability of servlets.

4.2.1.2 JavaServer Pages (JSP). The JSP technology, as described by Qusay H Mahmoud [B19], which abstracts servlets to a higher level--is an open, freely available specification developed by Sun Microsystems as an alternative to Microsoft's Active Server Pages (ASP)
technology, and a key component of the Java 2 Enterprise Edition (J2EE) specification. Many of the commercially available application servers (such as BEA WebLogic, IBM WebSphere, Live JRun, Orion, and so on) support JSP technology.

A JSP page is basically a web page with traditional HTML and bits of Java code. The file extension of a JSP page is .jsp rather than .html or .htm, which tells the server that this page requires special handling that will be accomplished by a server extension or a plug-in.

When a JSP page is called, it will be compiled (by the JSP engine) into a Java servlet. At this point the servlet is handled by the servlet engine, just like any other servlet. The servlet engine then loads the servlet class (using a class loader) and executes it to create dynamic HTML to be sent to the browser, as shown in Figure 3.2. The servlet creates any necessary object, and writes any object as a string to an output stream to the browser.

The next time the page is requested, the JSP engine executes the already-loaded servlet unless the JSP page has changed, in which case it is automatically recompiled into a servlet and executed.
4.2.2 Package csusb.s4

The system architecture of the Sequence Similarity Search Portal consists of four sub-packages under the package csusb.s4. The four sub-packages are csusb.s4.controller, responsible for manipulating the data presented to the WSFasta web service or NCBI QBLAST model component, csusb.s4.blast which is the business logic for accessing the NCBI QBLAST system and the csusb.s4.fasta, the business logic for accessing the WSFasta web service offered by EBI server which is explained in detail in section 3.2.3.3.
4.2.3 Sub-Package: csusb.s4.controller

The csusb.s4.controller consists of five independent controllers S4SearchController, S4ProfileController, S4ResultController, S4ExpertController and S4GroupController.

S4SearchController: responsible for accepting the search parameters from search.jsp view. It instantiates the objects for the model component and sets the values of
its attributes and routes it to the NCBI QBLAST invoke model or Fasta web service invoke model.

S4ResultsController: responsible for communicating with the Results.jsp view and saving the profiles if requested by the user to persistent storage.

S4ExpertController: responsible for receiving the decisions made by the S4Expert system, processing it suitably and sending it to the NCBI QBLAST invoke model or Fasta web service invoke model.

S4ProfileController: responsible for deleting the profiles chosen by the user and also responsible for retrieving the parameter values from the saved profiles in S4 database and forwarding it to the appropriate model component.

S4GroupController: responsible for persistent data operations to create/manage groups, its members and to share profiles.
Figure 4.4. csusb.s4.controller Sub-Package Diagram

4.2.4 Sub-Package: csusb.s4.blast

This sub-package consists of three classes: BlastPutParams, BlastGetParams and BlastWebService and constitutes the NCBI QBLAST invoke model of the system. The functionalities of each class is as follows
BlastPutParams: It consists of all parameters specified by the NCBI QBLAST system for the put command and get and set functions for all the parameters.

BlastGetParams: It consists of list of parameters specified by the NCBI QBLAST system for the get command and get and set methods for all the parameters.

BlastWebService: It is responsible for connecting to the NCBI server to access the NCBI QBLAST system. It retrieves the result of the query and sends it to the appropriate view component.
Figure 4.5. csusb.s4.blast Sub-Package Diagram
4.2.5 Sub-Package: csusb.s4.fasta

This sub-package that constitute the Fasta Web Service invoke model consists of two classes Fastaparams and FastaWebService. Fastaparams class consists of all the required search parameters for fasta algorithm and get and set methods to initialize and retrieve their values.

FastaWebService class consists of methods that invoke the fasta web service offered by the EBI server.

The csusb.s4 also consists of a sub-package uk.ac.ebi.www.WSFasta which is generated by the Apache Axis 1.1 toolkit. Axis is essentially a SOAP engine -- a framework for constructing SOAP processors such as clients, servers, gateways, etc. Apart from this, it also provides an extensive support for the Web Service Description Language (WSDL) and WSDL2Java emitter tooling that generates Java classes from WSDL.

The Axis WSDL2Java tool is found in "org.apache.axis.wsdl.WSDL2Java" of the axis installation directory. The basic invocation form looks like this:

% java org.apache.axis.wsdl.WSDL2Java (WSDL-file-URL)

The WSDL-File-URL for WSFasta web service is URL: "http://www.ebi.ac.uk/Tools/webservices/wsd1/WSFasta.w
The above invocation will generate only those bindings necessary for the client. Axis follows the JAX-RPC specification when generating Java client bindings from WSDL. The generated files will reside in the directory specified by the WSDL and namespaces map to Java packages. The generated classes are listed in table 3.1.

Table 4.1. Axis Generated Java Classes

<table>
<thead>
<tr>
<th>WSDL clause</th>
<th>Java class(es) generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each entry in the type section</td>
<td>A java class</td>
</tr>
<tr>
<td></td>
<td>A holder if this type is used as an in/out parameter</td>
</tr>
<tr>
<td>For each portType</td>
<td>A java interface</td>
</tr>
<tr>
<td>For each binding</td>
<td>A stub class</td>
</tr>
<tr>
<td>For each service</td>
<td>A service interface</td>
</tr>
<tr>
<td></td>
<td>A service implementation (the locator)</td>
</tr>
</tbody>
</table>
Figure 4.6. csusb.s4.fasta Sub-Package Diagram
Figure 4.7. uk.ac.ebi.www Sub-Package Diagram
4.3 Detailed Design

This section contains the detailed design for the NCBI QBLAST system invocation and WSFasta webservice invocation.

4.3.1 NCBI QBLAST System Invocation

The URLAPI as described in [B20] is a standardized application program interface (API) for accessing the NCBI QBLAST system. It also uses direct HTTP-encoded requests to NCBI web server. These encoded requests should be directed to the NCBI cgi-bin program:

A few of the highlighted features and advantages of using HTTP requests as described by [B20] for the service are:

• no program should be downloaded from NCBI ftp site to use NCBI BLAST service,
• easy to keep backward compatibility of API,
• no need to transfer sequences from NCBI to local machines,

The NCBI QBLAST system is invoked using two major steps: The first step is "Put", which pushes a search
request with appropriate parameters into the system. A successful "Put" step receives a unique Request ID (RID) in return. The second step is "Get", which retrieves the result for a completed search with its assigned RID and other formatting related parameters. The pseudocode described in Listing 3.1 discusses how these two steps are manipulated to receive result from the NCBI QBLAST system.
Function BlastWebService(String BlastputURL, String BlastgetURL)
  //BlastputURL: is the parameter list for put command
  //BlastgetURL: is the parameter list for get command

  //Sends the put command to NCBI Server
  putURL=Convert BlastputURL to URL object
  Open URLConnection
  Get Blast Result as InputStream
  Close Connection

  //Retrieves the RID value from the result
  Run through the blastresult by every line
  if (line.contains("<!--QBlastInfoBegin"))
      set RID=NextLine

  //Checks the status of RID
  getURL=Convert BlastgetURL to URL object
  Open URLConnection
  Get Result as InputStream
  Run through the blastresult by every line
  if (line.contains("status=WAITING")
      check status again
  else
      continue
  //if Status=Ready retrieve result
  Open URLConnection on getURL
  Get Result as InputStream
  Convert it to String
  Return

Listing 4.1 Pseudocode for NCBI QBLAST Invocation
4.3.2 Fasta Web Service Invocation

To perform analyses remotely without having to install software locally and requiring a local computer EBI server offers programmatic access to a selection of its services via web service. WSFasta is the web service offered for Sequence similarity searching on the EMBL-EBI molecular sequence database. To invoke this web service the three methods are specified in the WSFasta WSDL (http://www.ebi.ac.uk/ws/WSFasta.wsdl). They are getResult method, CheckStatus method and polljob method explained in chapter 3. The psuedocode below describes how the results are retrieved from the EBI server using the above methods.
function FastaWebService(FastaParams)
  // fastaparams is the list of all parameters for fasta search specified in the WSDL
  Instantiate the InputParams object
  // InputParams is an auto-generated class by the WSDL2JAVA tool
  Initialize the values of InputParams from Fastaparams object
  Instantiate a data object
  // data is an auto-generated class by the WSDL2JAVA tool
  Data.setContent(fastaparams.getInputSequence())
  Create a Service object service
  Set fasta = service.getWsFasta() Method
  Set jobid = fasta.runFasta()
  // Check Status of the Query
  Spawn a new thread
    If checkstatus(jobid) == "Running"
      Wait on thread for 30 sec
      Checkstatus(jobid)
    Else
      Close Thread
      Call Fasta.poll(jobid, xml) and
      Retrieve result
    End
  End
  Return result

Listing 4.2 Pseudocode for WSFasta Web Service Invocation
5.1 Introduction

Software quality assurance is a planned and systematic approach to the evaluation of the quality of and adherence to software product standards, processes, and procedures. Here, we only focus on using software testing to evaluate the system.

5.2 Unit Test

Unit test is the initial step in software testing phase. It is to test the system's basic components regarding functionality, validation and compatibility. In Sequence Similarity Search Portal, these basic components are independent compile unit such as a java class file or a combination of java class file and JSP pages. The report of the unit test for the portal is shown in Table 5.1. To facilitate illustration, tests are classified into categories. The test's goals as well as the files on which the test is conducted are listed. All the functions listed pass the test successfully.
<table>
<thead>
<tr>
<th>Category</th>
<th>Test objective</th>
<th>Source Code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAST Remote Server Invocation</td>
<td>To return a NCBI XML document</td>
<td>BlastWebService.java Blastputparams.java BlastGetParams.java</td>
<td>Pass</td>
</tr>
<tr>
<td>FASTA Remote Server Invocation</td>
<td>To return a Fasta XML document</td>
<td>FastaWebService.java Fastaparams.java</td>
<td>Pass</td>
</tr>
<tr>
<td>Login Function</td>
<td>The user should be able to login for a valid username password</td>
<td>Login.jsp S4LoginController.java</td>
<td>Pass</td>
</tr>
<tr>
<td>Login Function</td>
<td>The user should be given a message for invalid username or password while logging</td>
<td>Login.jsp S4LoginController.java</td>
<td>Pass</td>
</tr>
<tr>
<td>Login Function</td>
<td>A new user should be able to login with a new username and password</td>
<td>Login.jsp S4LoginController.java</td>
<td>Pass</td>
</tr>
<tr>
<td>Search Function</td>
<td>The user should be displayed BLAST parameters on selecting BLAST SEARCH</td>
<td>Search.jsp</td>
<td>Pass</td>
</tr>
<tr>
<td>Search</td>
<td>The user</td>
<td>Search.jsp</td>
<td>Pass</td>
</tr>
<tr>
<td>Function</td>
<td>should be displayed Fasta parameters on selecting BLAST SEARCH</td>
<td>Result.jsp</td>
<td>Pass</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>Result Function</td>
<td>The user should be able to view BLAST search parameters</td>
<td>Result.jsp</td>
<td></td>
</tr>
<tr>
<td>Result Function</td>
<td>The user should be able to view Fasta search parameters</td>
<td>Result.jsp</td>
<td></td>
</tr>
<tr>
<td>S4 Expert System Compatibility</td>
<td>The user should be able to view Fasta search parameters</td>
<td>S4.jar</td>
<td></td>
</tr>
<tr>
<td>S4 Expert System Compatibility</td>
<td>The user should be able excess expert system</td>
<td>S4.jar</td>
<td></td>
</tr>
<tr>
<td>Profile Function</td>
<td>The user should be able to view list of profiles</td>
<td>Profile.jsp</td>
<td></td>
</tr>
<tr>
<td>Profile Function</td>
<td>The user should be able to view details of every profile</td>
<td>Profile.jsp</td>
<td></td>
</tr>
<tr>
<td>Group Function</td>
<td>The user should able to create new group</td>
<td>Group.jsp</td>
<td></td>
</tr>
<tr>
<td>Group Function</td>
<td>The user should able to create new group</td>
<td>Group.jsp</td>
<td></td>
</tr>
</tbody>
</table>

75
new users for a group

<table>
<thead>
<tr>
<th>Group Function</th>
<th>The user should be able to view his group members.</th>
<th>Group.jsp S4GroupController.java</th>
<th>Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Function</td>
<td>The user should be able to share his profiles with other group members.</td>
<td>Group.jsp S4GroupController.java</td>
<td>Pass</td>
</tr>
</tbody>
</table>

5.3 Integration Test

Most of the functionalities in the system are provided by integrating some of the individual components. The related-function components are integrated at different levels and tested for compatibility. The test report is shown in Table 5.2. The integrated components with the tested functions are listed. All the integration tests were passed as desired.
Table 5.2. Report of Integration Test

<table>
<thead>
<tr>
<th>Test components</th>
<th>Test objective</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login + Search</td>
<td>The user should be able to login successfully and search using BLAST Parameters and output the result for the user.</td>
<td>Pass</td>
</tr>
<tr>
<td>+ Result</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login + Search</td>
<td>The user should be able to login successfully and search using Fasta Parameters and output the result for the user.</td>
<td>Pass</td>
</tr>
<tr>
<td>+ Result</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login + Search</td>
<td>The user should be able to login successfully and use S4 Expert System and send the values of its result to Search page</td>
<td>Pass</td>
</tr>
<tr>
<td>+ S4 Expert</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login + Profile</td>
<td>The user should be able to login successfully and view his profiles and get result for saved profiles</td>
<td>Pass</td>
</tr>
<tr>
<td>+ Result</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login + Group</td>
<td>The user should be able to login successfully and view his group page</td>
<td>Pass</td>
</tr>
<tr>
<td>Login + Group</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.4 System Test

The whole system is created by integrating all the components to fulfill all the functionalities of the portal. The system should maintain its integrity over its complete life cycle. If the system raises fatal exception, it should exit modestly.

A sample life cycle scenario is tested as below:
1. Login to the portal
2. Search NCBI molecular sequence database using BLAST algorithm
3. View the result
4. Save the profile
5. View the profile
6. Share the profile to one of his group.

All the functions are tested and passed successfully, as shown in Table 5.3.
Table 5.3. Report of System Test

<table>
<thead>
<tr>
<th>Function</th>
<th>Test objective</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login to the portal</td>
<td>Successful login for valid username and password and the user should be directed to Search.jsp</td>
<td>Pass</td>
</tr>
<tr>
<td>Search NCBI molecular sequence</td>
<td>The NCBI BLAST Parameters were displayed on selection at search page And the user should be directed to Result.jsp</td>
<td>Pass</td>
</tr>
<tr>
<td>database using BLAST algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>View the result</td>
<td>The user parameters should be viewed on selecting show parameters</td>
<td>Pass</td>
</tr>
<tr>
<td>Save the profile</td>
<td>The user should be able to save the profile and directed to Result.jsp</td>
<td>Pass</td>
</tr>
<tr>
<td>View the profile</td>
<td>The user should be able to view the profiles and on selecting view profile the details profile should be displayed</td>
<td>Pass</td>
</tr>
<tr>
<td>Share the profile with his group</td>
<td>The user should be displayed the list of his groups and on selection displayed the results he has shared already. He should be able to share his new profile with another group.</td>
<td>Pass</td>
</tr>
</tbody>
</table>
CHAPTER SIX
SYSTEM MAINTENANCE AND USER MANUAL

6.1 Maintenance Manual

This section contains all the structures and directories of files including source code, object codes as well as documentations. It also contains instructions on how to build and reinstall the program.

6.1.1 Directory Organization

The project is compressed and imported as S4.war file. The war file consists of the following directory organization: Under the S4 directory, there are twenty six JSP pages and five subdirectories: src, WEB_INF, css, js and xml.

/doc: This directory contains documentation for the portal.

/src: This directory contains the JAVA source code. The files are organized into the same subdirectories as those in /WEB_INF to reflect the package hierarchy.

/WEB_INF: This directory contains necessary executable files (class files) corresponding to the src
file. There are two subdirectories, classes and lib and a
web.xml file, which is a web application deployment
descriminator.

/WEB-INF/classes: This subdirectory contains all the
Java class files (and associated resources) required for
the portal. All the java class files are located in
/WEB_INF/classes/csusb/s4/ with three subdirectories:
controller, fasta, blast as it is in corresponding src
directory.

/WEB-INF/lib: The WEB-INF/lib directory consists of
14 jar files and can be classified into four broad
categories based on their functionality.

S4 jar files: Consists of one jar file S4.jar for and
contains library classes for the S4 Expert System.

JSTL jar files: Consists of two jar files jstl.jar
and Xalan.jar. jstl.jar is to use Java Standard Tag
Library in JSP pages. Xalan.jar is an XSLT processor for
transforming XML documents into HTML, text, or other XML
document types. This jar file is required to parse the xml
files in jsp pages.

AXIS jar files: Consists of 11 jar files: axis.jar,
axis-ant.jar, commons-discovery-0.2.jar, commons-logging-
1.0.4.jar, jaxrpc.jar, log4j-1.2.8.jar, saaj.jar,
standard.jar, wsdl4j-1.5.1.jar, xml-apis.jar, and XercesImpl.jar. Axis is a web service tool and it is required to invoke WSFasta web service.

/WEB-INF/web.xml: The web.xml file contains the declarative data for all the servlets in the web application and mapping between the servlet and its URL pattern for all the servlets in the web application.

/Js: The js directory consists of all the javascript files for the dynamic content in jsp pages.

/CSS: The CSS directory consists of Cascading Style Sheet to control the style and layout of multiple Web pages all at once.

JSP pages: There are 7 JSP pages corresponding to every tab in the web application. They are:

Login.jsp: to let the user login and also create a new user.

Search.jsp: to let the user select his choice of search (fasta or blast) or choose the S4 expert system and to get results.

Results.jsp: to let the user view the result and save profiles

S4.jsp: To extend the S4 expert system.

Profiles.jsp: To let the user manage his profiles.
Groups.jsp: To create/manage groups and to share his results with the group members.

Error.jsp: To display system exceptions.

6.1.2 Re-Compile the Source Code

The source code can be re-compiled by “javac” command whenever there are changes in any of the source files. First, the class path should be specified to allow “javac” to find third-party and user-defined classes -- that is, classes that are not Java extensions or part of the Java platform. “-d” options should be specified to set the destination directory for class files. Within the destination directory, “javac” places the class file in a subdirectory under reflecting the package name, creating directories as needed. For example, S4SearchController.java which specifies the package name as csusb.s4.controller has been edited. To re-compile it, go to the /src directory, and type the command “javac -d ../WEB_INF S4SearchControllerServlet.java”, which will compile the source file and place the S4SearchController.class into the directory /WEB_INF/csusb/s4/controller.
6.1.3 Installing on Tomcat Server

In order to be installed, the following software installations are required in the Web Server.

Apache Ant1.6 or higher: a Java-based build tool available for download at http://ant.apache.org/bindownload.cgi

Apache Tomcat 5.0 or higher: an Http Server/Servlet container available at http://tomcat.apache.org/download-55.cgi

MySql 5.0 or higher: a Database Server to support data persistence, available at http://dev.mysql.com/downloads/mysql/4.0.html

J2SE Java Runtime Environment (JRE) 1.4 or higher: to run java applications, available at http://java.sun.com/j2se/1.4.2/download.html

Connector/J MySQL JDBC Driver Ver 3.1 or higher: is official JDBC driver for MySQL available at http://dev.mysql.com/downloads/connector/j/5.0.html

To install S4 the following steps are executed in order.
Step 1: Locate the S4.properties file in doc\installation directory. Change the values of the following fields according to current system settings.

Db.username: specify the username for MySql server
Db.password: specify the password for the username
Db.driver: Specify a JDBC driver

Step 2: Locate the build.xml ant script in doc\installation directory. Change the values of MySql username and password as follows.

```xml
<property name="mysql.params" value=" -u mysqlusername -D s4 -pmysqlpassword" />
<property name="database.dir" value="..\\..\\..\\database" />
```

Run the ant script by executing 'ant' at the command prompt. This will build the project and also will create the required MySQL tables.

Step 3: The portal is finally deployed in Tomcat server by copying the S4.war file to the directory $CATALINA_HOME/webapps/ and Restart Tomcat.

6.2 User Manual

This section contains the user manual to guide the user to use the service properly.
6.2.1 The Login Page

Purpose: to allow the user to login and enter the S4 System. This helps to identify the user and protect his privacy.

Other Accessible Page: None
What is Sequence Similarity Search Portal?

Sequence Similarity searching on molecular sequence databases is one of the first and most important steps in analyzing a newly discovered sequence. If there is a similarity to a sequence that is from another species, then they may be homologous (i.e. sequences that descended from a common ancestral sequence). Knowing the function of a similar/homologous sequence will often give a good indication of the identity of the unknown sequence.

A Smart Sequence Similarity Search System (S4) is a single platform for sequence similarity searching. It allows the user to search the two most popular molecular sequence databases: NCBI and EMBL-EBI database and also to store their results for future references and share their results if needed. The novice users can use it as a learning tool to explore more about sequence similarity searches.

The Sequence Similarity Search Portal is also designed as a teaching aid to be used as an extension to wet lab and the students can submit their search results and reports online. Students can explore more about these two programs in particular by using S4 Expert system. Also note, the curiosity quest link shall guide the users to appropriate references on Sequence Similarity Searches as needed. If you are a new user choose a unique username and password and click on NewUser button.

Figure 6.1. The Login Page

Widgets:

Username Textbox: To enter Username
Password Textbox: To enter Password
Login Button: OnClick the system verifies the password for the given username, and if correct allows the user to enter the portal. On successful Login, the user would be directed to the Search page. If the username didn’t exist or in case of incorrect password, the system
would eject an “invalid login” message and Login Page would be displayed again.

New User Button: OnClick the system would create a new user and new password. If the username already exists the system would throw the message ‘Username Exists’ in Login Page else the user will be directed to the Search Page.

6.2.2 The Search Page

Purpose: to allow the user to enter his sequence similarity choice that is either BLAST or Fasta and also to allow the user to utilize the expert system to decide the search tool and its parameters.

Other Accessible Pages: Result Page, Profile Page, S4 Page, Groups Page
Figure 6.2. The Search Page

Widgets:

SearchType RadioButtons: The user may choose from any of the three options. If the user chose:

Option “Let S4 expert system decide Database and its Parameters”: The S4 Expert System Input fields are revealed. The user may enter an Input Sequence in Fasta Format and click the Submit Button to trigger the Expert System. OnClick a new Window opens and guides the user through the S4 Expert Pages.

Option “I will use BLAST algorithm on NCBI database”: The input fields for BLAST parameters are displayed. Here
the user may enter the corresponding Parameter values and click the BLAST button. OnClick the Blast Sequence Similarity search is executed and the user will be directed to Results page.

Option "I will use Fasta algorithm on EMBL-EBI database":

The input fields for FASTA parameters are displayed. Here the user may enter the corresponding Parameter values and click the RUN FASTA button. OnClick the Fasta Sequence Similarity search is executed and the user will be directed to Results page.

6.2.3 The Result Page

Purpose: Let the user view the results of his sequence similarity search. It also allows the user to save the profiles of his search and view the parameters of the search. If the results are not available, it displays the message “Result Not available” and all the widgets are disabled or hidden.
Figure 6.3. The Result Page
Widgets:

Enter Profile Name TextBox: allows the user to enter a profile name to save the profiles of the displayed result.

Save Profile Button: OnClick saves the profile as requested by the user and refreshes the page with the message "Profile Added".

Show parameters CheckBox: On Selecting the Check box, the parameters of the search are displayed. On deselect the parameters are hidden.

Search Result Form: The Sequence Similarity Search Result is displayed depending upon the search type Blast or Fasta.

6.2.4 The Profiles Page

Purpose: Allows the user to manage the profiles created in the Results Page and also allows the user to extract current results for the saved profile.
### Figure 6.4. The Profile Page

The Profile Page in the Smart Sequence Similarity Search System (S4) is designed to facilitate the identification of similar sequences based on pre-computed profile matrices. The page is divided into sections for easy navigation:

#### Profiles Section
- **Name**: envelope protein blast
- **GroupName**: None
- **Username**: guest
- **Date**: 2006-10-07
- **Type**: blast
- **View**: None
- **Delete**: None

#### BLAST SEARCH Section
- **Search**: 
- **Set subsequence**: From: [ ] To: [ ]
- **Choose database**: [ ]
- **Do CD-Search**: [ ]
- **New**: [BLAST] [Reset Query] [Reset All]
- **Options for advanced blasting**
  - **Limit by**
  - **Computational adjustments**: [ ]
- **Choose filter**: Low complexity [ ] Mask for lookup table only [ ] Mask lower case [ ]
- **Expect**: 10
- **Word Size**: 2 [ ]
- **Matrix**: PAM70
- **Gap Costs**: [ ]
- **Singletons & Extensions**: [ ]
- **Other advanced**: [ ]
- **PHI pattern**: [ ]

#### Additional Features
- **Home**
- **Results**
- **Profile**
- **Groups**

The page also includes a link to the Mozilla Firefox browser and a reference to the California State University, San Bernardino.
Widgets:

Profile Radio Button: Each radio Button option correspond to a result. The user may select any one of them.

View Profile CheckBox: If the user selects the checkbox, the corresponding profile is displayed. On deselect they are hidden.

GetResult Button: OnClick the user can retrieve the current result for the profile selected by the radio button and the user would be directed to Results Page.

Delete Profile Button: OnClick the user can delete the selected profile and the page would be refreshed.

6.2.5 The Groups Page

Purpose: Allows the user to create new group and manage existing group. The user can also add members to the groups and share the profiles with the groups.
Figure 6.5. The Group Page

Widgets:

Manage Users\Groups Radio Button: The user can select one of the three options. The options Manage Group, Manage Group Members and Share Profiles on selection displays the widgets to manage groups, Manage members of the group and share profiles respectively.
Figure 6.6. The Group Page—Create/Delete Groups

Option Manage Groups: Lets the user create a new group or delete an existing group. The user has to select a group from the Select Group DropdownBox and Click on Go to select a group. The user can delete the selected group. If the user needs to create a new group, the user has to select the New group from the DropDownBox widget and specify a name in New Group testbox.
Figure 6.7. The Group Page-Add/Delete Users
Option Manage Users: Allows the user to view the existing group members for a selected group and delete the users or add new users to the group. The user has to select a group from the Select Group DropdownBox and Click on Go to select a group. The user can create new members to the selected group by entering the users in the New Members textbox separated by ‘;’. The Member List checkbox consists of a check box corresponding every member of the group. The Onclick action of Delete User button deletes the user corresponding to the selected user.

Option Share Profile: Allows the user to share profiles with groups. The user has to select a group from the Select Group DropdownBox and Click on Go to select a group. The private profile textbox displays the private profiles of the user, the Shared profile textbox displays the already shared profile to the selected group. The user can move the profiles between the share profile and private profile. Onclick action of Save Changes Button leads to sharing the profiles in share profile textbox to all the members of the selected group and deleting the profiles moved to private result from shared profile.
Figure 6.8. The Group Page—Share Results

6.2.6 The S4 Page

Purpose: Allows the user to decide search tools and parameters through the expert system.

Figure 6.9. The $S^4$ Page
CHAPTER SEVEN

CONCLUSION AND FUTURE DIRECTION

7.1 Conclusion

Sequence similarity search has been widely used by biologists in discovering functions, structure, and biochemical properties of novel biological sequences. There are numerous molecular sequence databases available online for the molecular biologists to conduct sequence similarity searches. It is very difficult for the molecular biologist to keep track of all the available sequence databases and the searches they conducted on different databases.

Sequence Similarity Search Portal is a bioinformatics software program and it is a first step towards integrating the numerous molecular sequence databases available. Currently the system supports the Fasta search on the EMBL-EBI server and BLAST search on NCBI server. It provides a single, powerful user interface for building searches for the above mentioned molecular databases. It also allows the user to save his searches on our server, and to receive those whenever necessary thus helping researchers to maintain a COMPLETE set of search data on
their area of interest. To support group-oriented researches in academic environment, it also provides simple user interfaces to create groups among the users and share search data with other interested researchers. The Sequence Similarity Search Portal also interfaces the S4 expert system to enable the novice molecular biologists to select appropriate databases and corresponding search parameters.

The portal is made possible by the NCBI service offered by the NCBI server and the WSFasta web service offered by the EBI server. The NCBI QBLAST system is supported by the URLAPI that uses direct HTTP-encoded requests to NCBI web server. These encoded requests are directed to the NCBI cgi-bin program at the URL: http://www.ncbi.nlm.nih.gov/blast/Blast.cgi. The WSFasta web service is invoked by downloading the WSDL file available at URL: http://www.ebi.ac.uk/Tools/webservices/wsdl/WSFasta.wsdl. The Apache Axis web services tool is utilized to convert the WSFasta WSDL to java classes. These auto-generated classes have API’s to invoke WSFasta web service.

The Sequence Similarity Search Portal is built around Java Servlet and JSP technology. JavaScript and Java
Standard Tag Library have been utilized extensively to generate dynamic content of the web pages. The Apache Tomcat is used as web server for the project. The persistent data is stored at the web server and is supported by MySQL Database server. The database is connected to the middleware using Java Database Connectivity Driver.

7.2 Future Direction

The Sequence Similarity Search Portal is a first step towards integrating several molecular databases. It can be extended in different directions as follows:

First of all, other popular molecular sequence databases can be identified and integrated, so that the web application would be one stop shop for all the sequence similarity search needs of molecular biologists.

Second, interfaces to automatically resubmit saved searches to the appropriate database server periodically on user selected results can be included. The new results can either be e-mailed to the user or saved at the server.

Third, a result comparison interface can be included to compare the search results obtained from different molecular sequence databases.
Finally API’s can be developed to add new molecular sequence databases to the system by the user. This would let the user to select the database of his interest and add it to the portal as well as utilize the other features of portal.
APPENDIX A

LIST OF DATABASE TABLES
### Table Name: USER

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Key</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>INT</td>
<td>PRIMARY</td>
<td>AUTOINCREMENT</td>
</tr>
<tr>
<td>USERNAME</td>
<td>VARCHAR(50)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PASSWORD</td>
<td>VARCHAR(8)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table Name: PROFILE

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Key</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFILEID</td>
<td>INT(5)</td>
<td>PRIMARY</td>
<td>AUTOINCREMENT</td>
</tr>
<tr>
<td>USERNAME</td>
<td>VARCHAR(50)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROFILENAME</td>
<td>VARCHAR(50)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(50)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table Name: BLAST

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Key</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFILEID</td>
<td>INT(5)</td>
<td>FOREIGN KEY REFERENCES PROFILE (PROFILEID)</td>
<td>AUTOINCREMENT</td>
</tr>
<tr>
<td>PROFILENAME</td>
<td>VARCHAR(50)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPOSITION_BASED_STATISTICS</td>
<td>VARCHAR(10)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

106
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYDATABASE</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>DB_GENETIC_CODE</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>ENDPOINTS</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>ENTREZ_QUERY</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>EXPECT</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>FILTER</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>GAPCOSTS</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>GENETIC_CODE</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>HITLIST_SIZE</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>I_THRESH</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>LAYOUT</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>LCASE_MASK</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>MATRIX_NAME</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>NUCL_PENALTY</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>NUCL_REWARD</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>OTHER_ADVANCED</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>PERC_IDENT</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>PHI_PATTERN</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>QUERY</td>
<td>VARCHAR(7999)</td>
</tr>
<tr>
<td>QUERY_BELIEVE.DEFLINE</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>QUERY_FROM</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>QUERY_TO</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>SEARCHSP_EFF</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>SERVICE</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>Field Name</td>
<td>Type</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
</tr>
<tr>
<td>THRESHOLD</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>UNGAPPED_ALIGNMENT</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>WORD_SIZE</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>PROFILEID</td>
<td>INT</td>
</tr>
<tr>
<td>PROFILENAME</td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>INPUTSEQUENCE</td>
<td>VARCHAR(7999)</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>VARCHAR(20)</td>
</tr>
<tr>
<td>MYDATABASE</td>
<td>VARCHAR(30)</td>
</tr>
<tr>
<td>MOLTYPE</td>
<td>VARCHAR(15)</td>
</tr>
<tr>
<td>HISTOGRAM</td>
<td>VARCHAR(5)</td>
</tr>
<tr>
<td>NUCLEOTIDE</td>
<td>VARCHAR(5)</td>
</tr>
<tr>
<td>TOPSTRAND</td>
<td>VARCHAR(5)</td>
</tr>
<tr>
<td>BOTTOMSTRAND</td>
<td>VARCHAR(5)</td>
</tr>
<tr>
<td>GAPOPEN</td>
<td>INT(3)</td>
</tr>
<tr>
<td>GAPEXT</td>
<td>INT(3)</td>
</tr>
<tr>
<td>SCORES</td>
<td>INT(3)</td>
</tr>
<tr>
<td>ALIGNMENTS</td>
<td>INT(3)</td>
</tr>
<tr>
<td>KTUP</td>
<td>INT(2)</td>
</tr>
<tr>
<td>Field Name</td>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>GROUPID</td>
<td>INT(5)</td>
</tr>
<tr>
<td>GROUPNAME</td>
<td>VARCHAR(50)</td>
</tr>
<tr>
<td>OWNER</td>
<td>VARCHAR(50)</td>
</tr>
</tbody>
</table>

**TABLENAME: GROUPMEMBERS**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Key</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUPID</td>
<td>INT(5)</td>
<td>FOREIGN KEY</td>
<td>REFERENCES GROUP (GROUPID)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON DELETE {CASCADE}</td>
<td></td>
</tr>
<tr>
<td>Field Name</td>
<td>Type</td>
<td>Key</td>
<td>Extra</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>GROUPID</td>
<td>INT(5)</td>
<td>FOREIGN KEY</td>
<td>REFERENCES GROUP (GROUPID)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON DELETE {CASCADE}</td>
<td></td>
</tr>
<tr>
<td>PROFILEID</td>
<td>INT(5)</td>
<td>FOREIGN KEY</td>
<td>REFERENCES PROFILE (PROFILEID)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON DELETE {CASCADE}</td>
<td></td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY


[B16] Labarga, A. et al., European Bioinformatics Institute, Hinxton, United Kingdom, “Web Services at the European Bioinformatics Institute,”


[B19] Mahmoud, Q., ”Servlets and JSP Pages Best Practices,”
[B20] NCBI.NLM.NIH. Service Desk, "URLAPI for NCBI BLAST splitd System" Internet: