12-2017

WEB APPLICATION FOR GRADUATE COURSE RECOMMENDATION SYSTEM

Sayali Dhumal

WEB APPLICATION FOR GRADUATE COURSE RECOMMENDATION SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Sayali Mahendra Dhumal

December 2017

WEB APPLICATION FOR GRADUATE COURSE RECOMMENDATION SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Sayali Mahendra Dhumal

December 2017

Approved by:

Dr. Josephine Mendoza, Advisor, Computer Science and Engineering

Dr. Kerstin Voigt, Committee Member

Dr. Tong Lai Yu, Committee Member

ABSTRACT

The main aim of the course advising system is to build a course recommendation path for students to help them plan courses to successfully graduate on time. The recommendation path displays the list of courses a student can take in each quarter from the first quarter after admission until the graduation quarter. The courses are filtered as per the student's interest obtained from a questionnaire asked to the student.

The business logic involves building the recommendation algorithm. Also, the application is functionality-tested end-to-end by using nightwatch.js which is built on top of node.js. Test cases are written for every module and implemented while building the application.

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to my advisor, mentor and supporter Dr. Josephine Mendoza for giving me guidance and knowledge throughout this project. I would also like to thank Dr. Kerstin Voigt and Dr. Tong Lai Yu for being the committee members and for their valuable advice and support.

Lastly, this Master's degree has been made possible not only through the financial support but most importantly the moral support of my parents --- thank you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Background

The motivation behind collecting course enrollment data is to figure out courses in demand among students and to keep track of student's records. Based on the enrollment information the program administration plans the course list. Before starting a quarter a student needs to consult with the adviser. Typically, this process is very time-consuming. The process starts with a student setting up an appointment on a first come-first serve basis. If a student does not make an advising appointment, a registration hold is put on the student's record which prevents him/her from enrolling in courses.

These sometimes may result in problems such as student enrolling in courses before fulfilling its required prerequisites, courses not available in that quarter, selecting unsuitable elective courses. Selecting courses is entirely another decision a student should be making. In selecting courses, he/she may have problems like choosing courses that are offered in other quarters, minimum course units or low GPA because of selecting courses which he/she is not interested.

Purpose

Every student's goal is to graduate in the shortest time possible. The idea of this project is to determine an algorithm that incorporates the course offering data along with student constraints of work schedule, elective course interests, and preferred class schedule. This algorithm will be used to present a path of recommended courses for the student to take every quarter and graduate in time. Graduating in a timely manner contributes to student success and will meet the California State University's Graduation Initiative 2025 targets. Data analytics based on the aggregated recommendation paths of all current graduate students in Computer Science can provide invaluable information on several students that need to take a course and assist the School of Computer Science & Engineering (CSE) Director in the scheduling of courses and need for faculty and resources to support the demand.

Existing System

In the existing system, the CSE Graduate Coordinator for Advising (GCA) needs to take care of many aspects into consideration in advising graduate students. The GCA uses five sources of information in advising: graduate student's file folder, PeopleSoft Student System, MSCS Advising Database System, CSE Course Schedule for the quarter and CSE Flowchart of MSCS Courses. In the current system, the GCA refers to the latest advising sheet filed in the student's folder for the previous advising notes, the Graduate Decision

Form for pre-requisites required to be taken and whether the Graduate Writing Requirement has been satisfied either through a writing class or passing the WRE or waiver; the PeopleSoft Student System for the transcript for courses already taken and currently registered for and current GPA, status of the student (conditionally classified, classified, advanced to candidacy); CSE Course Schedule for the next quarter for courses that are offered and the CSE Flowchart of MSCS Courses to advise students on what courses to take for the next quarter not considering the time constraints of the student. The student is then left with choosing the elective courses based on his/her day/time constraints because of work and personal reasons. This process is repeated by the GCA for every student. In this project, the advisor time can be greatly reduced by the production of the path of recommended courses to take per quarter which will serve as a roadmap for the student in course registration. Thus, the advisor time which can be better spent with talking with the student about relevant issues regarding difficulties and challenges encountered and career counseling.

CHAPTER TWO

SYSTEM ANALYSIS

Proposed System

This web application is a course advising system that builds course advising path for students to help them plan courses to successfully graduate on time. The recommended path will display the list of courses the student can take in each quarter from the first quarter upon admission until the quarter the student is expected to graduate. The courses are filtered based on answers obtained from the student through a questionnaire – master option (project, thesis, comprehensive exam) desired, days/time of the week preferred to take courses, academic topics interest (to assist in the choice of electives).

The business logic involves building recommendation algorithm. Application functionality is tested end-to-end by using nightwatch.js which is built on top of node.js. Test cases are written for every module and implemented while building the application to ensure that the application is not broken. User authentication is done by a registration form. Each student can register using Coyote ID assigned by the University as user ID and a password of choice.

After registration, a student can login using his/her details. Every student has a different path for courses to be taken depending on the student's status (full-time or part-time) and preferences (based on the answers to the questionnaire).

System Requirement Specification

Hardware Requirements

- Laptop or PC running chrome browser for testing.
- PC for development.

Software Requirements

- Testing: Nightwatch.js - 0.9.16
- Backend: Express.js - 4.15.2
- Client-side: AngularJS 1.6, HTML 5
- Operating Systems: Mac OS X, Windows 10
- Database System: PostgreSQL 10.0
- IDE tools: WebStorm 171.4249.40

MVC (Model-View-Controller) architecture is implemented for the system. MVC is a software architectural pattern that implements user interfaces on the computer that represents the express model, REST (Representational State Transfer) controller, and angular factory view for this project. REST is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce performance, scalability, and modifiability, that enable services to work best on the Web.

Software Used

Node.js Framework

The popular platform for executing server-side code is known as node.js. Node is used in the back-end of the application and it helps in establishing a connection with the server. It is an open source framework. Ryan Dahl developed node initially in 2009 with a concept of a software that focuses on dynamically updating the progress instead of querying the server. [4]

Development of servers and modules collection that have core functionality are some of the key features of Node framework. Each module represents different functions -- cryptography, filesystem I/O, networking and data streams. The npm website has a huge collection of libraries that can be re-used by anyone working on a project.

In this project, node.js package npm is used to install all the dependencies required for the project. Figure 1 shows all the dependencies that are installed using npm.

Figure 1. Project Dependencies Using Npm.

Nightwatch.js

Nightwatch.js is an end-to-end testing framework built on top of the Node.js. Nightwatch.js uses a selenium server for executing the test cases. Nightwatch.js supports reusability by having export and import functionality.[3]

In this project, nightwatch.js is used to write test cases for each section of the project. Figure 2 shows the login page test case which is written using nightwatch.js.

```
module.exports = {
    'Does redirect to login page': function (client) {
        var login = client.page.loginPage();
        login
            .navigate()
            .waitForElementVisible('@Header', 1000)
            .assert.visible('@CoyoteID')
    },

    'Does login successful and redirects to student home page': function (client) {
        var login = client.page.loginPage();
        var loginProps = login.props;

        login
            .click('@CoyoteID')
            .setValue('@CoyoteID', loginProps.StudentID)
            .click('@Password')
            .setValue('@Password', loginProps.Studentpassword)
            .waitForElementVisible('@LoginButton', 1000)
            .assert.visible('@LoginButton')
            .click('@LoginButton');
        client
            .pause(1000)
            .assert.urlEquals(loginProps.StudentURL)
    }
}
```

Figure 2. Login Test Cases Using Nightwatch.js.

PostgreSQL

PostgreSQL is the database system used for this application. PostgreSQL is a cross-platform, open-source and object-relational database. It provides the following advantages:

- high performance,
- supports numerous datatypes and
- strong support from the community as well as third-parties.

There are fifteen tables created in the database using PostgreSQL. They will be described below in Tables 1 to 15.

LEGEND: PK-Primary Key, FK-Foreign Key, NN-NOT NULL, CK-Composite Primary Key

TABLE NAME**:** *ADMINISTRATOR* – specialization (subclass) of superclass USER.

Table 1. Administrator Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by the University.	numeric	PK, FK referencing PK of USER table.

TABLE NAME: *ADVISOR* – specialization (subclass) of superclass USER.

Table 2. Advisor Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by the University.	numeric	PK, FK referencing PK of USER table.

TABLE NAME: *CORE_COURSE* – specialization (subclass) of superclass COURSE.

Table 3. Core Course Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
course_id	3-digit unique ID assigned by CSE Department.	numeric	PK, FK referencing PK of COURSE table.

TABLE NAME: *COURSE* – generalization (superclass) that describes the details of a course offered in the MSCS Program.

Table 4. Course Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
course_id	3-digit unique ID assigned by CSE Department.	numeric	PK
has_lab	1- course has a laboratory component	boolean	NN

	0- course has no laboratory component		
course_level	graduate or undergraduate.	variable-length character	NN
units	total number of units assigned to the course.	integer	NN
course_dept	abbreviation of the department that offers the course.	variable-length character	NN
name	name of the course.	variable-length character	NN

TABLE NAME: *COURSE_PREREQUISITE* – specialization (subclass) of superclass COURSE that contains the information of a course that is a prerequisite to another course.

Table 5. Course Prerequisites Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
course_id	3-digit unique ID assigned by CSE Department.	numeric	CK, FK referencing PK of COURSE table.
prerequisite_id	3-digit unique ID assigned by CSE Department to the prerequisite.	numeric	CK, FK referencing PK of COURSE table.

TABLE NAME: *COURSE_SCHEDULE* – provides information on when (quarter/year, day of the week, time) a course is offered and who teaches the course.

Table 6. Course Schedule Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
course_id	3-digit unique ID assigned by CSE Department.	numeric	CK, FK referencing PK of COURSE table.
year	4-digit year when the course is offered.	numeric	CK
quarter	quarter (Fall, Winter, Spring, Summer) when the course is offered.	variable-length character	CK
session_no	differentiates between the different offerings of the same course in the same	numeric	NN

	quarter i.e. if the course is offered twice in a quarter, then S1 schedule will have session no.=1 and S2 schedule will have session no.=2.		
instructor	name of the instructor who teaches the course.	variable-length character	NN
course_day	day of the week (MondayAndWednesday, TuesdayAndThursday, Friday) when the lecture part of the course is offered.	variable-length character	
course_start_time	start time (in 24-hr format) when the	time	

	lecture part of the course is offered Example**:** 19:50		
course_end_time	end time (in 24-hr format) when the lecture part of the course is offered Example: 17:20	time	
lab_day	day of the week (MondayAndWednesday, TuesdayAndThursday, Friday) when the laboratory part of the course is offered.	variable-length character	
lab_start_time	start time (in 24-hr format) when the laboratory part of the course is offered	time	

	Example**:** 9:50		
lab_end_time	end time (in 24-hr format) when the laboratory part of the course is offered. Example**:** 10:50	time	

TABLE NAME: *COURSE_TAKEN* – indicates what and when courses were taken by a student and the grades earned by the student.

Table 7. Course Taken Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by the University.	numeric	CK, FK referencing PK of STUDENT table
course_id	3-digit unique ID assigned by CSE Department.	numeric	CK, FK referencing PK of COURSE table

quarter_year	year and quarter (e.g. Fall2017) when the course was taken.	variable-length character	CK
day_time	day and time. Time is in 24-hr format. (e.g. {“Day”: “MondayAndWednesday”, “time”: 9:50:00-07}) when the course was taken.	json	NN
instructor	name of the instructor who taught the course in that quarter.	variable-length character	NN
grade	letter grade (A, A-, B+, B, B-, C+, C, C-, D, F, I, RP) obtained by the student.	variable-length character	

TABLE NAME: *DAY_PREFERENCE* – indicates what days of the week a student prefers to take a course.

Table 8. Day Preference Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by the University.	numeric	CK, FK referencing PK of STUDENT_ PREFERENCE table
day_preference	day of the week (MondayAndWednesday, TuesdayAndThursday, Friday) preferred by the student to take the course as answered in the questionnaire.	variable-length character	CK

TABLE NAME: *ELECTIVE_COURSE* - specialization (subclass) of superclass COURSE.

Table 9. Elective Course Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
course_id	3-digit unique ID assigned by CSE Department.	numeric	PK, FK referencing PK of COURSE table.

TABLE NAME: *QUARTER_OFFERED* – indicates what quarter(s) a course is offered.

Table 10. Quarter Offered Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
course_id	3-digit unique ID assigned by CSE Department.	numeric	CK, FK referencing PK of COURSE table.
quarter	quarter when the course is offered.	variable-length character	CK

TABLE NAME: *STUDENT* - indicates whether the student is international/domestic, full-time/part-time and lists the recommended courses to take.

Table 11. Student Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by the University.	numeric	PK, FK referencing PK of USER table.

residency	1- international student; 0 - domestic student	boolean	NN
ft_or_pt	1 – part-time; 0 – full-time	boolean	NN
current_recommend-ation_path	recommendation path generated by the system	json	

TABLE NAME: *STUDENT_PREFERENCE*- indicates all the preferences like student selected in the questionnaire.

Table 12. Student Preference Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by the University.	numeric	PK, FK referencing PK of STUDENT table.

degree_preference	Degree preference (Project, Thesis, Comprehensive Exam) with which the student wants to graduate.	variable-length character	NN
course_count_ preference	total number of courses the student wants to take in a quarter.	numeric	NN
lecture_preference	1- student wants to take courses with laboratory; 0- student wants to take courses with no laboratory.	boolean	NN

summer_course_ preference	1- student wants to take summer courses; 0- student does not want to take summer courses	boolean	NN
course_overload_ preference	1 - student wants to take more than 16 units; 0 – student wants to take at the most 16 units	boolean	NN
independent_study _preference	1 - student wants to take independent study; 0 - student does not want to take independent study;	boolean	NN

TABLE NAME: *STUDENT_PREREQUISITE*- lists all the pre-requisite courses that the student needs to take as indicated from the Graduate Decision Form.

Table 13. Student Prerequisite Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by the University.	numeric	CK, FK referencing PK of STUDENT table
course_id	3-digit unique ID assigned by CSE Department.	numeric	CK, FK referencing PK of COURSE table

TABLE NAME: *TIME_PREFERENCE* - describes what time of the day the student prefers to take courses.

Table 14. Time Preference Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by CSE Department.	numeric	CK,

			FK referencing PK of STUDENT_ PREFERENCE table
time_ preference	Time of the day (Morning, Afternoon, Evening) preferred by the student to take the course as answered in the questionnaire.	variable-length character	CK

TABLE NAME: *USER* - contains information about each registered user in the system.

Table 15. User Database Table

ATTRIBUTE	DESCRIPTION	DATA TYPE	COMMENTS
coyote_id	10-digit unique ID assigned by the University.	numeric	PK

name	first middle and last name of the student.	variable-length character	NN
email_id	Coyote email ID (coyoteID@coyote.csusb.edu) of the student.	variable-length character	NN
phone	telephone contact number of the student.	numeric	NN
address	address (street address, city, state, zip code) of the student.	json	NN
date_of_birth	date of birth (MM/DD/YYYY) of the user	date	NN
password	encrypted password of the user.	variable-length character	NN

Package Definitions Used in the Project

There are some packages used in the development of this application. These packages have some class files that will be used in the development.

Pg

Pg package is used to establish connection with the database. Pg is a pure JavaScript client. Some of the features of PostgreSQL it supports are parametrized queries, asynchronous notifications, import & export functionality.[6]

```
function deleteUser(req, res, next) {
    const coyote_id= req.query.coyote_id;
    console.log("inside")
    pg.connect(connectionString, function(err, client, done){
        // Handle connection errors
        if(err) {
            done();
            console.log(err);
            return res.status(500).json({success: false, data: err});
        }
        const query = client.query('DELETE FROM "user" WHERE coyote_id =($1); ',
            [coyote_id]);
        query.on('end', function () {
            done();
            return res.json("success");
        });
    });
}
```

Figure 3. Connection with Database Using Pg.

Express

Express is a minimal web framework for node. Some of the main features of express are robust routing, high performance, redirection & caching. In this project, express was used to create the server quickly and is developed on top of node.js.

```
const express = require('express');
const router = express.Router();
const pg = require('pg');
const path = require('path');

const loginQueries = require('../queries/login.queries');
const questionnaireQueries = require('../queries/questionnaire.queries');
const userQueries = require('../queries/user.queries');
const courseQueries = require('../queries/course.queries');

router.get('/', function(req, res, next){
    console.log(path.join(
        __dirname, '..','..','index.html'))
    res.sendFile(path.join(
    __dirname, '..','..','index.html'));
});



router.get('/login', loginQueries.getUser);

module.exports = router;
```

Figure 4. Server Side Code Using Express.js.

Cookie Parser

Cookie parser enables parsing of cookie headers as well as in passing objects keyed with cookie names. If secret parameter is used cookie signing can be enabled as well. In the project, cookie parser is used in the login and sign-out functions. Using cookies helps to restore the data that is used throughout the project even if the page is reloaded. Therefore, user role and Coyote ID are stored in the cookie to handle browser refresh functionality.

```
UserService.getUserRole(vm.userID).then(
    function success(response) {
        vm.role = response.data[0];
        cookieData = {
            auth: vm.data.userID,
            role: vm.role
        };
        $cookies.put('usedata',cookieData);
        $state.go('root',{userID : vm.data.coyote_id, userRole: vm.role});
    },function error(response) {
        console.log(response)
    }
)
```

Figure 5. Setting Cookies in Login.

Nodemailer

Nodemailer is the node.js application for sending emails. It adds HTML content as well as plain text to the mail and attaches files to messages. In this project, Nodemailer is used to send randomly generated code to email to reset account password.[5]

```
function resetPassword(req, res,next) {
    var email = req.body.email;
    var code = req.body.code;
    var text = 'Code to reset password is: '+ code;
    var mailOptions = {
        from: '######@gmail.com',
        to: email,
        subject: 'Reset password',
        text: text
    };
    var transporter = nodemailer.createTransport({
        service: 'gmail',
        auth: {
            user: '######@gmail.com',
            pass: '############'
        }
    });

    transporter.sendMail(mailOptions, function(error, info){
        if(error){
            console.log(error);
            res.json({yo: 'error'});
        }else{
            console.log('Message sent: ' + info.response);
            res.json({yo: info.response});
        }
    });
}
```

Figure 6. Sending Reset Code to Email Using Nodemailer.

Cryptr

Cryptr is the node.js module uses the AES algorithm for encryption and decryption of strings. In this project, this cryptr is used in the login and signup modules to encrypt the password string before storing in the database and decrypt during the authentication process. [2]

```
function changePassword(req, res,next) {
    const coyote_id = req.body.coyote_id;
    const password =  cryptr.encrypt(req.body.password);

    pg.connect(connectionString, function(err, client, done){
        if(err) {
            done();
            console.log(err);
            return res.status(500).json({success: false, data: err});
        }
        const query = client.query('UPDATE "user" SET  password=($1) WHERE coyote_id=($2)',[password,coyote_id]);
        query.on('end', function(){
            done();
            return res.json('Success');
        });
    });
}
```

Figure 7. Use of Cryptr to Encrypt Password.

CHAPTER THREE

SYSTEM DESIGN

Data Flow Diagrams

Data flow diagram represents how the data flows in the system. It also provides a detailed description of each system component. The following are the data flow diagrams of Login and Registration page, Administrator home page, Student home page and Advisor home page.

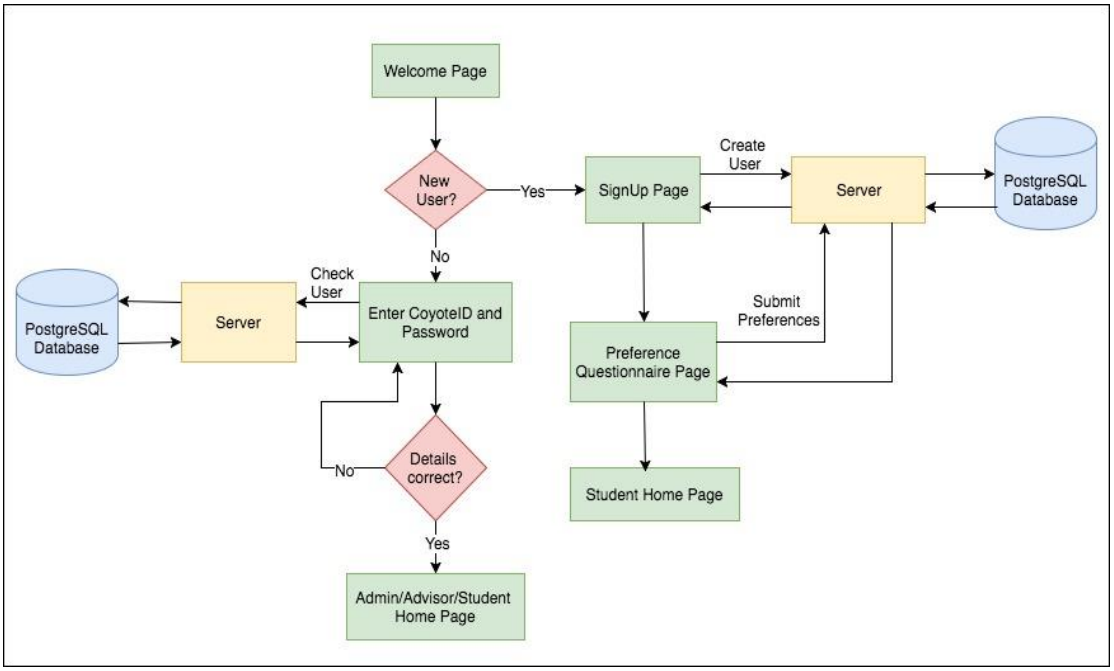

Figure 8.Data Flow Diagram for Login and Registration.

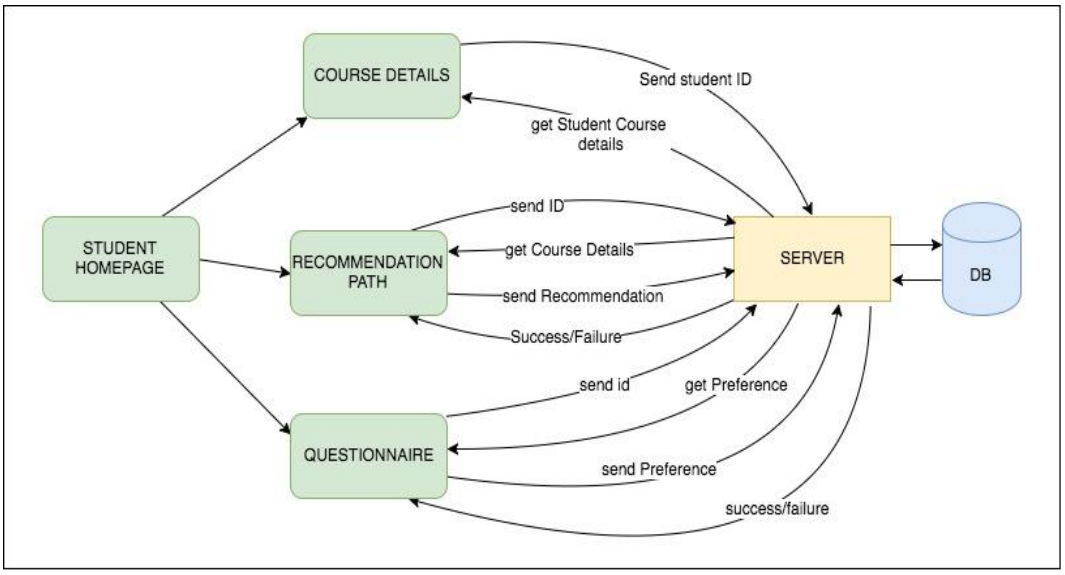

Figure 9. Data Flow Diagram for Student Actions.

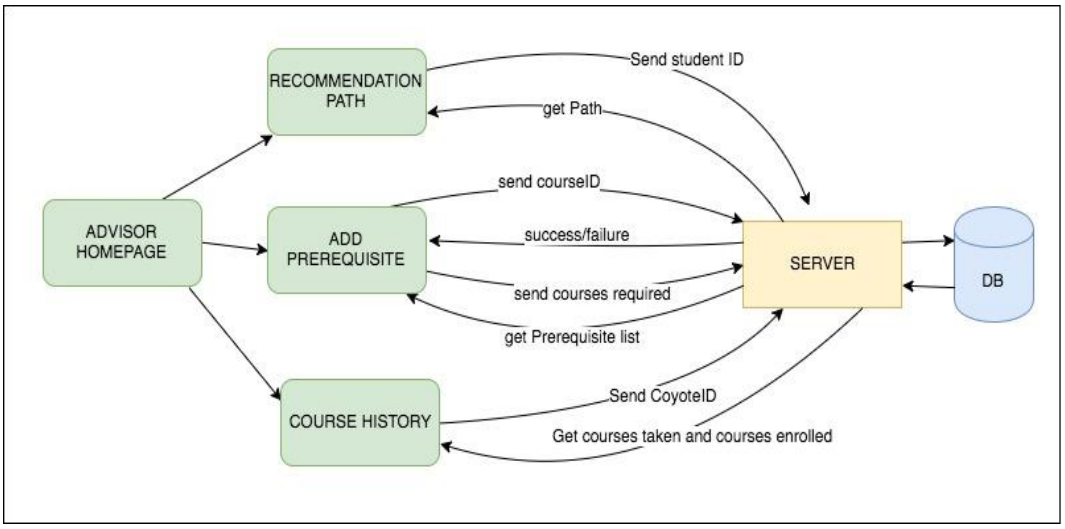

Figure 10. Data Flow Diagram for Advisor Actions.

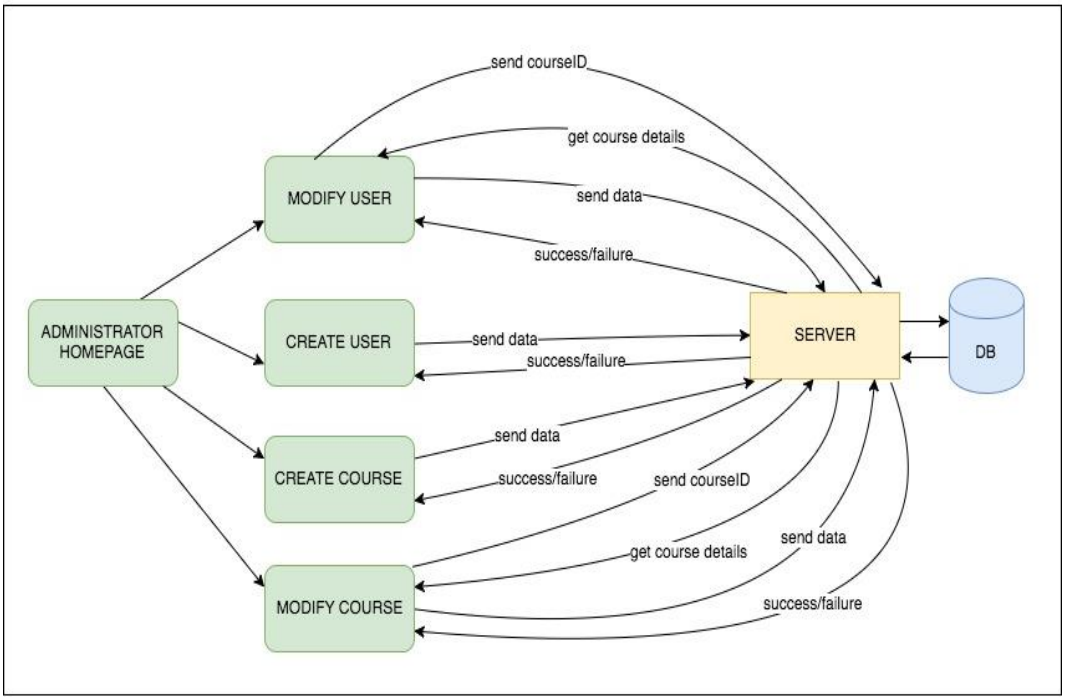

Figure 11. Data Flow Diagram for Administrator Actions.

UML Diagrams

Use Case Diagrams

A use case diagram is the graphical representation of user's interaction with the application. It also represents the relation between the user and other user cases in which the user is involved. Below are the use case diagrams for administrator, advisor and student.

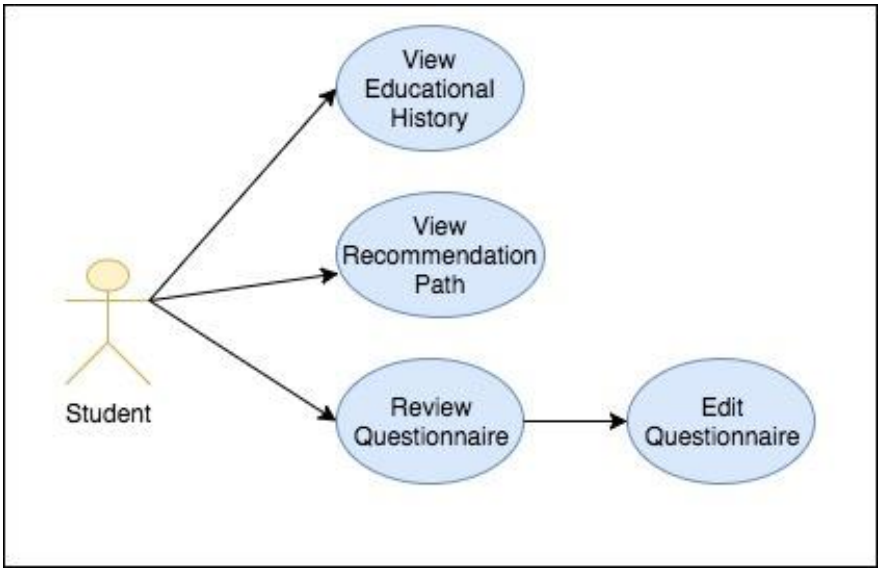

Figure 12. Use Case Diagram for Student.

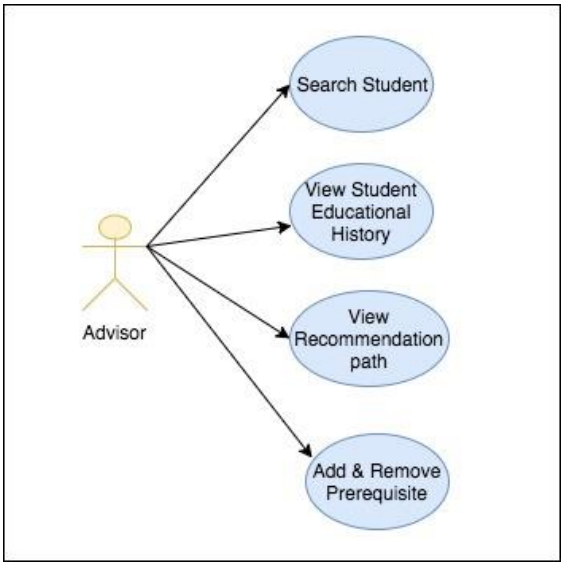

Figure 13. Use Case Diagram for Advisor.

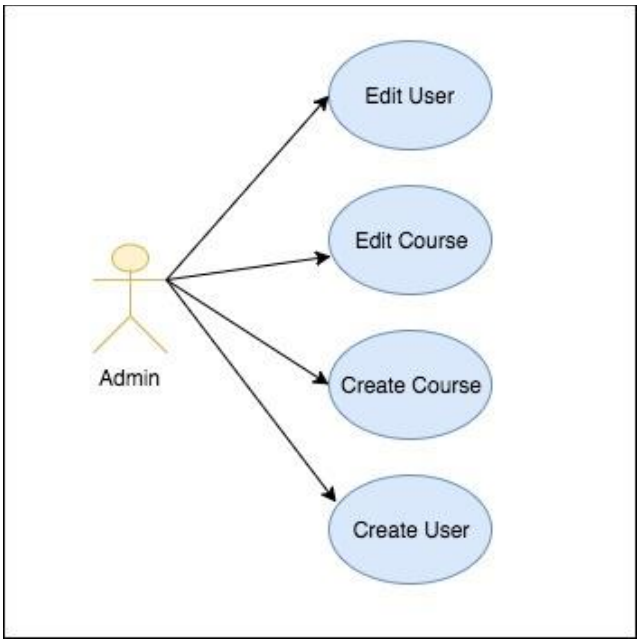

Figure 14. Use Case Diagram for Administrator.

Sequence Diagrams

A sequence diagram is an interaction diagram that represents how processes operate with one another and in what order. Below are the sequence diagrams for the Administrator, Student and the Advisor functionalities.

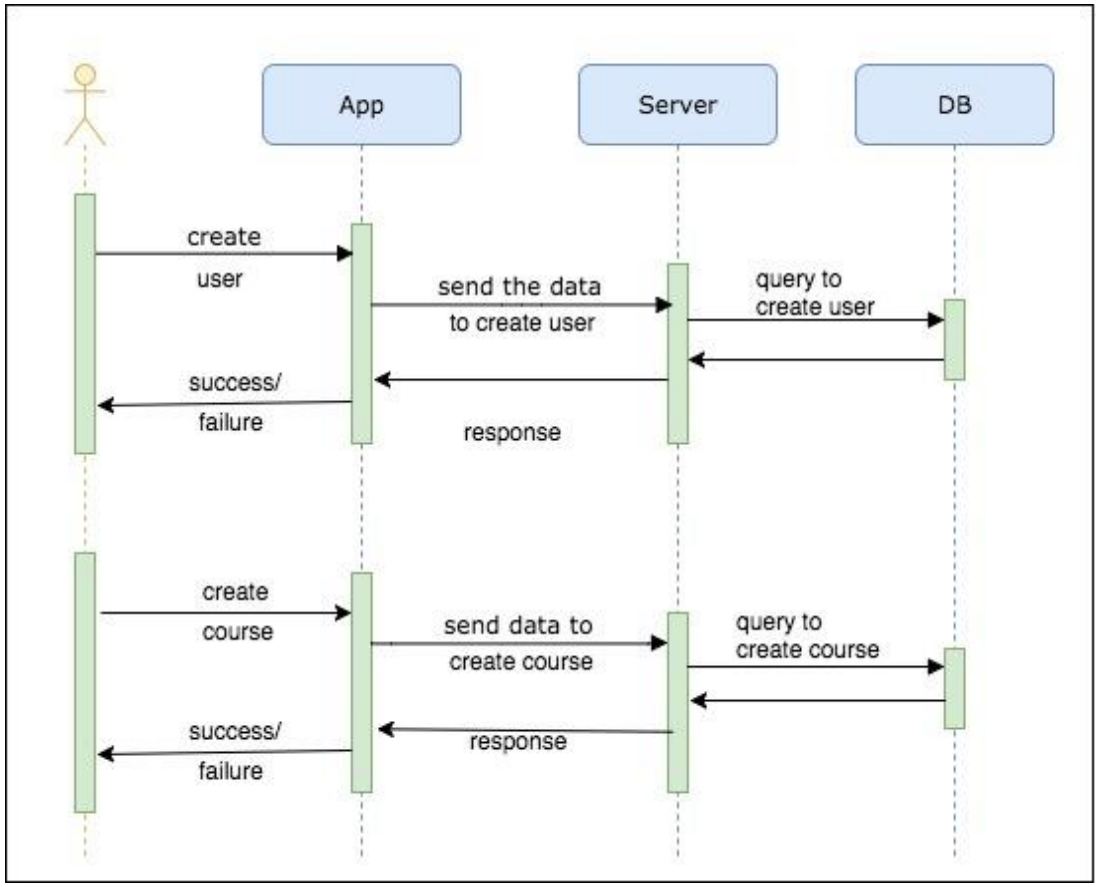

Figure 15. Sequence Diagram for Create User and Course.

Figure 15 shows how to create a user and a course. To create a user the administrator will click on the create user button. The application will then redirect to the create user page. The administrator enters the user information there and then that information is sent to the server. The server then saves this user information to the database and returns a response to the application. The

application will then display the success or error response. Similar steps will be followed to create a course.

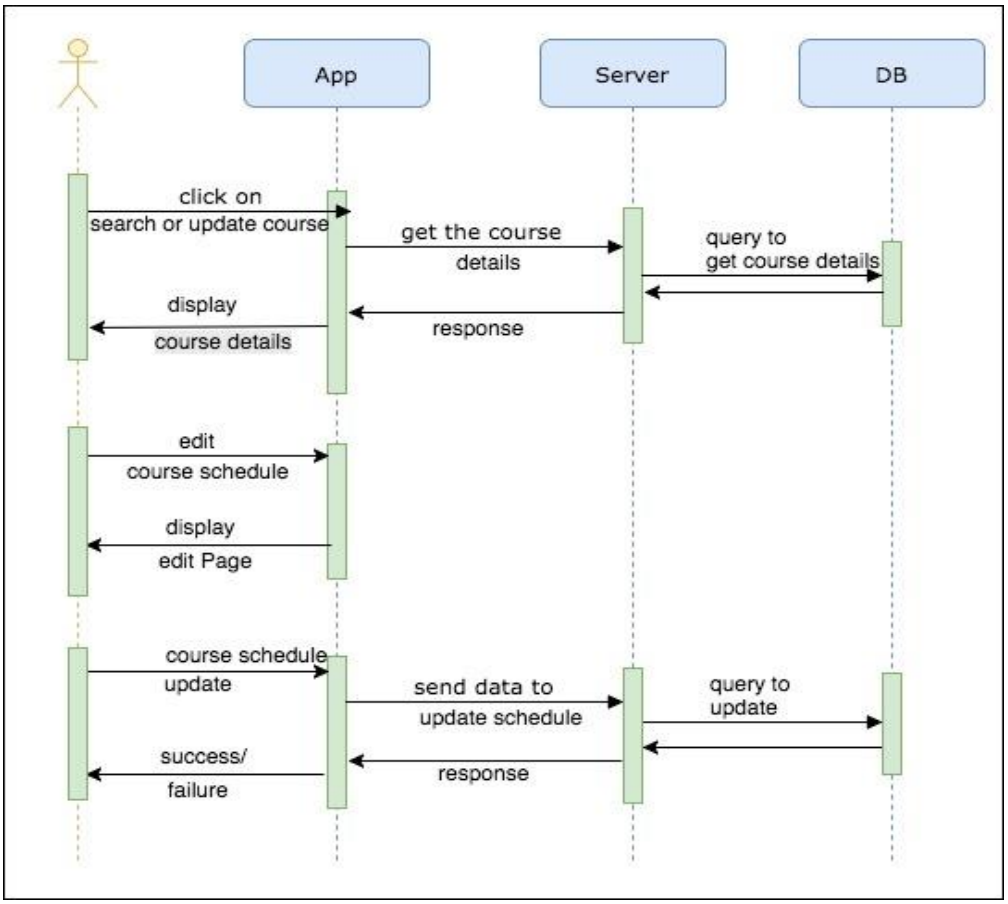

Figure 16. Sequence Diagram for Update Course.

Figure 16 displays how to update the course information. To edit the course, first, the administrator searches for the course information. He/she enters the course ID or course name and clicks on the search button. The entered information is then used to get the course details from the database and the details are then displayed to the user. To edit the information the administrator clicks on the schedule to be edited and the application redirects to the edit page. He/she then makes the changes and clicks on the update button to save the

changes. The updated data is then sent to the server which saves the data into the database and returns a response to the application. The application will then display the success or error response.

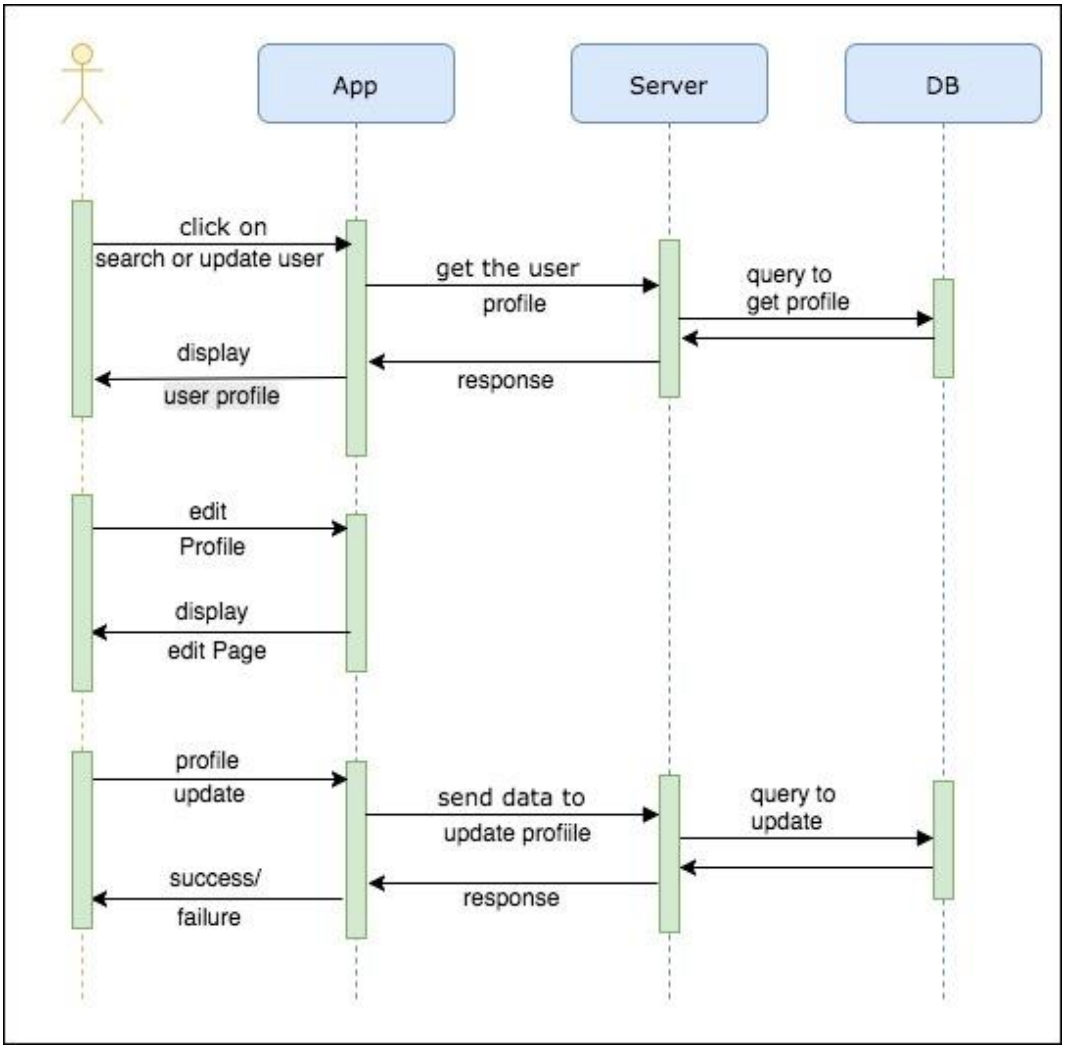

Figure 17. Sequence Diagram for Update User Profile.

Figure 17 illustrates how to update the user profile. To update the user profile, administrator clicks on the search and update button. On clicking that button, it shows the user search page. The search page finds the user by partially entering student name or student Coyote ID. Once the user is found,

click on view button to see the user profile. There is an “EDIT” button on this page. By clicking on that button, the administrator can then edit the user information as required and then click update button to update the information on server.

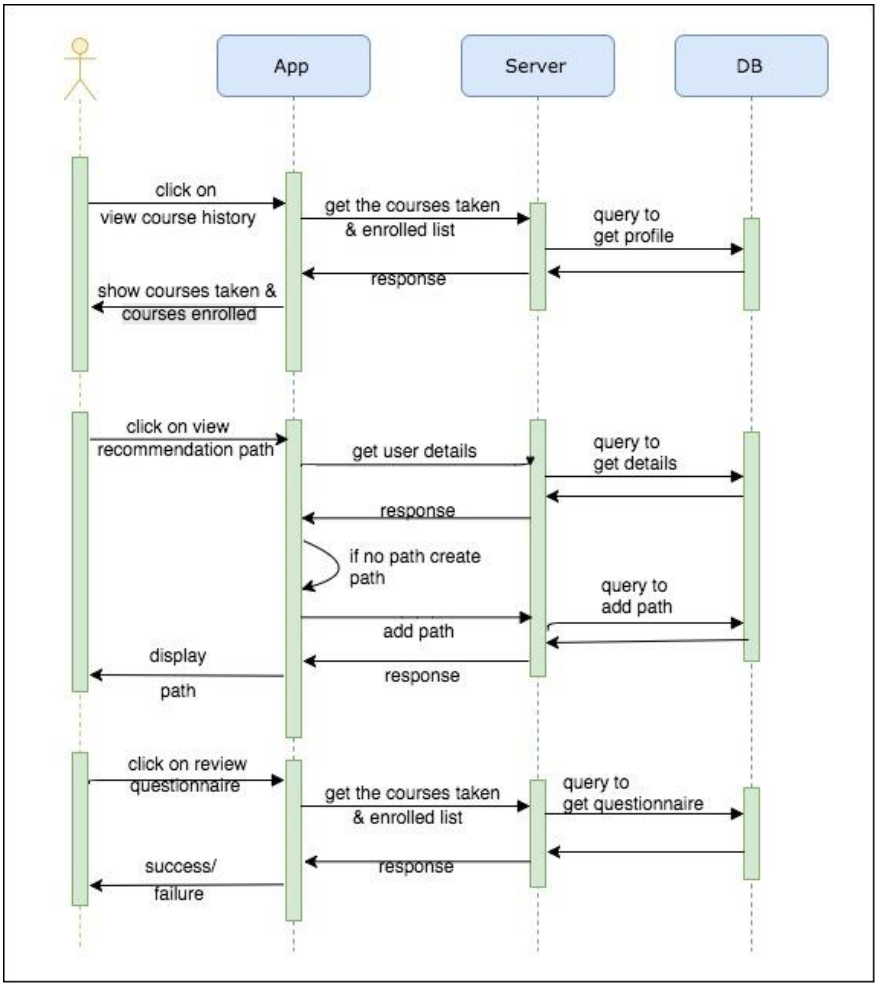

Figure 18. Sequence Diagram for Student Role.

The student homepage has three cards. They are course history, recommendation path and review questionnaire. Figure 18 displays how a student can use these cards. When the student clicks on course history card,

he/she can get his/her education history like courses he/she enrolled, has taken and view GPA for the courses he/she has taken. On clicking recommendation path, he/she can view the path of courses he/she needs to take from the current quarter to the final quarter. If review questionnaire card is clicked, he/she can view his/her existing preferences and modify them as needed.

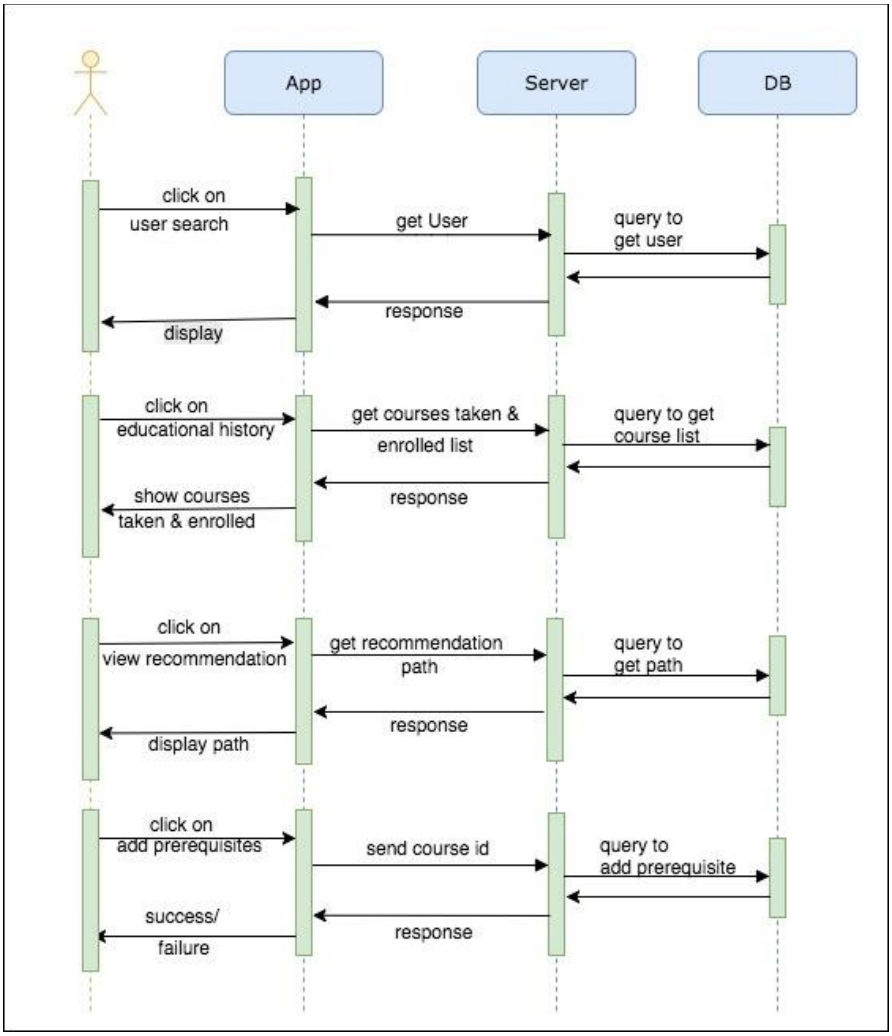

Figure 19. Sequence Diagram for Advisor Role.

Figure 19 gives the advisor functionalities. When the advisor clicks on user search by entering the student name or Coyote ID, a “get user” service is

executed on the server and retrieves the corresponding user record. An advisor can then view the educational history of this student, by using “get courses taken” service on the server side and retrieves the educational history of that individual. On clicking the view recommendation button, the recommendation path is retrieved from the server. The advisor can go to “add prerequisites” page to add any prerequisite course recommended for the student. This uses “post query” to add information to database. On successful transaction, a success message will be displayed.

CHAPTER FOUR

RECOMMENDATION ALGORITHM

The recommendation algorithm generates a path of courses for a student to take in the first quarter up to the last quarter using what courses are offered when for the quarter; pre-requisites indicated in the graduate decision form when the student was admitted into the program; and the preferences obtained from the questionnaire submitted by the student -- degree option (project, thesis, comprehensive exam), days/time preferred, subject interests for elective courses. This path contains all the prerequisites, core courses and electives that the student is recommended to take for each quarter.

Figure 20 shows the recommendation algorithm that is used to generate the path. In this algorithm, highest priority is given to prerequisite courses i.e. first the prerequisite courses are added to the path because the student cannot register for core course until the prerequisite for that course is completed. For example, a student who needs CSE 430 prerequisite course cannot register for CSE 630 until CSE 430 course is taken and passed. After prerequisites are added, the core courses not yet taken by the student will be added to the path. The quarter to take the core course is selected depending on which quarter its prerequisite is taken. Last to be added to the path will be the elective subjects which will be selected based on the preferences questionnaire submitted by the student.

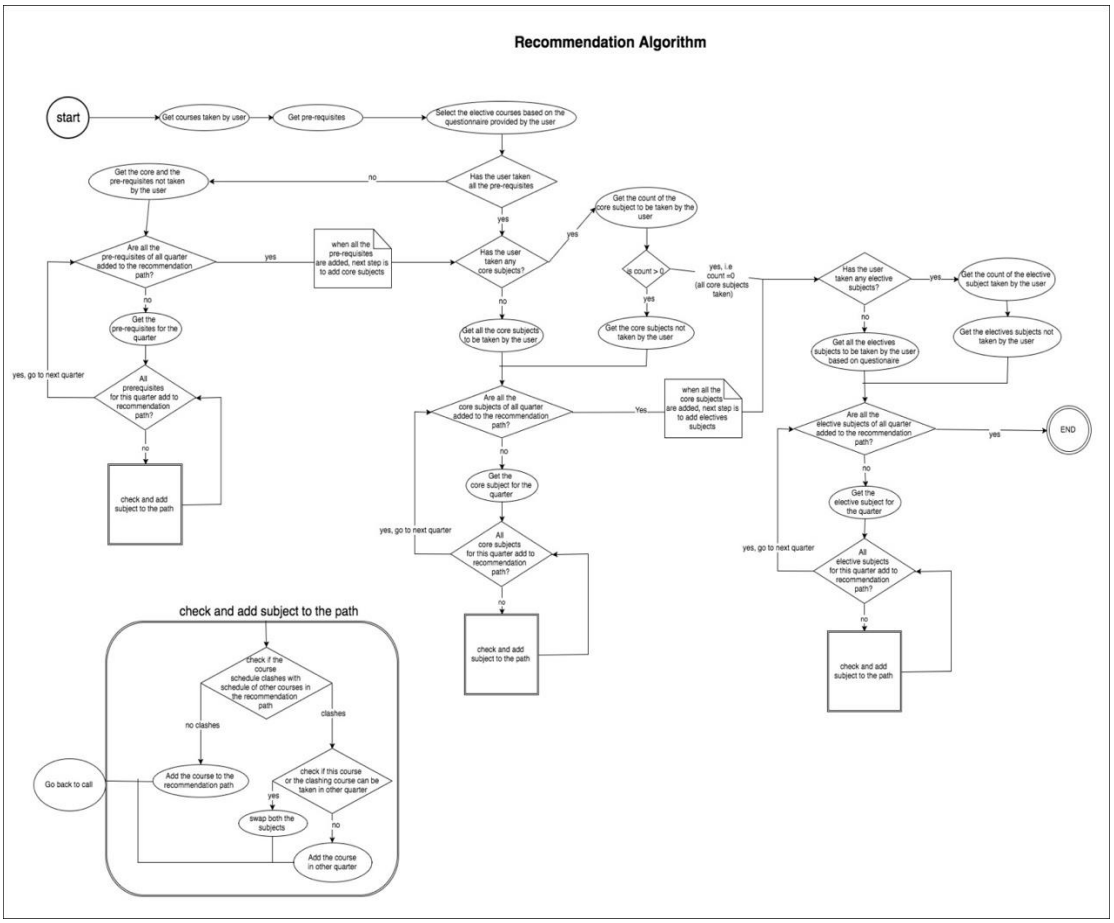

Figure 20. Recommendation Algorithm.

Step A. Add prerequisites

Step 1. Get the list of prerequisites not yet taken by querying the student_prerequisite table and course_taken table.

Step 2. Select one prerequisite (P1) from the list of prerequisites obtained in Step 1.

Step 3. Select the best quarter (Q1) (taken from the quarter_offered table in the database) where P1 can be added i.e. the quarter before the

quarter when its core is offered. Check if it satisfies the course count preference.

i. If does not satisfy, go to Step 3 to select different quarter for P1.

ii. If it satisfies, go to Step 4.

Step 4. Check if the schedule (taken from the course_schedule table in the database) for P1 clashes with any other course schedule (P2) that is already in the recommendation path for Q1.

a. If P1 clashes with P2 go to Step 6.

b. If no clash go to Step 5.

Step 5. Add P1 to the path.

i. If more prerequisites to be added to path go to Step 2.

ii. If all prerequisites are added to path then go to Step B.

Step 6. Check if P2 can be added to another quarter (Q2),

i. If yes, in the recommendation path add P2 to Q2 and go to step 5 to add P1 to Q1.

ii. If cannot be added, add P1 to next quarter when it is offered.

Step B. Add core courses

Step 1. Get all the core courses not yet taken by querying the core_course and the course_taken tables in the database.

Step 2. Select one course (C1) from the list of core subjects not yet taken obtained in Step 1.

Step 3. Select the best quarter(Q1) where C1 can be added i.e. immediate quarter Q1 when C1 is offered or immediate Q1 after the quarter when P is/will be taken.

Step 4. Check if the schedule (taken from the course_schedule table in the database) for C1 clashes with any other course schedule (C2) already in the recommendation path for Q1.

i. If C1 clashes with C2 go to Step 6

ii. If no clash go to Step 5

Step 5. Add C1 to path.

i. If more core courses to be added to path go to Step 2.

ii. If all core courses are added to path then go to Step C.

Step 6. Check if C2 can be added to another quarter (Q2),

i. If yes, in the recommendation path add C2 to Q2 and go to step 5 to add C1 to Q1.

ii. If no, add the C1 to the next quarter when it is offered.

Step C. Add elective courses

Electives are filter based on the preferences (degree option, lecture/ lecture and laboratory, day preferences, time preference, summer courses and independent study) submitted by the student.

Step 1. Get the total number of electives that is to be taken depending upon the degree option (project- 5, thesis-4, comprehensive exam-6) preference.

Step 2. Determine the remaining number of electives still to take and choose one elective (E1) from the list of elective courses offered according to the student preferences by querying the elective_course, course_taken, student_preference, day_preference and time_preference tables in the database.

Step 3. Select the best quarter (Q1) where E1 can be added.

Step 4. Check if the schedule for E1 clashes with any other course schedule (E2) already in the recommendation path for Q1.

   i. If E1 clashes with E2 go to Step 6

   ii. If no clash go to Step 5

Step 5. Add E1 to the path.

   i. If more elective courses to be added to path go to Step 2.

   ii. If all elective courses are added to path then go to Step C.

Step 6. Check if E2 can be added to another quarter (Q2),

   i. If yes, in the recommendation path add E2 to Q2 and go to step 5 to add E1 to Q1.

   ii. If no, add the E1 to next quarter when it is offered.

Step D. Add degree option i.e. CSE 690 for MS Project Option, CSE 699 for MS Thesis Option and CSE 689 for MS Comprehensive Exam Option.

Step 1. In the recommendation path, check for the quarter (Q) when the last core course is added.

Step 2. Add the degree option course in the quarter after Q i.e. if the last core course was taken in Winter 2015 quarter, degree option course is added in the Spring 2015 quarter.

CHAPTER FIVE

SYSTEM TESTING

The main aim of testing is to evaluate bugs and errors in the system. When checking for defects the normal behavior of all the source code is checked. Testing helps in understanding the state and performance of the system. Performance does not just depend on errors but when used on different platforms system might show strange behavior because of compatibility issues. This deep level introspection into system would provide code and resource optimization in some cases.

Testing can be done using different methods. The methods used to test this application are as follows.

- Unit Testing
- Module Testing
- Integration Testing
- Output Testing and
- User Acceptance Testing

Unit Testing

The logic and functionality of the system is measured by this testing. First units which are individual methods are created by identifying the smallest testable components. Unit test is performed only in initial test stages using a

white box test method. Unit testing helps in adapting to a changing system, in accelerating the development process and in developing a reliable code.

Module Testing

Like unit testing, module testing involves testing of system components. A module is defined as a combination of several units. Module testing can also be categorized as white box testing. The main objective of this testing is to find defects in specific modules. This testing is more about error detection than functionality testing. Testing multiple modules at the same time, makes it time effective for developers. All modules can be tested independently or incrementally.

Integration Testing

Integration testing is done to know defects that might occur during the merging of units. Unit tests may precede integration testing because this cannot be done without units. It helps in detecting the interface defects among system components. Integration testing has two approaches -- bottom-up and top-down methods. Under top-down higher-level modules are tested first before the lower levels. Under the bottom-up approach, the lower level components are tested first before the higher levels.

Output Testing

As the name suggests output testing depends on outputs of selected functions. In this testing, test cases which when executed will provide the known outcomes are developed. These test cases are then executed and will help in discovering the defects in the system if outcomes of execution are different from known outcomes. Sometimes, the generation of the test cases takes so much time, but the effort pays off to know the system performance. This helps the developer to know how the application will perform in a real-time environment.

User Acceptance Testing

This testing is very important in knowing how user interaction is done. The acceptance of the end user is the key aspect of software growth. As part of the testing, many users are chosen to play with the system and evaluate how user-friendly the system is. It is like a company releasing an alpha version of the system. This helps the company to capture user interests and thereby provide important updates to it. User acceptance testing is a very important part of software maintenance.

Testing Conclusion

Each testing mentioned above is performed at different stages of the application development. Unit tests help in understanding the algorithmic behavior. Validation testing is performed to understand if the system satisfies the

requirements specified by stakeholders. Integration testing helps in making the build process very effective. A wide number of test cases are used to find out if there are any defects in the system. Error handling is also made efficient. The application is given to some end users to get their feedback for further development. I acknowledge and appreciate all users that helped in testing the system, discovering bugs and suggesting ways to make the system more efficient.

Table 16. Test Modules

Testing Type	Used in Module
Unit Testing	1. Add prerequisite courses unit 2. Add core courses unit 3. Sort elective unit 4. Add elective unit 5. Add degree preference unit
Module Testing	Recommendation module
Integration Testing	Generate / update recommendation module

Users and Feedback

Table 17. Users and Feedback

User Name	Admission Status	User Comment/Feedback	Notes
Bhavana Narla	Female; Conditionally Admitted: Fall 2015	Recommendation path is generated correctly according to my preferences that I provided in the questionnaire.	Problem: User forgot the password so had to reset the password by "update" query. Fixed: Included the functionality to reset password.
Sindhu Hari	Female; Conditionally Admitted: Spring 2015	According to my actual current courses taken the recommendation path generated the courses accurately. Electives were selected considering some of my preferences.	Problem: User refreshed the browser and system broke. Fixed: the system broke because the data like Coyote ID and role which are

			used by the system was cleared on refresh. Cookies were then used to store Coyote ID and role of the user which is used by the system. The cookies are set at login time and cleared at logout time.

NOTE: E – elective course, C – core course, P – pre-requisite course

Case 1: The User has prerequisites and has not yet taken any courses.

This case considers a newly admitted student who has prerequisites and has not yet taken any courses. The path will be generated from the first quarter to the last quarter. At first, all the prerequisites are added to the path and depending on the quarter they are added, the respective core course is also added to the path. Last to be added will be the elective courses which will be selected based on the preferences indicated by the student in the questionnaire submitted. An example will be used to clearly illustrate how the recommendation path is generated following the steps A-D described below.

Table 18. Test Case 1 Information

Degree Option	Project
Start Quarter	Fall 2015
Prerequisites Needed	P1 – offered in Winter; -- required for core C1 P2 – offered in Winter, Spring; -- required for core C2
Electives Needed *Chosen from list of all possible electives and student's preferences*	E1 - offered in Fall, Spring - based on preferences: day preference: Monday, Tuesday, Wednesday and Thursday, time preference: Morning, Evening. E2 - offered in Winter - based on preferences: day preference: Monday, Tuesday, Wednesday and Thursday, time preference: Morning, Evening. E3 - offered in Spring

	- based on preferences: day preference: Monday, Tuesday, Wednesday and Thursday, time preference: Morning, Evening. E4 - offered in Fall, Winter, Spring, Summer - based on preferences: day preference: Monday, Tuesday, Wednesday and Thursday, time preference: Morning, Evening, independent study preference: True E5 - offered in Fall - based on preferences: day preference: Monday, Tuesday, Wednesday and Thursday, time preference: Morning, Evening.
Core Courses Needed *Uses the MSCS Yearly Course Offering to determine what quarters core course is taught and the Flowchart of*	C1 - offered in Fall, Spring; requires P1 C2 - offered in Fall; requires P2 C3 - offered in Winter; requires P3 C4 - offered in Winter; requires P4 C5 - offered in Spring; requires P5

Course Dependencies for the prerequisites required by the core course	
Project Option Course *Can only be taken after all the core courses have been taken, passed the oral exam and the project proposal is presented and approved*	CSE690

Step A: Add prerequisite

From the test case information above, the student needs two prerequisites – P1 and P2. Quarter(s) when prerequisites are offered is taken from CSUSB class listing and by querying course_schedule and quarter_offered tables. The prerequisites are randomly selected from the “need prerequisites” list.

P1 is randomly selected from the list of prerequisites needed. P1 is the prerequisite for core course C1. Therefore, P1 should be taken before C1. As P1 is offered in Winter quarter and C1 is offered in Fall and Spring quarter, the best

quarter to take P1 is Winter 2016 and C1 in Spring 2016. As P1 is the first course to be added in the recommendation path there will be no clashes and P1 is added to Winter 2016 quarter in the recommendation path. Next prerequisite is randomly selected from the list of prerequisites needed. P2 is the next prerequisite that is to be added to the path.

Since P2 is the prerequisite for core C2 P2 should be taken before C2. As P2 is offered in Winter and Spring quarters and C2 is offered in Fall quarter, the best quarter to take P2 is Winter 2016/ Spring 2016 and C2 in Fall 2016. For prerequisites that are offered in more than one quarter, the first quarter when the prerequisite is offered will be chosen. In this case, Winter 2016 schedule is selected for P2 and is checked against all the other course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so P2 is added to the recommendation path. All the prerequisites are added to the recommendation path. Go to Step B.

Step B: Add Core

The core courses are randomly selected from the “courses still needed” list. Information when core courses are offered is taken from the CSUSB class listing and by querying course_schedule and quarter_offered tables.

C1 core is randomly selected from the list of core courses to be taken. As P1 is added to the Winter 2016 quarter in the recommendation path, C1 is added in Spring 2016 quarter. In the current recommendation path, there are still no courses added to the Spring 2016 quarter, C1 is the first course to be added.

Therefore, there are no clashes and C1 is added to the path. Next core that is to be added is randomly selected from the list of courses to be taken, say C2.

C3 is offered in Winter quarter. Best quarter for C3 is Winter 2016. Winter 2016 schedule for C3 is checked with all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so C3 is added to the recommendation path. Next core, C4 is randomly selected from the list of courses to be taken.

C4 is offered in Winter quarter. Winter 2016 schedule for C4 is checked against all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so C4 is added to the path. Next core, C5 is randomly selected from the list of courses to be taken.

C5 is offered in Spring quarter. The best quarter to take C5 is Spring 2016. Spring 2016 schedule for C5 is checked against all the course schedules that are already in the recommendation path for Spring 2016 quarter. There are no clashes so C5 is added to the path. All the core courses have already been added, so go to Step C.

Step C: Add Electives

The degree option for this case is “Project” therefore, the student should take 5 electives. Electives are filtered based on the preference student submitted by the student. Preferences used were day preference, time preference, summer course preference, independent study preference.

E1 is randomly selected from the list of elective courses to be taken. E1 is offered in Fall and Spring. Fall 2015 schedule for is checked against all the course schedules that are already in the recommendation path for Fall 2015 quarter. There are no clashes so E1 is added to the path. Next elective, E2 is randomly selected from the list of courses to be taken.

E2 is offered in Winter. Winter 2016 schedule for E2 is checked against all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so E2 is added to the path. Next elective, E3 is randomly selected from the list of courses to be taken.

E3 is offered in Spring quarter. Spring 2016 schedule for E3 is checked with all the course schedules that are already in the recommendation path for Spring 2016 quarter. There are no clashes so E3 is added to the path. Next elective, E4 is randomly selected from the list of courses to be taken.

E4 is offered in Fall, Winter, Spring, Summer. Fall 2015 schedule for E4 is checked with all the course schedules that are already in the recommendation path for Fall 2015 quarter. There are no clashes so E4 is added to the path. Next elective, E5 is randomly selected from the list of courses to be taken.

E5 is offered in Fall so the Fall 2015 schedule for E4 is checked against all the course schedules that are already in the recommendation path for Fall 2015 quarter. As the start time of E5 is 6:00 PM and start time of C2 is also 6:00 PM, E5 schedule clashes with schedule for C2. Since C2 is a core course and is offered only in Fall it cannot be added to any other quarter. Therefore, the next

possible quarter (Fall 2016) schedule is checked against all courses schedules that are in Fall 2016 quarter in the recommendation path. There are no clashes so E5 is added to the path. All electives have already been added, so go to Step D.

Step D: Add degree option course

Fall 2016 quarter is the last quarter where the last core C2 is added. The degree option that the student selected is Project, therefore, course CSE690 is to be added to Winter 2017 quarter i.e. after C2 is taken.

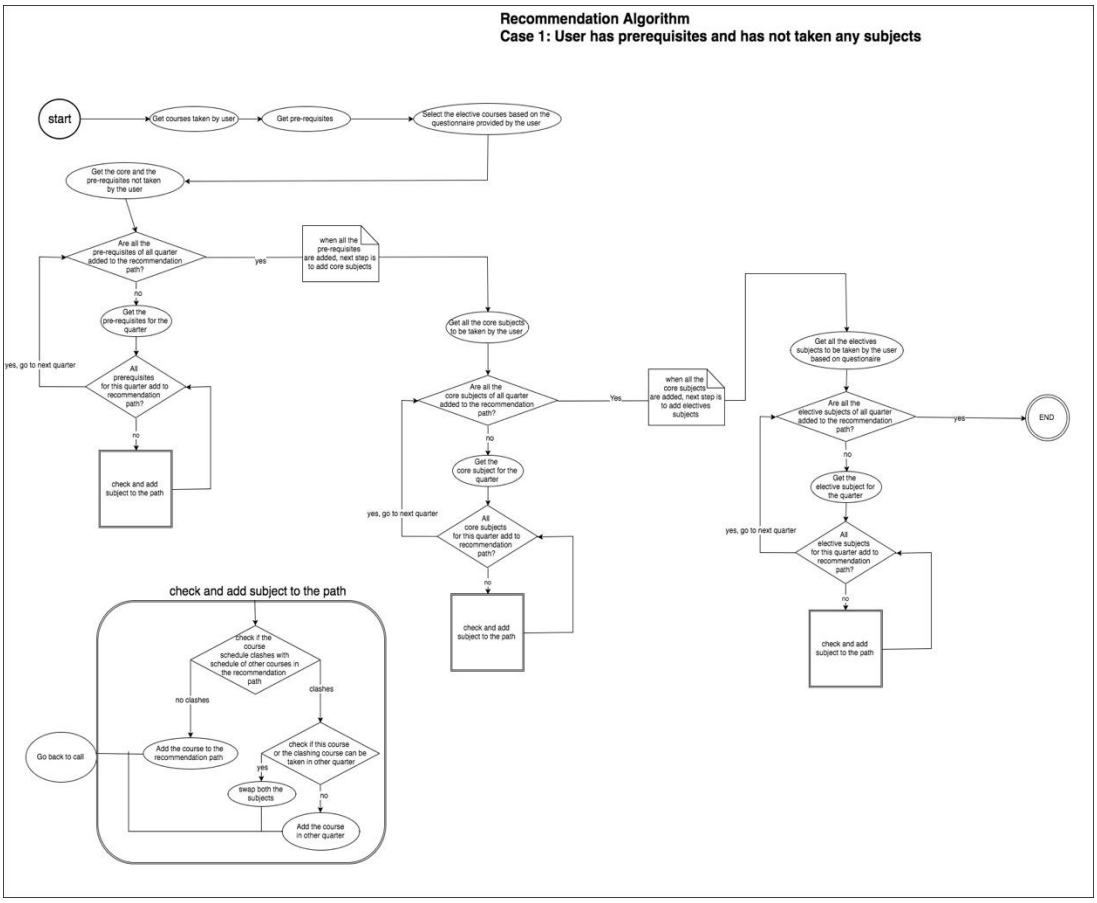

Figure 21. The User has Prerequisites and has not yet Taken any Courses.

Case 2: The user has prerequisites and has taken some courses.

This case considers a student who needs to take prerequisites and has taken some courses. The path will be generated from the current quarter until the last quarter. At first, all the prerequisites are added to the path and depending on the quarter in which it is added, the respective core course is also added to the path. Last will be elective courses which will be selected based on the preferences in the questionnaire submitted by the student. In this case, only those courses which are not yet taken by the student are added to the path. An example will be used to clearly illustrate how the recommendation path is generated following the steps A-D described below.

Table 19. Test Case 2 Information

Degree Option	Thesis
Start Quarter	Spring 2015
Courses Taken	E1 - taken in Spring 2015 E2 - taken in Spring 2015
Prerequisites Needed	P3 – offered in Winter; -- required for core C3
Electives Needed	E3 - offered in Winter, - based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday and Friday; time preference:

Chosen from list of all possible electives and student's preference	Morning, Evening; lecture Preference: Lecture and laboratory. E4 - offered in Fall, - based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday and Friday; time preference: Morning, Evening; lecture Preference: Lecture and laboratory.
Core Courses Needed *Uses the MSCS Yearly Course Offering to determine what quarters core course is taught and the Flowchart of Course Dependencies for the prerequisites required by the core course*	C1 - offered in Fall, Spring; requires P1 C2 - offered in Fall; requires P2 C3 - offered in Winter; requires P3 C4 - offered in Winter; requires P4 C5 - offered in Spring; requires P5
Thesis Option Course	CSE699

Can only be taken after all the core courses have been taken, thesis proposal is presented and approved	

Step A: Add prerequisite

From the test case information above, the student needs only one prerequisite – P3. Information on the quarter(s) when prerequisite is offered is taken from CSUSB class listing and by querying course_schedule and quarter_offered tables.

P3 is the prerequisite for core course C3. Therefore, P3 should be taken before C3. As P3 is offered in Winter quarter and C3 is also offered in Winter quarter, the best quarter to take P3 is Winter 2016 and C3 in Winter 2017. As P3 is the first course to be added in the recommendation path there will be no clashes and P3 is added to Winter 2016 quarter in the recommendation path. All the prerequisites have already been added to the path. Go to Step A.

Step B: Add Courses

C1 is offered in Fall and Spring quarters. Fall 2015 schedule for C1 is checked against all the course schedules that are already in the recommendation path for Fall 2015 quarter. There are no clashes so C1 is added to the path. Next core, C2 is randomly selected from the list of core courses to be taken.

C2 is offered in Fall quarter. Fall 2015 schedule for C2 is checked with all the course schedules that are already in the recommendation path for Fall 2015 quarter. There are no clashes so C2 is added to the path. Next core, C3 is randomly selected from the list of core courses to be taken.

C3 is offered in Winter quarter. P3 has been added to Winter 2016 quarter in the recommendation path, so Winter 2017 is the best quarter to take C3. Therefore, Winter 2017 schedule for C3 is checked against all the course schedules that are already in the recommendation path for Winter 2017 quarter. There are no clashes so C3 is added to the path. Next core, C4 is randomly selected from the list of courses to be taken.

C4 is offered in Winter quarter. Winter 2016 schedule for C4 is checked with all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so C4 is added to the path. Next core, C5 is randomly selected from the list of courses to be taken.

C5 is offered in Spring quarter. The best quarter for C5 is Spring 2016. Spring 2016 schedule for C5 is checked against all the course schedules that are already in the recommendation path for Spring 2016 quarter. There are no clashes so C5 is added to the path. Next, all core courses are added, go to Step C.

Step C: Add Electives

The degree option for this case is “Thesis” therefore, the student should take 4 electives. Electives are filtered based on the preference student submitted

by the student. As given in the test case information, the student has already taken 2 electives. Therefore, only 2 more electives are needed. E3 and E4 are the electives that are to be added to the recommendation path. First the electives are filtered depending on the day preference, if the filtered list(L1) count is more than 2(no. of electives needed to be taken) then electives in L1 are filtered depending on the time preference. Next if the filtered list(L2) count is more than 2(no. of electives needed to be taken) then electives in L2 are filtered depending on the lecture preference.

E3 is offered in Winter quarter. Winter 2016 schedule for E3 is checked against all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so E3 is added to the path. Next elective is randomly selected from the list of elective courses to be taken. Let's say E4 is the next elective course that is to be added to the path.

E4 is offered in Fall quarter. Fall 2015 schedule for E4 is checked with all the course schedules that are already in the recommendation path for Fall 2015 quarter. There are no clashes so E4 is added to the path. All electives are added so go to Step D.

Step D: Add degree option course

Winter 2017 quarter is the last quarter where the last core C3 is added. The degree option that the student selected is Thesis, therefore, course CSE 699 is to be added to Spring 2017 quarter i.e. after C3 is taken.

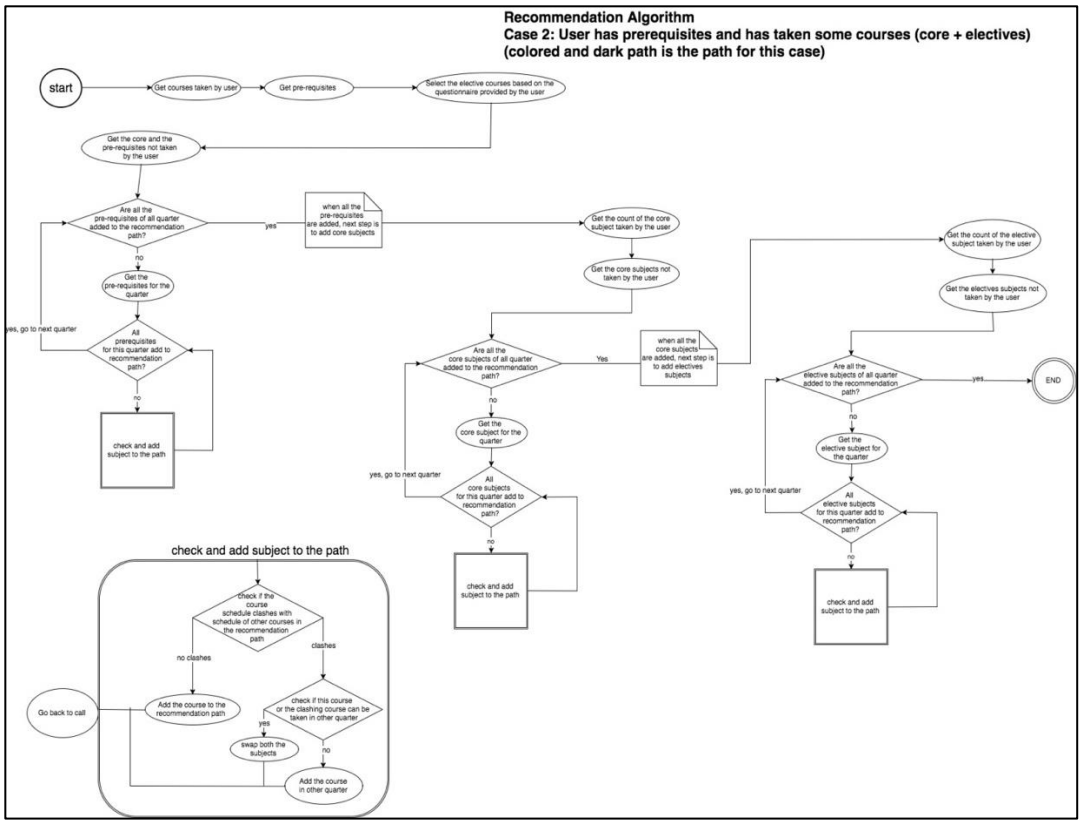

Figure 22. The User has Prerequisites and has Taken some Courses.

Case 3: The user has no prerequisites and has taken some courses.

This case considers all the students who have no prerequisites and have taken some courses. The path is then generated from current to last quarter. As the student has no prerequisites, core subjects that are not yet taken are added to the path. Last will be the elective subjects which will be selected based on the preferences questionnaire submitted by the student. In this case, only those courses are added to the path which are not yet taken by the student. An example will be used to clearly illustrate how the recommendation path is generated following the steps A-D described below.

Table 20. Test Case 3 Information

Degree Option	Thesis
Start Quarter	Spring 2015
Courses Taken	C1 - taken in Spring 2015; C5 - taken in Spring 2015; E1 - taken in Spring 2015; E2 - taken in Summer 2015;
Electives Needed *Chosen from list of all possible electives and student's preference*	E3 - offered in Fall, Spring; -based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday and Friday; time preference:

	Morning, Evening, Lecture Preference: Lecture and laboratory. E4 - offered in Winter; -based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday and Friday; time preference: Morning, Evening, Lecture Preference: Lecture and laboratory.
Core Courses Needed *Uses the MSCS Yearly Course Offering to determine what quarters core course is taught and the Flowchart of Course Dependencies for the prerequisites required by the core course*	C2 - offered in Fall; requires P2 C3 - offered in Winter; requires P3 C4 - offered in Winter; requires P4
Thesis Option Course	CSE699

Can only be taken after all the core courses have been taken, thesis proposal is presented and approved	

Step A: Add Prerequisites

Student has no prerequisites so go to Step B.

Step B: Add Courses

The core courses are randomly selected from the course needed list. Quarter(s) when core courses are offered is taken from CSUSB class listing and by querying course_schedule and quarter_offered tables.

C2 is randomly selected from the list of courses to be taken. C2 is offered in Fall quarter. As C2 is the first course to be added in the recommendation path it will not have any clashes. Therefore, C2 is added to the Fall 2015 quarter in the recommendation path. Next core is randomly selected from the list of core courses to be taken. Let's say C3 is the next core that is to be added to the path.

C3 is offered in Winter quarter. Best quarter for C3 is Winter 2016. Winter 2016 schedule for C3 is checked with all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so C3 is added to the recommendation path. Next core is randomly selected from the list

of core courses to be taken. Let's say C4 is the next core that is to be added to the path.

C4 is offered in Winter quarter. Winter 2016 schedule for C4 is checked with all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so C4 is added to the path. All the core subjects are added to the recommendation path. Go to Step C.

Step C: Add Elective

The degree option for this case is "Thesis" therefore, the student should take 4 electives. Electives are filtered based on the preference student submitted by the student. As given in the test case information, the student has already taken 2 electives. Therefore, he/she has to take 2 more electives. E3 and E4 are the electives that are to be added to the recommendation path. First the electives are filtered depending on the day preference, if the filtered list(L1) count is more than 2(no. of electives needed to be taken) then electives in L1 are filtered depending on the time preference. Next if the filtered list(L2) count is more than 2(no. of electives needed to be taken). Here L2 count is equal to 2 therefore electives in L2 will not be filtered according to lecture preference.

E3 is offered in Fall and Spring. Fall 2015 schedule for E3 is checked with all the course schedules that are already in the recommendation path for Fall 2015 quarter. There are no clashes so E3 is added to the path. Next elective is randomly selected from the list of courses to be taken. Let's say E4 is the next elective that is to be added to the path. Go to step 2.

E4 is offered in Winter. Winter 2016 schedule for E4 is checked with all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so E4 is added to the path. All the elective courses are added to the recommendation path. Go to Step D.

Step D: Add degree option course

Winter 2016 quarter is the last quarter where the last core C4 is added. The degree option that the student selected is Thesis, therefore, CSE 699 is to be added to Spring 2016 quarter i.e. after C4 is taken.

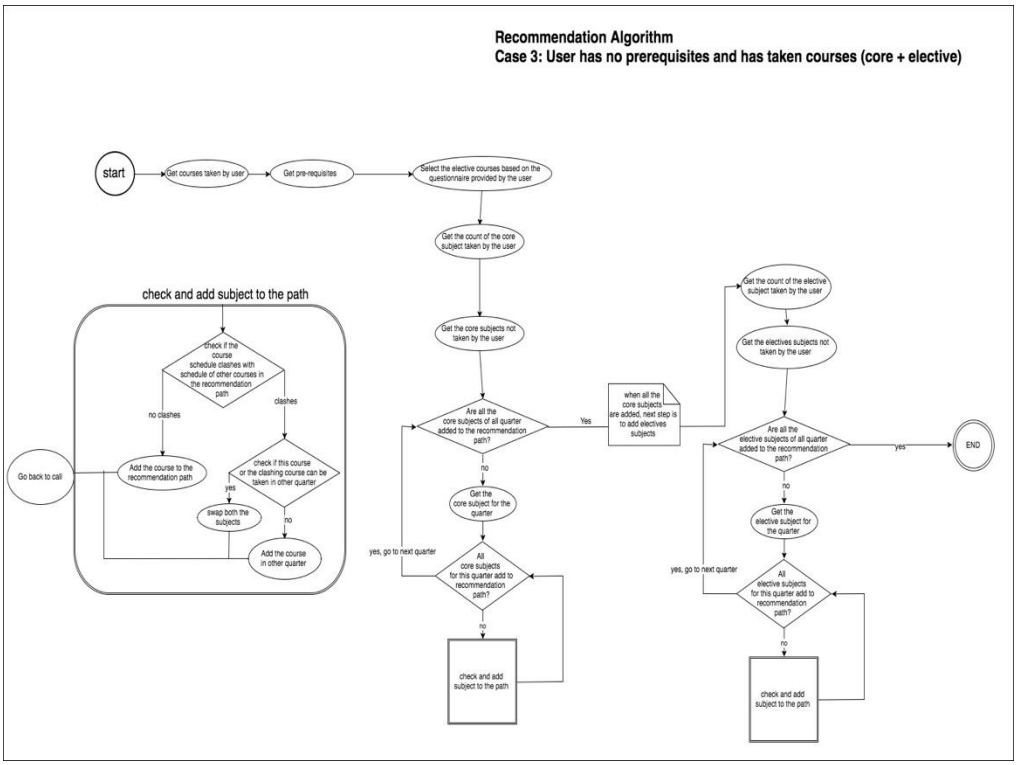

Figure 23. The User has no Prerequisites and has Taken some Courses.

Case 4: The user has no prerequisites and has not yet taken any courses.

This case considers all the newly enrolled students who have prerequisites and have not yet taken any courses. The path is then generated from first quarter to last quarter. As the student has no prerequisites, all the core subjects are added to the path and the depending on the student's preference electives are selected and added to the path. An example will be used to clearly illustrate how the recommendation path is generated following the steps A-D described below.

Table 21. Test Case 4 Information

Degree Option	Comprehensive Exam
Start Quarter	Fall 2015
Electives Needed *Chosen from list of all possible electives and student's preference*	E1 - offered in Winter; - based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday; time preference: Morning, Afternoon, Evening; lecture Preference: Lecture. E2 - offered in Fall, Spring; - based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday; time preference: Morning,

	Afternoon, Evening; lecture Preference: Lecture. E3 - offered in Winter; - based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday; time preference: Morning, Afternoon, Evening; lecture Preference: Lecture. E4 - offered in Winter; - based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday; time preference: Morning, Afternoon, Evening; lecture Preference: Lecture. E5 - offered in Fall; - based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday; time preference: Morning, Afternoon, Evening; lecture Preference: Lecture. E6 - offered in Fall, Winter, Spring, Summer;

	- based on preferences: day preference: Monday, Tuesday, Wednesday, Thursday; time preference: Morning, Afternoon, Evening; lecture Preference: Lecture; Independent Study Preference: True
Core Courses Needed *Uses the MSCS Yearly Course Offering to determine what quarters core course is taught and the Flowchart Of course Dependencies for the prerequisites required by the core course*	C1 - offered in Fall, spring; requires P1 C2 - offered in Fall; requires P2 C3 - offered in Winter; requires P3 C4 - offered in Winter; requires P4 C5 - offered in Spring; requires P5
Comprehensive Exam Option Course *Can only be taken after all the core courses have*	CSE699

been taken, pass a written examination on the material in the core courses.	

Step A: Add Prerequisites

Student has no prerequisites so go to Step B.

Step B: Add core courses

The core courses are randomly selected from the course needed list. Quarter(s) when core courses are offered is taken from CSUSB class listing and by querying course_schedule and quarter_offered tables.

C1 is randomly selected from the list of core courses to be taken. C1 is offered in Fall and Spring quarter. Best quarter to add C1 is Fall 2015. As of current recommendation path, there are still no courses added to the Winter 2016 quarter. C1 is the first course to be added. Therefore, there are no clashes and C1 is added to the path. Next core is randomly selected from the list of courses to be taken. Let's take C2 core to be added to the path.

C2 is offered in Winter quarter. Winter 2016 schedule for C2 is checked with all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so C2 is added to the path. Next core is randomly selected from the list of courses to be taken. Let's take C3 core to be added to the path.

C3 is offered in Winter quarter. Best quarter for C3 is Winter 2016. Winter 2016 schedule for C3 is checked with all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so C3 is added to the recommendation path. Next core is randomly selected from the list of courses to be taken. Let's take C4 core to be added to the path.

C4 is offered in Winter quarter. Winter 2016 schedule for C4 is checked with all the course schedules that are already in the recommendation path for Winter 2016 quarter. There are no clashes so C4 is added to the path. Next C5 core is to be added. Go to step 1.

C5 is offered in Spring quarter. Best quarter for C5 is Spring 2016. Spring 2016 schedule C5 is checked with all the course schedules that are already in the Spring 2016 quarter of recommendation path. There are no clashes so C5 is added to the path. Next, all core courses are added so go to Step C.

Step B: Add elective courses

The degree option for this case is "Comprehensive Exam" therefore, the student should take 6 electives. Electives are filtered based on the preference student submitted by the student. As given in the test case information, E1, E2, E3, E4, E5 and E6 are the electives that are to be added to the recommendation path. First the electives are filtered depending on the day preference. This filtered list contains 6 electives. Therefore, this will not be filtered anymore as the count of electives needed to be taken is 6.

As the core subjects were added to the path in the same way all the 6 electives (E1, E2, E3, E4, E5, E6) will be added to the recommendation path.

Step D: Add degree option course

Spring 2016 quarter is the last quarter where the last core C4 is added. The degree option that the student selected is Comprehensive exam, therefore, CSE689 is to be added to Fall 2016 quarter i.e. after C4 is taken.

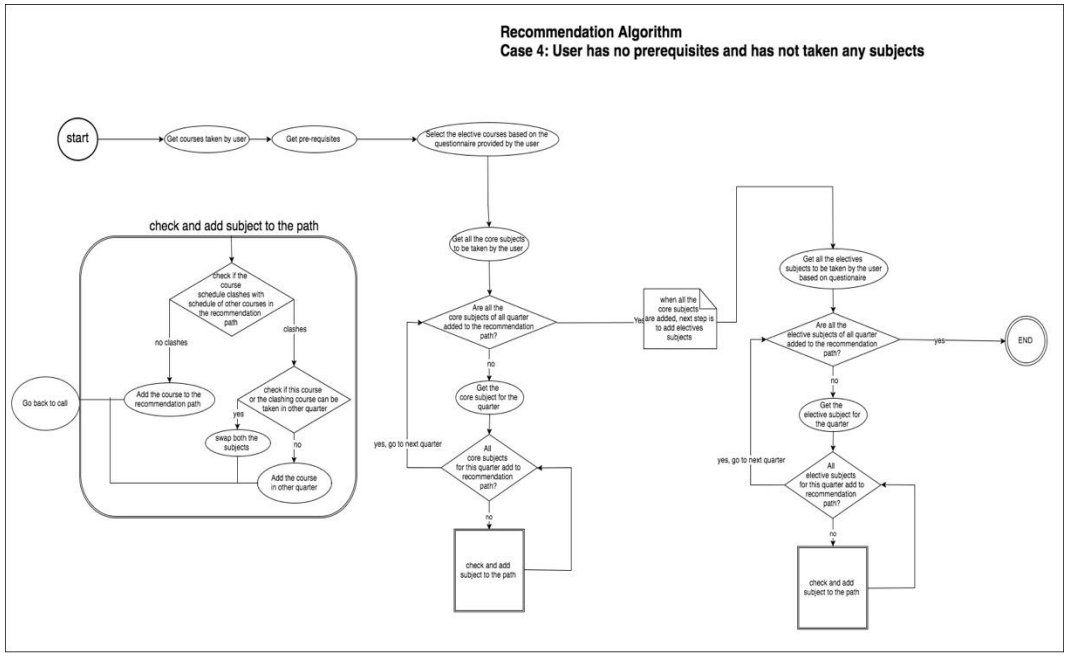

Figure 24. The User has no Prerequisites and has not yet Taken any Courses.

CHAPTER SIX

SYSTEM SCREENSHOTS

This project has been tested with different types of tests including unit-tests, module tests, integration tests, system tests and user acceptance tests and passed successfully with no defects encountered. Below are the screenshots that were taken during the testing process.

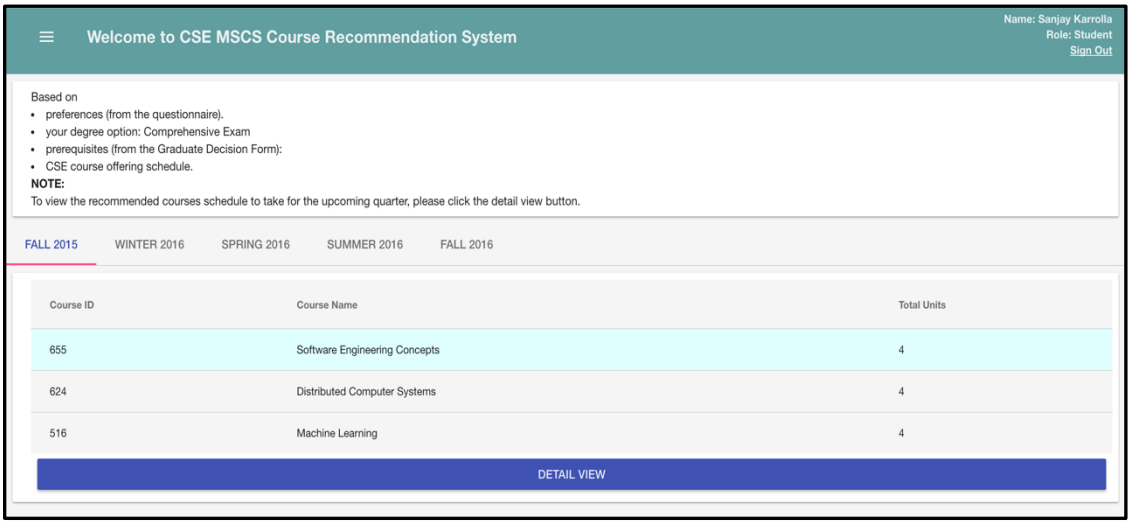

Figure 25. Recommendation Path for Student Role – Part 1.

Figure 25 displays the recommendation path for the student in a quarter and year sequence.

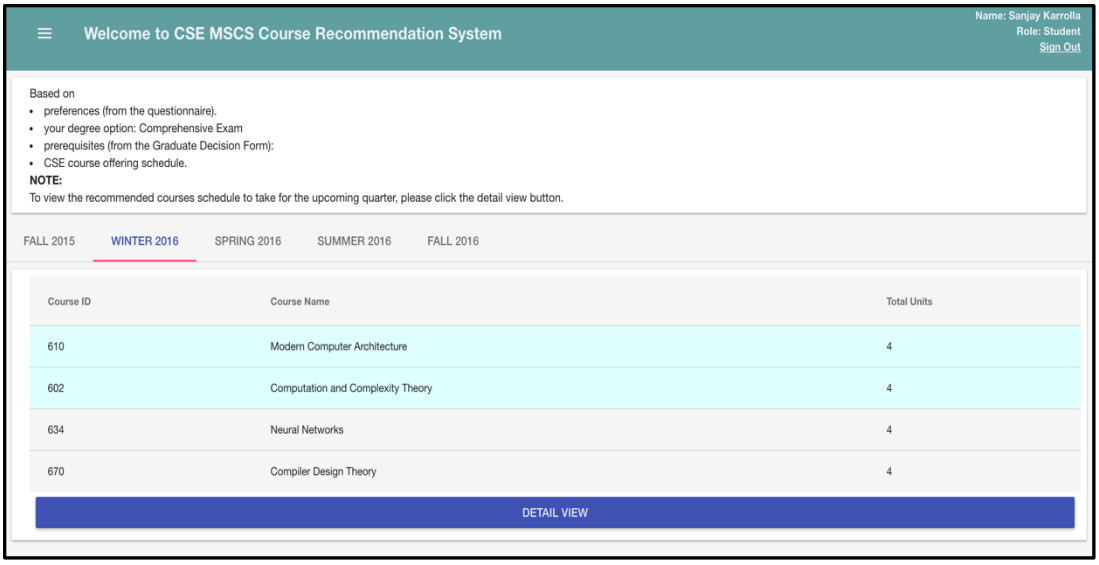

Figure 26. Recommendation Path for Student Role – Part 2.

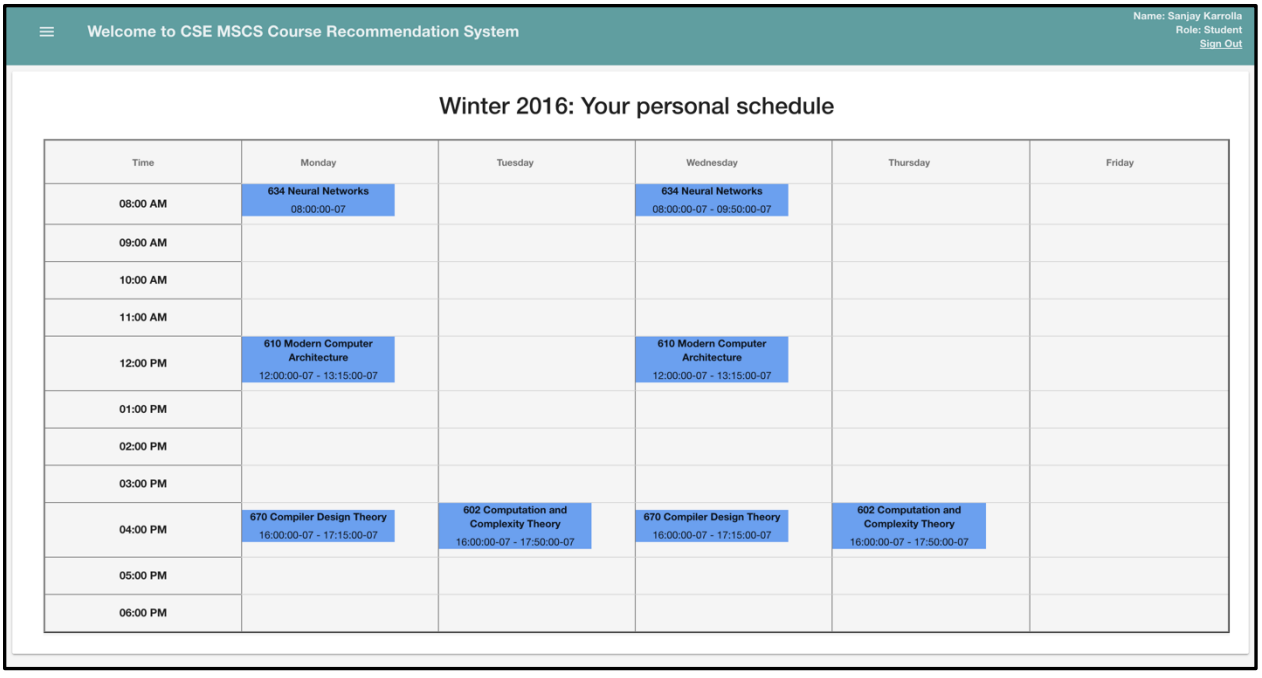

Figure 27. Detail View of Courses for a Quarter.

Figure 27 displays the schedule for a quarter the student will have if he/she takes the courses as generated by the recommendation algorithm.

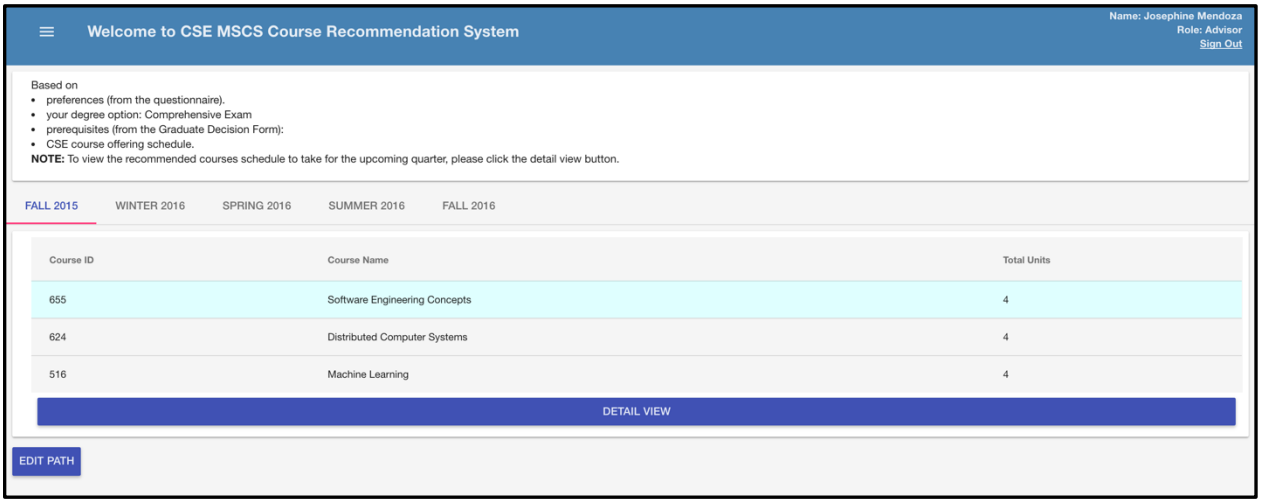

Figure 28. Recommendation Path for Advisor Roles.

Figure 28 displays the recommendation path page for a student as viewed by the advisor. The path is displayed by quarter and year sequence. When the “DETAIL VIEW” button is clicked, the advisor is redirected to the detail view page as shown in Figure 29 and when the “EDIT PATH” button is clicked, the application redirects the advisor to the edit recommendation path page as shown in Figure 30.

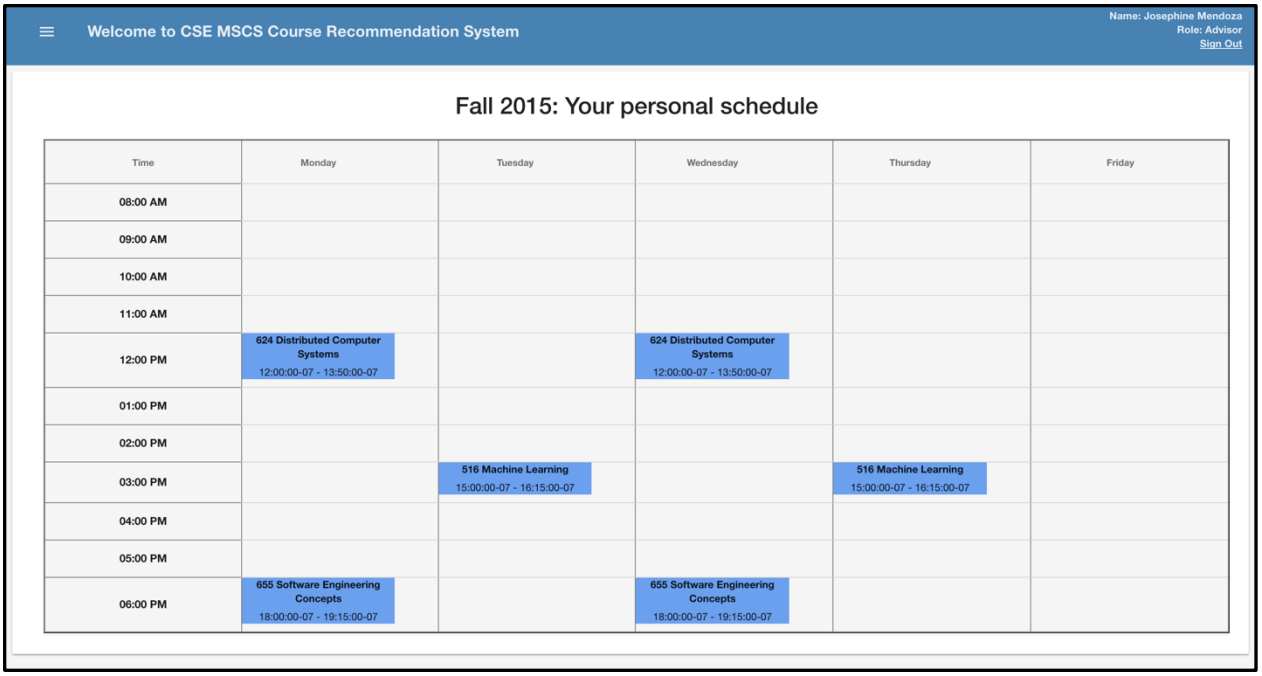

Figure 29. Detail View of Course for a Quarter.

Figure 29 displays the detail view of courses for a quarter. This page displays the schedule for a quarter the student will have if he/she takes the courses as recommended in the recommendation path.

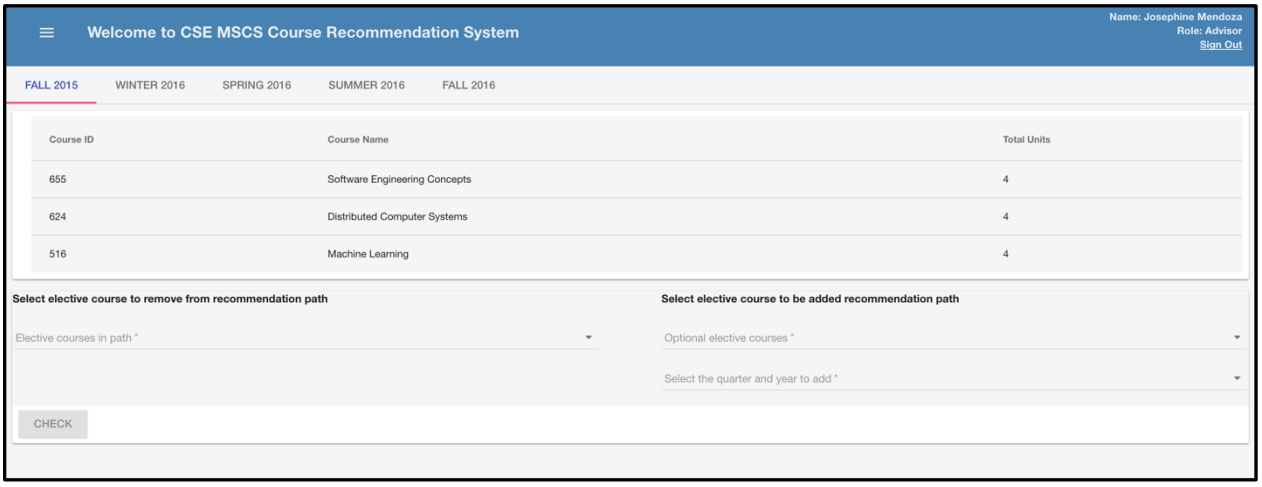

Figure 30. Edit Recommendation Path for Advisor Role.

Figure 30 displays the edit recommendation path page where an advisor can edit the path by selecting the 2 courses first, course that is to be removed from the recommendation path and second, new course that is to be added to recommendation path. Quarter and year in which the new course is to be added is also selected. Figure 31 shows edit recommendation path with all the information filled.

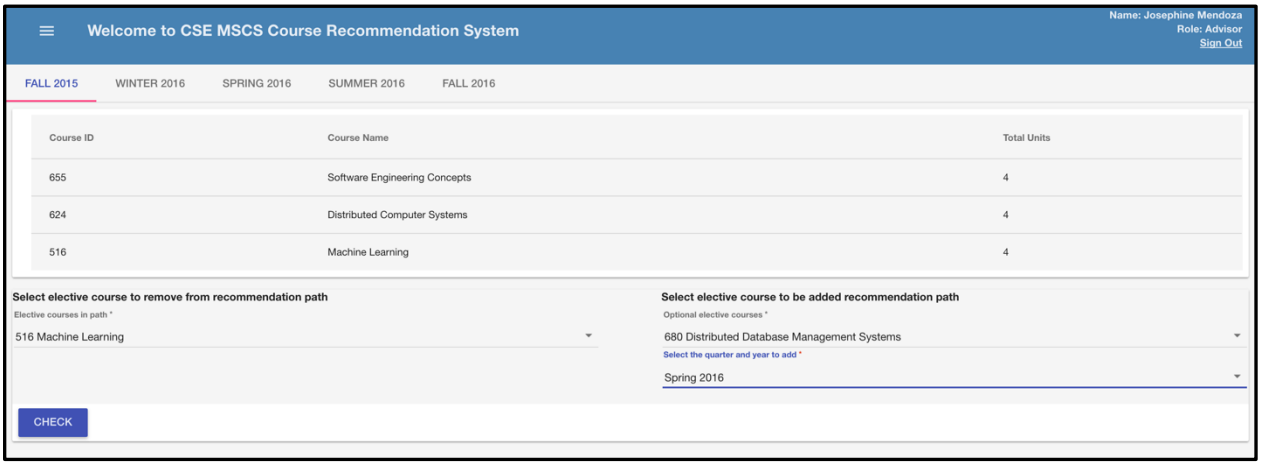

Figure 31. Edit Recommendation Path with Filled Information.

CHAPTER SEVEN

FUTURE ENHANCEMENTS

Audio on Low GPA

The future scope of this project can be expanded by integrating audio feature into the application. If a student has low GPA, then the system can issue a sound indicating a low GPA. As advisor will be reviewing a lot of student profiles, this feature will be useful for the advisor to immediately spot the low GPA.

Disallow Use of Old Passwords

When student changes the password, the system should know if the password was already used. For example, if the user changed the password P1 to P2 and again changes P2 back to P1 the system should throw an error indicating the user cannot use the same password that had already been used in the past.

In addition to disallowing the reuse of password, “age” password functionality can be added to the application. “age” password is adding an expiration period to the password. When the password expires the user is forced to change the password. This could be a part of a security measure to this application.

Reading Unofficial Transcripts

In the current project, courses taken by the student are manually added by the administrator to the database. A useful enhancement is to add a module for reading the student's unofficial transcript so the task of adding courses taken by the student can be automatically done.

CHAPTER EIGHT

CONCLUSION

In this graduate course recommendation system, a path of courses to be taken by the student is generated depending upon the courses taken, prerequisites not yet taken and the preferences that the student provided in the questionnaire form. The generated recommendation path can be used by students to plan the roadmap to graduation. In the current system, the advisor spends a lot of time in determining what courses to advise the student to take for the next quarter. This application automates the process by automatically generating the recommendation path. Thus, saving a lot of time and reducing anxiety and work pressure not only for the student but most importantly for the advisor.

REFERENCES

[1]. Angular (2010). AngularJS Introduction [Online].

Available: https://docs.angularjs.org/guide/introduction

[2]. Cryptr (2015). Node.js Cryptr [Online]

Available: https://www.npmjs.com/package/cryptr

[3]. Nightwatch.js (2015). Nightwatch.js Developer Guide [Online]

Available: http://nightwatchjs.org/guide

[4]. Node.js (2017). Node.js Guides [Online]

Available: https://nodejs.org/en/docs/guides

[5]. Nodemailer (2017). About Nodemailer [Online]

Available: https://nodemailer.com/about/

[6]. PostgreSQL (2017). PostgreSQL Developer [Online]

Available: https://www.postgresql.org/developer

[7]. Pg (2016). Node-Postgres [Online]

Available: https://www.npmjs.com/package/pg

[8]. Todd Motto. AngularJS Tutorial: A Comprehensive 10,000 Word

Guide, [Online]. Available: https://www.airpair.com/angularjs

[9]. Tutorials Point (2016). AngularJS - First Application. [Online].

Available: http://www.tutorialspoint.com/angularjs