WEB APPLICATION FOR GRADUATE COURSE ADVISING SYSTEM

Sanjay Karrolla
005291464@coyote.csusb.edu

Follow this and additional works at: http://scholarworks.lib.csusb.edu/etd

Part of the Other Computer Sciences Commons, Software Engineering Commons, and the Theory and Algorithms Commons

Recommended Citation

http://scholarworks.lib.csusb.edu/etd/606

This Project is brought to you for free and open access by the Office of Graduate Studies at CSUSB ScholarWorks. It has been accepted for inclusion in Electronic Theses, Projects, and Dissertations by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
WEB APPLICATION FOR GRADUATE COURSE
ADVISING 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
Sanjay Karrolla
December 2017
WEB APPLICATION FOR GRADUATE COURSE ADVISING SYSTEM

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

by
Sanjay Karrolla
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 recommendation system is to build a course recommendation path for students to help them plan courses to successfully graduate on time. The Model-View-Controller (MVC) architecture is used to isolate the user interface (UI) design from the business logic. The front-end of the application develops the UI using AngularJS. The front-end design is done by gathering the functionality system requirements -- input controls, navigational components, informational components and containers and usability testing. The back-end of the application involves setting up the database and server-side routing. Server-side routing is done using Express JS.
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.

I would also like to thank my parents for providing me with moral support and financial assistance throughout my Master's degree.
# TABLE OF CONTENTS

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

ACKNOWLEDGEMENTS ............................................................................................ iv  

LIST OF TABLES ........................................................................................................ viii  

LIST OF FIGURES ....................................................................................................... ix  

CHAPTER ONE: INTRODUCTION  
  Background ................................................................................................................... 1  
  Purpose ......................................................................................................................... 2  
  Existing System ........................................................................................................... 2  

CHAPTER TWO: SYSTEM ANALYSIS  
  Proposed System ........................................................................................................ 4  
  System Requirement Specifications ........................................................................... 5  
    Hardware Requirements ............................................................................................ 5  
    Software Requirements ............................................................................................ 5  

CHAPTER THREE: SELECTED SOFTWARE  
  MVC Architecture ...................................................................................................... 6  
    AngularJS Framework .............................................................................................. 7  
    Express.js ............................................................................................................... 7  
    PostgreSQL ............................................................................................................ 8  
  Package Definitions Used in the Project .................................................................. 9  
    angular-resource .................................................................................................... 9  
    angular-ui-router ..................................................................................................... 9  
    angular-material ..................................................................................................... 9
CHAPTER FOUR: SYSTEM DESIGN

Unified Modelling Language Diagrams .......................................................... 11
Use Case Diagrams ......................................................................................... 11
Sequence Diagrams ......................................................................................... 13
Data Flow Diagrams ....................................................................................... 18

CHAPTER FIVE: SYSTEM TESTING

Testing ............................................................................................................... 21
Unit Testing ....................................................................................................... 21
Validation Testing ............................................................................................ 22
Output Testing ................................................................................................. 22
Integration Testing ........................................................................................... 22
User Acceptance Testing .................................................................................. 23
Users and Their Feedback ................................................................................. 23

CHAPTER SIX: OUTPUT SCREENSHOTS

Welcome Page ................................................................................................. 25
Signup Page ...................................................................................................... 26
Validation for the Signup Page ......................................................................... 26
Questionnaire Page ............................................................................................ 35
Questionnaire Validation .................................................................................. 36
Login Page ......................................................................................................... 39
Login Validation ............................................................................................... 39
Student Homepage ............................................................................................ 42
Courses History Page ........................................................................................ 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendation Path Page</td>
<td>43</td>
</tr>
<tr>
<td>Review Questionnaire Page</td>
<td>45</td>
</tr>
<tr>
<td>Search and Modify User Information Page</td>
<td>47</td>
</tr>
<tr>
<td>Search User Validation</td>
<td>47</td>
</tr>
<tr>
<td>Edit User Validation</td>
<td>51</td>
</tr>
<tr>
<td>Create User Page</td>
<td>52</td>
</tr>
<tr>
<td>Create User Validation</td>
<td>52</td>
</tr>
<tr>
<td>Search and Modify Course Page</td>
<td>53</td>
</tr>
<tr>
<td>Search Course Validation</td>
<td>53</td>
</tr>
<tr>
<td>Edit Course Validation</td>
<td>57</td>
</tr>
<tr>
<td>Advisor Homepage</td>
<td>60</td>
</tr>
<tr>
<td>Course Details History Page</td>
<td>61</td>
</tr>
<tr>
<td>Add Prerequisites Page</td>
<td>62</td>
</tr>
<tr>
<td>View Recommendation Path Page</td>
<td>63</td>
</tr>
<tr>
<td><strong>CHAPTER SEVEN: FUTURE ENHANCEMENTS</strong></td>
<td></td>
</tr>
<tr>
<td>Grab Prerequisites from Graduate Decision Form</td>
<td>66</td>
</tr>
<tr>
<td>Password Strength</td>
<td>66</td>
</tr>
<tr>
<td>Display Format for Recommendation Path</td>
<td>66</td>
</tr>
<tr>
<td><strong>CHAPTER EIGHT: CONCLUSION</strong></td>
<td>67</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>68</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1. Usability Testing: Users and their Feedback ........................................ 23
Table 2. Signup Page Validation. ........................................................................ 27
Table 3. Questionnaire Validation...................................................................... 36
Table 4. Login Validation .................................................................................... 39
Table 5. Search User Validation ........................................................................ 47
Table 6. Search Course Validation ..................................................................... 53
Table 7. Edit Course Validation ......................................................................... 57
# LIST OF FIGURES

Figure 1. MVC Architecture. ................................................................. 6  
Figure 2. Angular Module and Controller Example. ................................. 7  
Figure 3. Router Module in Express.js .................................................. 8  
Figure 4. Administrator Use Case Diagram. .......................................... 12  
Figure 5. Advisor Use Case Diagram. .................................................... 12  
Figure 6. Student Use Case Diagram. ................................................... 12  
Figure 7. Student Role Sequence Diagram. .......................................... 13  
Figure 8. Advisor Sequence Diagram. .................................................. 14  
Figure 9. Administrator Update User Profile Sequence Diagram ............... 15  
Figure 10. Administrator Create Course and User Sequence Diagram ........ 16  
Figure 11. Administrator Update Course Sequence Diagram ..................... 17  
Figure 12. Data Flow Diagram for Login and Signup ............................. 18  
Figure 13. Data Flow Diagram for Student Role. .................................. 19  
Figure 14. Data Flow Diagram for Advisor Role. .................................. 19  
Figure 15. Data Flow Diagram for Administrator Role. ........................... 20  
Figure 16. Welcome Page. .................................................................. 25  
Figure 17. Signup Page. ..................................................................... 26  
Figure 18. First Name and Last Name Test Case: No Data Entered .......... 32  
Figure 19. Coyote ID Length Test Cases. ............................................. 32  
Figure 20. Empty and Correct Coyote ID Test Case. ............................... 32  
Figure 21. Incorrect Email Address Test Case: Missing Domain Name ....... 32
Figure 44. Search User Page. ................................................................. 47
Figure 45. Coyote ID Error Test Case. .................................................. 49
Figure 46. Coyote ID Empty and Correct Test Case. ............................. 49
Figure 47. Record not Found Test Case. ............................................... 49
Figure 48. Searched User Information is Displayed. ............................. 50
Figure 49. Details of the User Searched.................................................. 50
Figure 50. Edit User Page. ................................................................. 51
Figure 51. Delete User. ..................................................................... 51
Figure 52. Create User Page for Administrator Role. ............................ 52
Figure 53. Search Course Page. ........................................................... 53
Figure 54. Search Course Test Case. .................................................... 54
Figure 55. Course not Found Test Case. ................................................. 55
Figure 56. Searched Course is Displayed. ............................................. 55
Figure 57. Details of Course Searched.................................................... 56
Figure 58. Add Schedule Page............................................................ 56
Figure 59. Edit Course Test Case. ........................................................ 58
Figure 60. Edit Course Page. .............................................................. 58
Figure 61. Delete Course Schedule....................................................... 58
Figure 62. Create Course Page............................................................ 59
Figure 63. Search Student Page for Advisor. ....................................... 60
Figure 64. Searched Student is Displayed. .......................................... 60
Figure 65. Advisor Home Page.......................................................... 61
Figure 66. Course Details History Page View for Advisor Role. ......................... 61
Figure 67. Add Prerequisites Page View for Advisor Page............................. 62
Figure 68. Add Prerequisites Showing Prerequisites That Can be Added......... 62
Figure 69. Recommendation Path for Advisor Roles.................................. 63
Figure 70. Detail View of Course for a Quarter............................................. 64
Figure 71. Edit Recommendation Path View for Advisor Role...................... 64
Figure 72. Edit Recommendation Path with Filled information..................... 65
CHAPTER ONE
INTRODUCTION

Background

Universities around the globe share two ways of offering courses to students. One way is offering a fix set of courses throughout the program i.e. where every student takes the same courses, and the other way is providing students the liberty to choose courses to take. In this second case, the advisor plays an important role. Every quarter, each student meets with the advisor to plan what courses to register for the upcoming quarter. Advising is a time-consuming task for the advisor as it involves reviewing the academic history and ability of the student advisee. This process will differ for each student.

In these days, the administrative work is moved from hard copy (paper) files to digital files (word, excel, pdf, etc.). However, advising still requires one-on-one sessions between the advisor and student. During the advising period, there is a long queue of students in front of the advisor's office. This is the time where the advisor is under a tremendous pressure to finish advising within a tight schedule. At this point, most of the advisor's time is spent on repetitive tasks. This project recommends using current technologies in the advising process to save time and provide a more qualitative advising to students.
Purpose

This project aims on building an advising model that will facilitate graduate student success. This system helps the student to achieve his/her goals through a detailed road map that considers academic program requirements as well as the student's interests and schedule.

Student success is one of the core goals of the University's core goals in its strategic plan 2015-2020 document. “Objective 4 of this core goal 1 is to “Increase student success by maintaining high academic standards while reducing the overall DFWI (D, F, Withdrawal, Incomplete) rate through improved course learning conditions and enhance co-curricular support. Objective 5 aims to stay on track to meet or exceed CSU’s Graduation Initiative 2025 targets. Objective 6 is to foster the success of graduate students. A strategy that contributes to these objectives is to explore, develop and implement effective practices in graduate education that promote retention, graduation, end time to degree.”

Existing System

In the existing system, the CSE Graduate Coordinator for Advising (GCA) uses the following information sources 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 Writing Requirement Exemption Exam (WREE); the PeopleSoft (PS) 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 directed to choose the elective courses based on his/her day/time constraints due to work and personal reasons. This advising process is done 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 can be better utilized by communicating with the student about relevant issues regarding difficulties and challenges encountered in the program and counsel the student on career path.
CHAPTER TWO
SYSTEM ANALYSIS

Proposed System

This is a web application where a student can view the path of recommended courses he/she needs to get enrolled in to graduate successfully on time. Based on the student’s degree preferences, availability and schedule of offering of core courses and prerequisite courses, the application will validate the student’s status and provide the recommended schedule of courses. This application will also have the functionality for advisors to modify the recommended courses as needed.

User authentication is one of the important factors in this proposed system. All students will be authenticated based on their unique user identification number. This unique identification number (e.g. Coyote ID) is the number which is assigned by the university. The identification number along with a password set by user is used for the authentication.

First the student should answer a set of questions which provides the system with his/her preferences. After the questionnaire is answered, the student is directed to the student home page where a set of cards will be displayed. These cards will be to view the (1) student’s course history, (2) course recommendation path and to view/edit (3) the answers provided for the
questionnaire. Based on the questionnaire preferences the recommendation path is revised accordingly.

Since this questionnaire is key in determining the student preferences, it needs to be updated every quarter by the student. Thus, the system can come up with the best recommendation for that quarter.

System Requirement Specifications

This project will consist of a development phase and a production phase (prototype) that requires the following hardware and software requirements.

Hardware Requirements
- Laptop or PC running chrome browser for testing
- PC for development

Software Requirements
- Frontend/UI: AngularJS 1.6, HTML5,
- Backend: Express.js(Node.js) 4.15.2,
- Operating Systems: Windows 10
- Database: PostgreSQL 10.0
- IDE tools: IntelliJ WebStorm 171.4249.40
- Internet
CHAPTER THREE
SELECTED SOFTWARE

MVC Architecture

The system is built using MVC architecture. MVC is a software architecture pattern for implementing user interfaces on computers. It divides an application into three interconnected parts to separate internal representations of information from the ways information is presented to, and accepted from, the user.

Figure 1. MVC Architecture.

AngularJS (VIEW) framework is used to build the client side of application. Express.js (CONTROLLER) based on Node.js is used to build the server-side and PostgreSQL (MODEL) is used to store the server-side data.
AngularJS Framework

This powerful front-end technology is maintained by Google. AngularJS helps in building complex logic by providing simple components. AngularJS Framework is a library first developed by Misko Hevery in 2009 and mainly written in JavaScript.

```javascript
(function() {
    angular.module('AdvisorHome').controller('AdvisorHomeController', AdvisorHomeController);

    AdvisorHomeController.$inject = ['$stateParams', '$state', '$http', 'AuthService', 'appconfig', '$mdToast', 'UserService'];

    function AdvisorHomeController($stateParams, $state, $http, AuthService, appconfig, $mdToast, UserService) {
        var vm = this;
        vm.view = view;
        vm.history = history;
        vm.recommendation = recommendation;
        vm.coyote_id = $stateParams.coyote_id;
        console.log($stateParams, 'params');
        vm.role = $stateParams.Studentrole;
        UserService.getUser(vm.coyote_id, null).then(
            function success(response) {
                vm.user = response.data[0];
                console.log(vm.user);
            },
            function error() {
            }
        );

        function history() {
            $state.go('root.advisor.advisorhome.educationalhistory');
        }

        function recommendation() {
            $state.go('root.advisor.advisorhome.recommendationpath');
        }

        function view() {
            $state.go('root.advisor.advisorhome.addprerequisites');
        }
    }
})();
```

Figure 2. Angular Module and Controller Example.

Express.js

Express is a popular web framework written in JavaScript and hosted within the Node.js runtime environment. Node.js is an open-source, cross-platform runtime environment that allows developers to create all kinds of server-side files and application in JavaScript. The runtime used extends the browser
specified APIs and adds support for the core functional OS APIs like the HTTP and file system libraries.

Express is used for its great performance since by optimizing throughput & scalability in web apps; and spending less time with context shift across languages between browser and server code. It is portable with versions on most of the operating systems; and is well supported by many web host providers.

```javascript
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'));
});

module.exports = router;
```

Figure 3. Router Module in Express.js.

**PostgreSQL**

PostgreSQL is an object-relational database system that is used for this application. PostgreSQL is a cross-platform, open-source and relational database. It provides high performance, supports numerous datatypes, and strong support from the community as well as third-parties.
Package Definitions Used in the Project

These are the packages used in the development of this application.

**angular-resource**

Angular resource is a factory mostly used to create an angular service. A factory is used to create an object with some properties to return the same object. This is mainly used to interact with the RESTful server-side source of data. In this project, the objects were created using angular-resource to access the server-side code.

**angular-ui-router**

This is a routing framework used in the application. The main advantage of using ui-router is that it provides a state machine to manage transitions in application like transactions. In this project, routing in the application is done using angular-ui-router. UI router considers each URL as a state. So, if we want to go from one URL to another then the state change mechanism is used by angular-ui-router.

**angular-material**

This is a front-end UI framework which was developed based upon the Google material design. It provides reusable and accessible UI components. In this project, the user interface is inspired from the angular-material design. All the components i.e. buttons, input text, cards, etc. are styled using the angular-material library.
The material data table is material design table used to display the data in rows. There are many features to this data table such as pagination, sorting and filtering. All the tables used in the project user interface are created and styled using this library.
CHAPTER FOUR
SYSTEM DESIGN

Unified Modelling Language Diagrams

Use Case Diagrams

A use case diagram represents the behavior of a user and it describes a set of actions a user can perform on the application developed. Each use case(action) will convey different functionalities provided by the system. These use cases are essential for presenting how the system is valuable to the stakeholders.

The end-users of the system are represented as actors. A system can have many actors. But this system has three actors: administrator, advisor and student. These actors are called target audience for the application developed.

![Use Case Diagram]

- Admin
- Edit User
- Edit Course
- Create Course
- Create User
Figure 4. Administrator Use Case Diagram.

Figure 5. Advisor Use Case Diagram.

Figure 6. Student Use Case Diagram.
Sequence Diagrams

Sequence diagrams are used to find interactions among objects in a sequential manner. These diagrams display events based on time. They are useful in designing the system and to understand the system functionalities. Figure 7 represents the sequence diagrams for the student role.

Figure 7. Student Role Sequence Diagram.
When the user clicks on the view course history button, the server calls a method “get courses taken” which is redirected to the server-side query “get courses taken”. The server then sends the response back to the application. Similarly, when a user clicks the view recommendation path button the server calls a method “get user details” which in turn waits for “get user details” response. If there is no path saved to the profile then a new path will be created and saved in the database. Next is the review questionnaire card. Here a user can view the preferences he/she provided. He/she can change preferences and the path is dynamically updated.

Figure 8. Advisor Sequence Diagram.
Figure 8 shows the advisor sequence diagram. The advisor searches for a student profile and then views his/her profile. The advisor can add prerequisites, view education history and recommendation path. The education history presents the course history details of the student. When the advisor clicks on “add prerequisite” button, a list of prerequisites is presented, from which an advisor can select those required for the student. Then these prerequisites are posted to the student’s records. When the advisor clicks on “view recommendation path”, then the path of recommended courses for the student is presented.

Figure 9. Administrator Update User Profile Sequence Diagram.
Figure 9 shows the administrator update user profile sequence diagram. To update the user, profile the administrator first clicks on the search or update user button. The application then redirects to the search user page where the administrator enters the Coyote ID/name of the user. Based on the Coyote ID/name entered the details for the corresponding student are retrieved from the database and displayed. The administrator makes all the necessary changes and clicks on the update button to save the changes to the database. The server then updates the changes and sends back a success or a failure message that is then displayed to the administrator.

Figure 10. Administrator Create Course and User Sequence Diagram.
Figure 10 shows the create course and create user functions. The administrator uses these functions to create a new course or a new user. Both redirect to forms which when submitted, posts accordingly to the database. The server methods used are “create user” and “create course” queries. The response is either “success” or “failure”.

Figure 11. Administrator Update Course Sequence Diagram.
Figure 11 shows the last function an administrator can perform -- update course. The update course function modifies the course information like schedule, course details like course name, ID, has lab, course type (elective or core) and units. Clicking on the “search or update course” button, the administrator is directed to the appropriate page and enters the course ID or course name to search/update; edits the course schedule and clicks update. The update query runs on the server-side and information is modified.

Data Flow Diagrams

A data flow is a way to represent how data is flowing in the system. It helps in understanding the system at a deeper level. This mainly clarifies the structural orientation of the application. This diagram involves the data functions to be used. Figures 12 to 15 show the registration, student, administrator and advisor data flow diagrams.

Figure 12. Data Flow Diagram for Login and Signup.
Figure 13. Data Flow Diagram for Student Role.

Figure 14. Data Flow Diagram for Advisor Role.
Figure 15. Data Flow Diagram for Administrator Role.
This section provides an overview of methodologies used for verifying the functionality and behavior of different components in the project. Testing not only provides a way to investigate the quality of application built but indicates the status of the application components. Testing not only assists in fixing bugs but can help in measuring the performance and behavior of the application components.

There are five types of testing:

- Unit Testing
- Validation Testing
- Integration Testing
- Output Testing
- User Acceptance Testing

**Unit Testing**

As the name suggests, the whole system is tested unit-wise. The smallest component is called a single unit. Each component is checked if working according to the requirements. This testing is mainly done to know if something goes wrong while executing a unit. Every unit has some set functionality and if it does not respond as expected then the unit is defective. Unit testing makes it
easier for integration testing. In this project, each algorithm is unit tested. So, this testing facilitates error handling.

**Validation Testing**

Validation tests the build of the application developed. It is important for an application that is developed to satisfy the requirements of the end-user. There is no set time to do validation testing. It can be done at any stage of the product development. Validation helps the developer to understand the degree of efficiency in system design and to know if the application is robust.

**Output Testing**

Output testing is done after performing validation testing. Here, selected inputs are taken for which the desired outputs are known. The test is done by comparing the desired outputs and actual outputs of the application. Here, the behavior of application is also tested by updating selected inputs in different ranges. This will indicate how the system can handle the end-user traffic in real time.

**Integration Testing**

This testing is the next stage to the unit testing as all the units are integrated. After integrating the units, the application is checked if it is still behaving normally. If there are any problems during the combination of single units then these are flagged and the defects are evaluated and corrected. The procedure makes the build process effective.
User Acceptance Testing

This is the final testing that concludes whether the system is ready to be deployed. This testing includes how the system is accepted by the end-user. For this testing, the application is given to some targeted users of the application. The developer interacts with the end-users, considers their feedback and make appropriate adjustments to the application system.

Users and Their Feedback

Table 1. Usability Testing: Users and their Feedback

<table>
<thead>
<tr>
<th>User</th>
<th>User Comment/Feedback</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant 1</td>
<td>The system was simple. The recommendation path created was almost accurate as to what I have planned.</td>
<td>Problem: User entered more than 10 courses in questionnaire asking for number of courses user wants to take. (garbage input to database) Fixed: Validated field to restrict input to 1-4 range</td>
</tr>
<tr>
<td>Participant 2</td>
<td>Creating an account and using the system was a simple task.</td>
<td>Problem: keeps going back to homepage to access other cards. Fixed: Created a sidebar with links to other cards.</td>
</tr>
</tbody>
</table>
| Participant 3 | It gives entire information of which year and which quarter I should take the course considering my prerequisites, graduation preference, course day and course time preference. | **Problem:** The cancel button on signup page was not working.  
**Fixed:** Corrected the code and thus after clicking cancel, the user is redirected to welcome page. |
CHAPTER SIX
OUTPUT SCREENSHOTS

This project has been tested with different types of tests that includes unit, module, integration, system and user acceptance tests. Below are the screenshots presented by sequence of usage. Validation tests done on each page follow the screenshots.

Welcome Page

Figure 16. Welcome Page.

Figure 16 shows the welcome page of the application. First time users (only students) can use this application by clicking on signup and creating an account. The administrator and advisor get their login credentials (created by administrator) with a common password (first 7 letters of first name and first letter of last name combination) and then they can use forgot password to reset their password. Users already registered can use their Coyote ID and password (which were provided at signup) to login into the application.
Figure 17. Signup Page.

Figure 17 shows the signup page that every new user (students only) must first fill out to create an account before using the system.

Validation for the Signup Page

The table below shows which fields in the signup page are being checked for data input.
Table 2. Signup Page Validation.

<table>
<thead>
<tr>
<th>ID</th>
<th>Validation Description</th>
<th>Error Message</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1. | First name is a required field.  
  • If the field is empty  
  • If the field is not empty | Please enter your first name.  
  None | Cursor stays in the field and user enters first name.  
  Cursor moves to Last Name field. |
| 2. | Last name is a required field.  
  • If the field is empty  
  • If the field is not empty | Please enter your last name.  
  None | Cursor stays in the field and user enters last name.  
  Cursor moves to Coyote ID field. |
<table>
<thead>
<tr>
<th>Coyote ID field is ten numeric characters long.</th>
<th>The field is too short: must be ten numeric characters.</th>
<th>Cursor stays in the field and user enters the correct data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Less than 10 characters</td>
<td>The field is too long: must be ten numeric characters.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
</tr>
<tr>
<td>• More than 10 characters</td>
<td>None</td>
<td>Correct input; cursor goes to the Email field.</td>
</tr>
<tr>
<td>• Exactly 10 characters</td>
<td>Please enter your Coyote ID.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
</tr>
<tr>
<td>• Empty</td>
<td>None</td>
<td>Keyboard is disabled.</td>
</tr>
<tr>
<td>• Non-numeric characters</td>
<td>None</td>
<td>User must enter numbers only.</td>
</tr>
</tbody>
</table>
|   | Email address has valid format               | None                                      | Correct input; cursor goes to the password field  
|   | username@domain                             |                                           | Cursor stays in the field and user enters the correct data.  
|   | • No @ or no domain name                   | Please enter your valid email.            |                                           
| 5 | Password is a required field.              | None                                      | Cursor stays in the field and user enters details  
|   | • If the field is empty                    | Please enter your password.              | Cursor moves to Residency field.             
<p>|   | • If the field is not empty                | None                                      |<br />
| 6 | Phone number field is ten numeric characters long. |                                           |                                           |</p>
<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 10 characters</td>
<td>Please enter valid phone number: must be ten numeric characters.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
</tr>
<tr>
<td>More than 10 characters</td>
<td>Please enter valid phone number: must be ten numeric characters.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
</tr>
<tr>
<td>Exactly 10 characters</td>
<td>None</td>
<td>Correct input; cursor goes to the password field.</td>
</tr>
<tr>
<td>Empty</td>
<td>Please enter your phone number.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
</tr>
<tr>
<td>Non-numeric characters</td>
<td>None</td>
<td>Keyboard is disabled. User must enter numbers only.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Street Address, City, Zip code are required fields.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- If the field is empty: Please enter your address.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- If the field is not empty: None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cursor stays in the field and user enters address details.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cursor moves to Please review checkbox field.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Checkbox is a required field.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- If the checkbox is checked: None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- If the checkbox is not checked: None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CREATE ACCOUNT is disabled. User is given the chance to review what s/he has entered.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CREATE ACCOUNT is enabled. User clicks CREATE ACCOUNT to finish registration.</td>
<td></td>
</tr>
</tbody>
</table>
Figure 18. First Name and Last Name Test Case: No Data Entered.

Figure 19. Coyote ID Length Test Cases.

Figure 20. Empty and Correct Coyote ID Test Case.

Figure 21. Incorrect Email Address Test Case: Missing Domain Name.
Email *
007654567@coyote.csusb.edu

Figure 22. Correct Email Address Test Case.

Password *
....

Password *
Please enter your password.

Figure 23. Password Test Cases: Password Entered and No Password Entered.

Phone Number *
9089
The field is too short: must be ten numeric characters.

Phone Number *
9099999990989
The field is too long: must be ten numeric characters.

Figure 24. Phone Number Length Test Cases.

Phone Number *
6692646297

Figure 25. Correct Phone Number Test Case.
Figure 26. Correct Address Test Case.

Figure 27. Street Address Test Case: Not All Fields Have Data

Figure 28. Signup Checkbox Test Cases.
When the user clicks on “CREATE ACCOUNT” button in the signup page (See Figure 17.) the application redirects to this page. The student is required to answer the questionnaire because the generation of recommended courses to take is based on the preferences collected from this questionnaire.
### Questionnaire Validation

Table 3. Questionnaire Validation

<table>
<thead>
<tr>
<th>ID</th>
<th>Validation Description</th>
<th>Error Message</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Question 1-8 are required fields.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• a question is not answered.</td>
<td>This field is required.</td>
<td>Submit button is disabled. Cursor stays at the question and waits until user provides the answer.</td>
</tr>
<tr>
<td></td>
<td>• a question is answered.</td>
<td>None</td>
<td>Submit is enabled. Answers to the questionnaire are stored in the database.</td>
</tr>
</tbody>
</table>
Figure 30. Submit Button Enabled Test Case.
Figure 31. Submit Button is Disabled Test Case.
The administrator, advisor and students log into the application from the login screen as shown in Figure 32.

**Login Validation**

**Table 4. Login Validation**

<table>
<thead>
<tr>
<th>ID</th>
<th>Validation Description</th>
<th>Error Message</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Coyote ID field is ten numeric characters long.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td>Description</td>
<td>Correct Input</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>Less than 10 characters</td>
<td>The field is too short: must be ten numeric characters.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
<td></td>
</tr>
<tr>
<td>More than 10 characters</td>
<td>The field is too long: must be ten numeric characters.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
<td></td>
</tr>
<tr>
<td>Exactly 10 characters</td>
<td>None</td>
<td>Correct input; cursor goes to the Email field.</td>
<td></td>
</tr>
<tr>
<td>Empty</td>
<td>Please enter your Coyote ID.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
<td></td>
</tr>
<tr>
<td>Non-numeric characters</td>
<td>None</td>
<td>Keyboard is disabled. User must enter numbers only.</td>
<td></td>
</tr>
</tbody>
</table>

2. Password is a required field.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
<th>Correct Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Please enter your password</td>
<td>Login button is disabled</td>
</tr>
</tbody>
</table>
• Not empty | None | Login is enabled and displays the student homepage.

<table>
<thead>
<tr>
<th>Coyote ID *</th>
<th>Coyote ID *</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345678920</td>
<td>10</td>
</tr>
</tbody>
</table>
| The field is too long; must be ten numeric characters. | The field is too short; must be ten numeric characters.

Figure 33. Login Page Coyote ID Length Test Cases.

![Coyote ID Test Case](image)

Figure 34. Empty and Correct Coyote ID Test Case.

<table>
<thead>
<tr>
<th>Password *</th>
<th>7786126981</th>
</tr>
</thead>
<tbody>
<tr>
<td>Please enter your password.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 35. Password Test Case: Empty Password Field.
Student Homepage

![Student homepage screenshot](image)

**Figure 36. Student Home Page.**

Figure 36 shows the student home page with three cards: Courses History, Recommendation Path and Review Questionnaire.

Courses History Page

![Courses history screenshot](image)

**Figure 37. Course History for Student.**

Figure 37 displays the details of all the courses taken by the student.
Recommendation Path Page

Figure 38. Recommendation Path for Student Role – Part 1.

Figure 38 displays the recommendation path for the student that is displayed by quarter and year order.

Figure 39. Recommendation Path for Student Role – Part 2.

Figure 39 displays the recommendation path for the student that is displayed by quarter and year order.
Figure 40. Detail View of Courses for a Quarter.

Figure 40 shows the detail view of courses that are recommended to be taken by the student for the specified quarter.
Figure 41. Review Questionnaire Page.

Figure 41 shows the questionnaire when the student wants to review it. In this page, the user is given an opportunity to edit the questionnaire. If changes are done, the recommendation path will be updated to reflect the changes in the recommendation path.
Figure 42. Edit Questionnaire for Student Role.

Figure 42 shows the edit questionnaire for student to edit. When the student clicks on the “EDIT QUESTIONNAIRE” button as shown in Figure 41, the student is redirected to this page. Here the student can make changes with his/her preferences and click on “SUBMIT” button to save changes. If questionnaire is updated, the recommendation path will be re-generated. The validation tests performed when the questionnaire is edited are the same as those performed on the questionnaire when first filled out by a student after signup.
Figure 43. Administrator Home Page.

Figure 43 shows the administrator home page with four cards: Search and Modify User Information, Create New User, Search and Modify Course Information and Create New Course.

Search and Modify User Information Page

Figure 44. Search User Page.

Figure 44 shows the search user page where the administrator can either enter the Coyote ID or the name of the student to be searched.

Search User Validation

Table 5. Search User Validation

<table>
<thead>
<tr>
<th>ID</th>
<th>Validation Description</th>
<th>Error Message</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Coyote ID field is ten numeric characters long.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Case</td>
<td>Description</td>
<td>Action</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Less than 10 characters</td>
<td>The field is too short: must be ten numeric characters</td>
<td>Cursor stays in the field and user enters the correct data.</td>
<td></td>
</tr>
<tr>
<td>More than 10 characters</td>
<td>The field is too long: must be ten numeric characters</td>
<td>Cursor stays in the field and user enters the correct data.</td>
<td></td>
</tr>
<tr>
<td>Exactly 10 characters</td>
<td>None</td>
<td>Correct input; cursor goes to the password field.</td>
<td></td>
</tr>
<tr>
<td>Non-numeric characters</td>
<td>None</td>
<td>Keyboard is disabled.</td>
<td></td>
</tr>
<tr>
<td>Not empty but Coyote ID does not exist</td>
<td>No record found. Please enter valid Coyote ID or name.</td>
<td>Then user must enter numbers only.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cursor stays in the field and user enters the correct data.</td>
<td></td>
</tr>
</tbody>
</table>
- Not empty and Coyote ID exists.
- None
- Cursor moves to password field and user must enter the password.

<table>
<thead>
<tr>
<th>Coyote ID *</th>
<th>Coyote ID *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Please enter your coyote ID.</td>
<td>Please enter your coyote ID.</td>
</tr>
<tr>
<td>The field is too short: must be ten numeric characters.</td>
<td>The field is too long: must be ten numeric characters.</td>
</tr>
</tbody>
</table>

Figure 45. Coyote ID Error Test Case.

<table>
<thead>
<tr>
<th>Coyote ID *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Please enter your coyote ID.</td>
</tr>
<tr>
<td>7789/2081</td>
</tr>
</tbody>
</table>

Figure 46. Coyote ID Empty and Correct Test Case.

<table>
<thead>
<tr>
<th>User Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coyote ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

SEARCH

No record found. Please enter valid coyote ID or name.

Figure 47. Record not Found Test Case.
Figure 48. Searched User Information is Displayed.

Figure 48 shows the name, Coyote ID and the role of the target user searched for.

Figure 49. Details of the User Searched.

Figure 49 shows the details – name, Coyote ID, email, address, date of birth and role of the user searched when “VIEW” button is clicked. (See Figure 48) If the “EDIT” button is clicked, the application redirects the user to “edit user” page (See Figure 50). When “DELETE” button is clicked, a dialog box pops as shown in Figure 51.
Figure 50. Edit User Page.

Figure 50 displays “edit user” page where the administrator can make changes to any user information. Clicking on the “SUBMIT” button will save the changes made.

Figure 51. Delete User.

Edit User Validation

Refer to Table 2 for what validation is done on data input.
Figure 52. Create User Page for Administrator Role.

Figure 52 shows the “create user” page for the administrator role.

Administrator can only create two roles using create user form. They are administrator and advisor role. Here the administrator enters the information of the user and clicks on the “CREATE ACCOUNT” button.

Create User Validation

Refer Table 2 for what validation is done on data input.
Search and Modify Course Page

Figure 53 shows the search course page where the administrator enters the course ID or the course name to be searched.

Search Course Validation

Table 6. Search Course Validation

<table>
<thead>
<tr>
<th>ID</th>
<th>Validation Description</th>
<th>Error Message</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1. | Course ID field is three numeric characters long.  
  • Less than 3 characters | The field is too short: must be four numeric characters | Cursor stays in the field and user enters the correct data. |
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Error Message</th>
<th>User Interface Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>More than 3 characters</td>
<td>The field is too long: must be four numeric characters</td>
<td>Cursor stays in the field and user enters the correct data.</td>
</tr>
<tr>
<td>Non-numeric characters</td>
<td>None</td>
<td>Keyboard is disabled. Then user must enter numbers only.</td>
</tr>
<tr>
<td>Not empty but Course ID does not exist</td>
<td>Course not found.</td>
<td>Cursor stays in the field and user enters the correct data.</td>
</tr>
<tr>
<td>Not empty and valid Course ID</td>
<td>None</td>
<td>When user clicks search after entering course ID, course is displayed</td>
</tr>
</tbody>
</table>

Figure 54. Search Course Test Case.
Figure 55. Course not Found Test Case.

Figure 56. Searched Course is Displayed.

Figure 56 shows the course name and course ID for the target course being searched. When the “VIEW” button is clicked, details of the course (See Figure 57)– course name, course id, course type (core, elective), whether it has laboratory component, number of units and quarters taught. If a quarter is clicked, the screen will display the instructor assigned to teach the course and the day/time that the course is offered.
Figure 57. Details of Course Searched.

Figure 57 shows three buttons – EDIT, DELETE and ADD SCHEDULE.

Figure 58. Add Schedule Page.

When the “ADD SCHEDULE” button is clicked, the administrator enters the schedule details—quarter and year to offer the course, name of the instructor who will teach the course, day of the week, start time and end time when the course is to be offered and lab day and lab start time and end time if the course has a laboratory component. (See Figure 58) Clicking the “SUBMIT” button will add schedule to database.
## Edit Course Validation

Table 7. Edit Course Validation

<table>
<thead>
<tr>
<th>ID</th>
<th>Validation Description</th>
<th>Error Message</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1. | Year field is four numeric characters long.  
- Less than 4 characters  
- More than 4 characters  
- Non-numeric characters | The field is too short: must be four numeric characters  
The field is too long: must be four numeric characters  
None | Cursor stays in the field and user enters the correct data.  
Cursor stays in the field and user enters the correct data.  
Keyboard is disabled.  
Then user must enter numbers only. |
Figure 59. Edit Course Test Case.

Figure 60. Edit Course Page.

Figure 60 shows the edit course page where the administrator can make the changes to the course information. To save the changes, the “SUBMIT” button is clicked. If changes must be discarded, CANCEL button must be clicked!

Figure 61. Delete Course Schedule.
Figure 61 shows the delete course schedule pop-up box. When the user clicks on delete the schedule is completely removed from the database.

Figure 62 shows the create course page where the administrator enters the course information—Course id, course name, units, level (undergraduate, graduate), course department name that offers the course; day/year/quarter and start time/end time when the course is offered; lab day/start time/end time. Since a new course is being created, it is mandatory to enter at least one schedule detail for the course. For the course to be created, the “SUBMIT” button must be clicked.

![Create Course Page](image)

Figure 62. Create Course Page.

After filling the information, administrator clicks on SUBMIT and course is created successfully in database.
Advisor Homepage

Figure 63. Search Student Page for Advisor.

Figure 63 shows the search student page view for the advisor. After successful login, the advisor is first redirected to this page where the advisor enters the student Coyote ID or name and clicks on the “SEARCH” button. Figure 64 shows what is displayed.

Figure 64. Searched Student is Displayed.
Figure 65. Advisor Home Page.

Figure 65 shows the Advisor home page that has three cards: Course Details History, Add Prerequisites and View Recommendation Path. These cards display information about the user who is being searched. (see Figure 64)

Course Details History Page

Figure 66. Course Details History Page View for Advisor Role.

Figure 66 shows the details of all the courses taken and enrolled by the
target student.

Add Prerequisites Page

Figure 67. Add Prerequisites Page View for Advisor Page.

Figure 67 shows the add prerequisites page for advisor role. Here the advisor can view the prerequisites that are already added and can also add more prerequisites from the drop-down menu as shown in Figure 68.

Figure 68. Add Prerequisites Showing Prerequisites That Can be Added.
Figure 69. Recommendation Path for Advisor Roles.

Figure 69 shows the recommendation path page view for the advisor role. This page displays by quarter and year order, the courses that the student is recommended to take. When the “DETAIL VIEW” button is clicked, the application redirects the user to the detail view page as shown in Figure 70.

Figure 71 displays the edit recommendation path page when the “EDIT PATH” button is clicked.
Figure 70. Detail View of Course for a Quarter.

Figure 70 displays what courses a student needs to take when in a quarter as generated in the recommendation path.

Figure 71. Edit Recommendation Path View for Advisor Role.

Figure 71 shows the edit recommendation path as seen by the advisor role. The advisor can edit the path by selecting the subject that is either to be
removed or added and the quarter and year in which the subject is to be added.

Figure 71 shows edit recommendation path with all the information filled.

Figure 72. Edit Recommendation Path with Filled information.
CHAPTER SEVEN  
FUTURE ENHANCEMENTS  

Grab Prerequisites from Graduate Decision Form  
As of now, the prerequisites for a new student are manually entered into the database by database administrator by using the Graduate Decision Form. This work can be automated by making the system read the uploaded Graduate Decision Form and grab the prerequisites and store them into database directly.  

Password Strength  
This system currently does not determine the password strength. The password strength should be developed so it provides additional security to the application from brute force attacks.  

Display Format for Recommendation Path  
This system currently displays the recommendation path using tabs i.e. each quarter has its own separate tab. In this display format the user can view only courses only in one quarter at a time. The user needs to each tab separately to view courses in remaining quarters. To view courses in all quarter in page, the recommendation path should be displayed in tabular format.
CHAPTER EIGHT

CONCLUSION

This system could be a part of student course planning tools, as the main idea is to recommend the path of courses that student must take to graduate at the earliest time possible based on student degree option (comprehensive, thesis or project), student preferences (from a questionnaire form answered by student) and course availability (from course schedule). The path is from first quarter to expected graduation quarter for new students. For students who have already taken some courses, the recommendation path starts from the current quarter and ends with the expected graduation quarter.

There are three roles in the system- administrator, advisor and student. Administrator takes care of system tasks like creating and modifying users and courses. Advisor is responsible for viewing student profiles and their course details history, adding prerequisites and altering recommendation path as needed. Student can view his/her course history, recommendation path of courses and view/modify his/her preferences on the questionnaire.

In terms of results, the recommendation path dynamically changes as soon as there are any changes in the course schedule, prerequisites or preferences that the student selected in the questionnaire.
REFERENCES

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

   Available: https://material.angular.io/guide/getting-started

   Available: https://ui-router.github.io/guide/


   Available: http://nightwatchjs.org/guide

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

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


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