**Software Design**

**Document**

**for**

**WUMBO**

**Version 1.0 approved**

**Prepared by Roi Chico, Kristen Marenco, Luis Gonzalez, Francisco Gudino**

**Richard Cross**

**11/12/2017**

Table of Contents...................................................................................................................2

Revision History.....................................................................................................................5

1. Introduction................................................................................................................ 6
   1. Purpose........................................................................................................... 6
   2. Document Conventions……………………................................................. 6
   3. Intended Audience and Reading Suggestions................................................ 6
   4. System Overview........................................................................................... 6
2. Design Considerations............................................................................................... 7
   1. Assumptions and dependencies..................................................................... 7
   2. General Constraints........................................................................................7 Goals and Guidelines......................................................................................7
   3. Development Methods................................................................................... 7
3. Architectural Strategies..............................................................................................8
4. System Architecture................................................................................................... 9
   1. ....................................................................................................................... 9
   2. ....................................................................................................................... 10
5. Policies and Tactics.................................................................................................... 11
   1. Specific Products Used.................................................................................. 11
   2. Requirements traceability...............................................................................11
   3. Testing the software....................................................................................... 11
   4. Engineering trade-offs....................................................................................11
   5. Guidelines and conventions........................................................................... 11
   6. Protocols.........................................................................................................11
   7. Maintaining the software............................................................................... 11
   8. Interfaces........................................................................................................ 11
   9. System's deliverables..................................................................................... 11
   10. Abstraction..................................................................................................... 11
6. Detailed System Design............................................................................................. 13

6.1 Admin…............................................................................................ 13

6.1.1 Responsibilities...................................................................................... 13

6.1.2 Constraints..............................................................................................13

6.1.3 Composition........................................................................................... 13

6.1.4 Uses/Interactions.................................................................................... 13

6.1.5 Resources............................................................................................... 13

6.1.6 Interface/Exports.................................................................................... 13

6.2 Roadmap…............................................................................................ 13

6.2.1 Responsibilities...................................................................................... 13

6.2.2 Constraints..............................................................................................13

6.2.3 Composition........................................................................................... 14

6.2.4 Uses/Interactions.................................................................................... 14

6.2.5 Resources............................................................................................... 14

6.2.6 Interface/Exports.................................................................................... 14

1. Detailed Lower level Component Design

7.1 Login Controller..................................................................................... 15

7.1.1 Classification.......................................................................................... 15

7.1.2 Processing Narrative(PSPEC)................................................................ 15

7.1.3 Interface Description.............................................................................. 15

7.1.4 Processing Detail.................................................................................... 15

7.1.4.1 Design Class Hierarchy................................................................... 15

7.1.4.2 Restrictions/Limitations.................................................................. 15

7.1.4.3 Performance Issues........................................................................ 15

7.1.4.4 Design Constraints......................................................................... 15

7.1.4.5 Processing Detail For Each Operation........................................... 15

7.2 Registration Controller.................................................................................... 15

7.2.1 Classification.......................................................................................... 15

7.2.2 Processing Narrative(PSPEC)................................................................ 15

7.2.3 Interface Description.............................................................................. 16

7.2.4 Processing Detail.................................................................................... 16

7.2.4.1 Design Class Hierarchy................................................................... 16

7.2.4.2 Restrictions/Limitations.................................................................. 16

7.2.4.3 Performance Issues........................................................................ 16

7.2.4.4 Design Constraints......................................................................... 16

7.2.4.5 Processing Detail For Each Operation........................................... 16

7.3 Home Controller...................................................................................... 16

7.3.1 Classification.......................................................................................... 16

7.3.2 Processing Narrative(PSPEC)................................................................ 16

7.3.3 Interface Description.............................................................................. 16

7.3.4 Processing Detail.................................................................................... 16

7.3.4.1 Design Class Hierarchy................................................................... 17

7.3.4.2 Restrictions/Limitations.................................................................. 17

7.3.4.3 Performance Issues........................................................................ 17

7.3.4.4 Design Constraints......................................................................... 17

7.3.4.5 Processing Detail For Each Operation........................................... 17

7.4 Admin Home Controller................................................................................. 17

7.4.1 Classification.......................................................................................... 17

7.4.2 Processing Narrative(PSPEC)................................................................ 17

7.4.3 Interface Description.............................................................................. 17

7.4.4 Processing Detail.................................................................................... 17

7.4.4.1 Design Class Hierarchy................................................................... 17

7.4.4.2 Restrictions/Limitations.................................................................. 17

7.4.4.3 Performance Issues........................................................................ 18

7.4.4.4 Design Constraints......................................................................... 18

7.4.4.5 Processing Detail For Each Operation........................................... 18

7.5 Admin Course Controller................................................................................ 18

7.5.1 Classification.......................................................................................... 18

7.5.2 Processing Narrative(PSPEC)................................................................ 18

7.5.3 Interface Description.............................................................................. 18

7.5.4 Processing Detail.................................................................................... 18

7.5.4.1 Design Class Hierarchy................................................................... 18

7.5.4.2 Restrictions/Limitations.................................................................. 18

7.5.4.3 Performance Issues........................................................................ 18

7.5.4.4 Design Constraints......................................................................... 18

7.5.4.5 Processing Detail For Each Operation........................................... 18

1. Database Design.........................................................................................................20
2. User Interface.............................................................................................................21
   1. Overview of User Interface.......................................................................... 21
   2. Screen Frameworks or Images..................................................................... 21
   3. User Interface Flow Model........................................................................... 22
3. Requirements Validation and Verification.................................................................23
4. Glossary..................................................................................................................... 27
5. References................................................................................................................. 28

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1. Introduction**

**1.1 Purpose**

This document will start by giving the users an overview of what the software is intended to do and give a detailed list for developers of each function that needs to be incorporated.

**1.2 Document Conventions**

None

**1.3 Intended Audience and Reading Suggestions**

This documentation is intended for developers and users. The users should focus on sections 1 through 3 to have a description of what the project is and what it does. The developers should read the rest which contains the detailed information on how to create the project.

**1.4 System Overview**

The desired design approach was a simple way for CS students to access an automated roadmap. The administrator also had a simple way of manipulating data. The student registers and logs into their account. Then they only view the roadmap, personal information, or course information. The administrator has access to student information and courses. He can change course information such as the coordinator and the semester the course will be offered in. This will automatically affect the students’ roadmap.**2. Design Considerations**

**2.1 Assumptions and Dependencies**

* Assumes that the web server is Tomcat and the relational database is SQL
* Assumes that all web browsers are compatible
* Assumes user either an admin or CS student from the school
* Assumes the students will pass all their classes
* Assumes the students who are signing up are incoming freshmen

**2.2 General Constraints**

* Time Constraints
  + Instead of asking the user which classes they have taken. The software will assume the student’s information is given by the school.
* Testing
  + To verify the Admin is changing courses, we must display all courses in the database.

**2.3 Goals and Guidelines**

* The whole design of both the student and admin users strictly follows the KISS principle.
* The software must be delivered before the end of the cs3337 class.
* Speed was favored over memory because of the many dynamic variables used.
* The product was meant to be similar to the roadmap on GET, but much more user friendly to access and use.

**2.4 Development Methods**

The development method we used for our project was the Waterfall Development method. This method was applied to our project because the way we started our project was by gathering and documenting requirements. Once we received our requirements, we began to design the front end and back end for both the admin and student page. Code and unit test came after we agreed on which design pattern we were going to use for our project. During our coding implementation, we performed routine testing to make sure our project would run as designed and compiled properly as well as performing user acceptance testing. After debugging and fixing any issues that occurred, we compiled and uploaded our finished project to the cs3 server.

**3. Architectural Strategies**

**3.1 Programming Language Decision**

We chose to use sql prepared statements in Java servlets versus using PHP. This decision was made because we were more familiar with coding in servlets rather than with PHP syntax. Although, it is arguable that PHP is simpler and specifically made to work with databases we went with a more familiar territory as it would cut down the time needed to learn a new language.

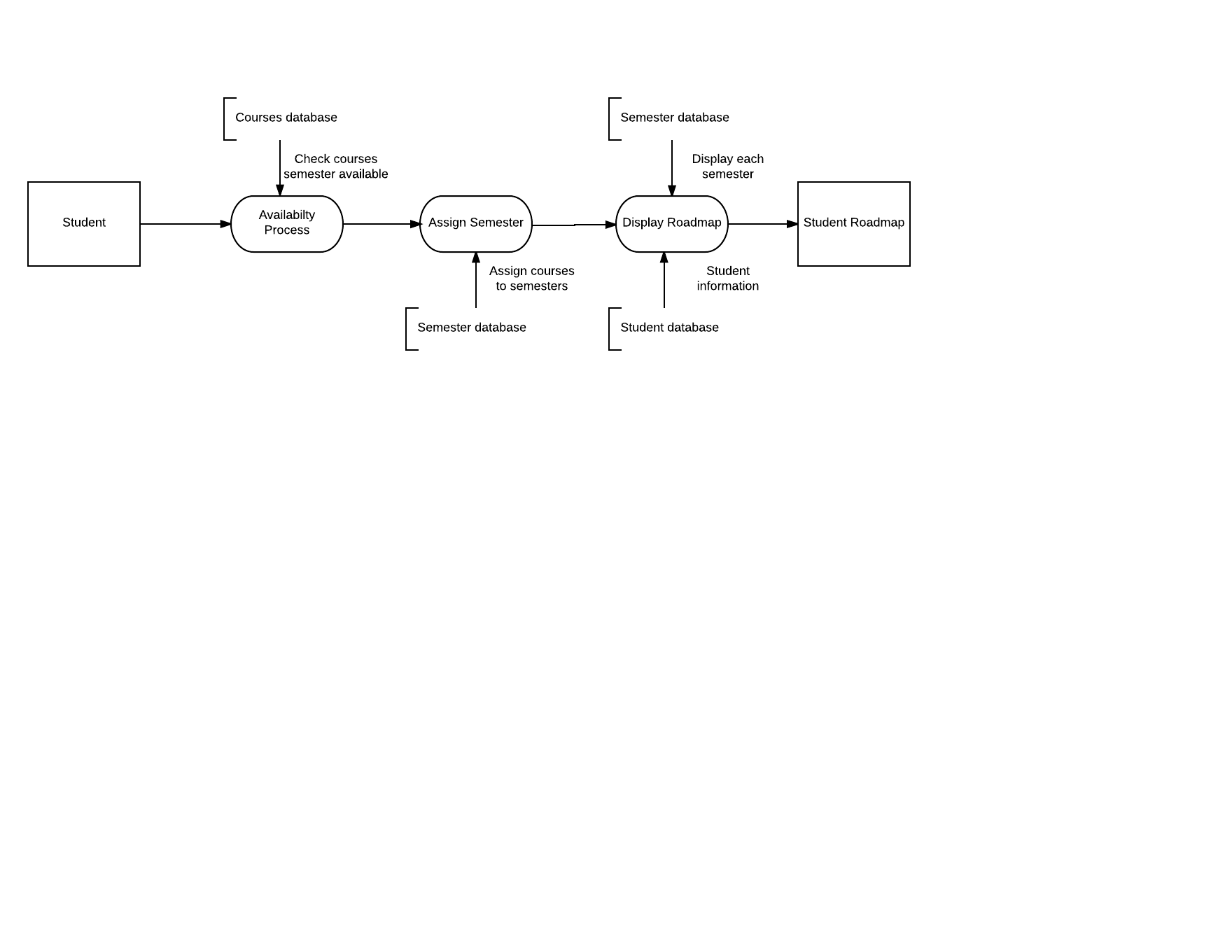
**3.2 Future Plans**

In order to stray away from relying on a school to provide a student’s information, in the future we plan to introduce the ability to read a student’s transcript to determine what classes they have already taken. Once having done that we will generate the roadmap having the completed classes in the corresponding semesters the student completed them in.

**3.3 Roadmap Design**

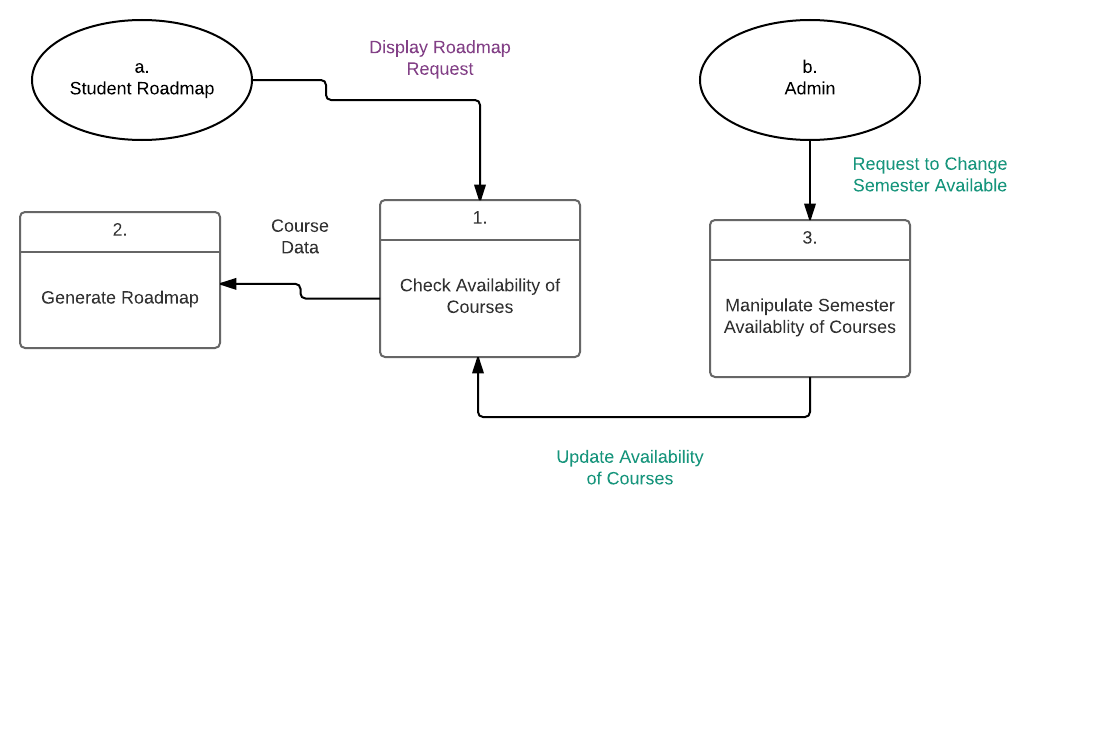
Initially we planned to have a profile creator where we would list all of the courses needed to attain a Bachelors in Computer Science, then prompt the user to select which courses they had completed and from there we planned to create the roadmap. We decided against this design because it would have taken more time and we would have had to use Javascript or create another Webpage with a form to get the information we needed. Ultimately, we chose to simulate the school providing a student’s information in order for us to generate the roadmap. The student’s information being what classes they had taken already.

**4. System Architecture**



**4.1 Student to Student Roadmap DFD**

We begin at the Student who wants to access their personalized roadmap. This prompts the “Availability Process” which looks into the Courses database to check what semester the courses are available. Then, we run the “Assign Semester” process that looks into the Semester database and assigns the courses the student must take to a Semester in the student’s personalized roadmap. Then, we run the “Display Roadmap” process which looks in the Semester database and displays each row containing all the courses a student must take and which semester of their academic career they will take the class. This process also looks into the Student database and retrieves the student’s information to display on the web page, such as, name, gpa, cin, and email. Thus we have the Student Roadmap.



**4.2 Student Roadmap and Admin CFD**

Admin has the ability to change which semester a course is available in. When Admin does this, it affects the Student Roadmap. Student Roadmap has a display roadmap request so it checks the availability of the courses in order to generate the roadmap. When Admin requests to change which semester a course is available in, it runs process 3 in our diagram which then updates the availability of courses and the Student Roadmap generates its roadmap accordingly to the new availability of courses.

**5. Policies and Tactics**

Our design policy was to make both the admin page, and the student page, as simple and clean as possible. The tactic we used to to accomplish this was to use bootstrap for our css in order to have a similar, and a smooth transitional feel between all pages. Originally, our plan was to manually create all the css files and then have each page reference them from a separate repository. After researching and brainstorming what would be the best way to implement css be, we decided that it would be most convenient to just use bootstrap for most of the css load. Another design issue we faced that was later changed was the original design layout we had envisioned for the admin and the student pages. Our original plan was to design the admin page to have a more “linkedin” type of feel. Different color pattern, more information displayed about the admin and information of the department and students as well in a scrollable timeline. We were then going to have the students page display more information about themselves as well as a different color scheme and drop down menu buttons. However, after a few mock ups, and considering the design direction we wanted to go, we felt it was best to go with the design we did and keeping both pages as simple and clean as possible.

**5.1 Choice of which specific products used**

* IDE: Eclipse Oxygen
* Languages: Java, Javascript, HTML
* Database: MySql

**5.2 Plans for ensuring requirements traceability**  
 None

**5.3 Plans for testing the software**

5.3.1:

We chose the design of buttons to switch from classes to students in the admin page because we found it to be more clean and direct than if we would have made links to separate web pages in the navigation bar.

5.3.2:

The current build of the website is backed by the latest version of java, HTML and CSS. As updated versions of the languages are released, we will update our code to the latest release of each language.

5.3.3

The end-users interface that we decided to do was a simple interface designed to give the user the information they desired as clean and as direct as possible. No multiple interactions with the user to find basic information that should be presented to them immediately. A clean and simple interface that gives the user the information they want.

5.3.4:

Use eclipse and create a series of sql, jsp and servlet files.Once files have been created, tested and compiled, all within eclipses developer/tester option, upload files onto the cs3 server.

5.3.5:

The database information is stored in a generic class model that stores the properties in its own values. The classes that communicate with SQL will need to be changed so that the sql queries and connection will match that of a different database.

**6. Detailed System Design**

**6.1 Admin**

**6.1.1 Responsibilities**

The responsibility of Admin is to control which semesters the courses in the course database are available in. Because the Admin exists, we are able to accommodate students by updating their personalized Roadmaps whenever the Admin changes the availability of a course (See section 4.2 for visual representation).

**6.1.2 Constraints**

The Admin is not able to render a course unavailable. The Admin can only change which semester a course is available in. This constraint keeps the Admin from deleting a course in the course database.

**6.1.3 Composition**

Admin has a subcomponent that allows it to change a course’s Coordinator in the course database. Admin also has a subcomponent that allows it to change a course’s Semester in the course database. Admin also may view all courses in the database as well as all students in the student database.

**6.1.4 Uses/Interactions**

The Admin entity affects the Student Roadmap entity and the course database.

**6.1.5 Resources**

The Admin relies on the following databases: courses, persons, and cins.

**6.1.6 Interface/Exports**

None

**6.2 Roadmap**

**6.2.1 Responsibilities**

The responsibility of the roadmap is to display accurate information on the availability of the courses listed in a student’s roadmap. This means that whenever a course’s semester available is changed then the roadmap will automatically move that course to another semester and replace it with a different class.

**6.2.2 Constraints**

The roadmap relies on the information provided by a school regarding a student.

**6.2.3 Composition**

The roadmap displays all courses listed in the semester database for each student. The roadmap may also view a list of all courses offered and the student’s information such as name, cin, gpa and email.

**6.2.4 Uses/Interactions**

The Student Roadmap entity does not affect other entities but it is affected by the Admin.

**6.2.5 Resources**

The Roadmap relies on the following databases: courses, persons, cins, and semesters.

**6.2.6 Interface/Exports**

None

**7. Detailed Lower level Component Design**

**7.1 Login Controller**

**7.1.1 Classification**

Java servlet and JSP

**7.1.2 Processing Narrative (PSPEC)**

Refer to 7.1.4.5

**7.1.3 Interface Description**

Leads to the JSP of the same name that displays a form requesting a username and password.

**7.1.4 Processing Detail**

Refer to 7.1.4.5

**7.1.4.1 Design Class Hierarchy**

Parent Class: Register Controller

Child Classes: Register Controller, Home Controller, Admin Home Controller

**7.1.4.2 Restrictions/Limitations**

Limited by the users in the database.

**7.1.4.3 Performance Issues**

None

**7.1.4.4 Design Constraints**

None

**7.1.4.5 Processing Detail For Each Operation**

* Retrieves information from the user input
* information is run through SQL queries
* SQL database sends back confirmed data
* creates a session for user
* If student, sends to Home Controller
* Else if administrator, sends to Admin Home Controller
* Else, invalid user input

**7.2 Registration Controller**

**7.2.1 Classification**

Java servlet and JSP

**7.2.2 Processing Narrative (PSPEC)**

Refer to 7.2.4.5

**7.2.3 Interface Description**

Leads to the JSP of the same name that displays a form requesting a username, name, email, cin, password, and password confirmation.

**7.2.4 Processing Detail**

Refer to 7.2.4.5

**7.2.4.1 Design Class Hierarchy**

Parent Class: Login Controller

Child Classes: Login Controller

**7.2.4.2 Restrictions/Limitations**

Limited by the users in the database.

**7.2.4.3 Performance Issues**

None

**7.2.4.4 Design Constraints**

None

**7.2.4.5 Processing Detail For Each Operation**

* Retrieves information from the user input
* Sends username, cin, email, and password to javascript for validation
* information is run through SQL queries
* updates SQL database
* Redirects to Login Controller

**7.3 Home Controller**

**7.3.1 Classification**

Java servlet and JSP

**7.3.2 Processing Narrative (PSPEC)**

Refer to 7.3.4.5

**7.3.3 Interface Description**

Leads to the JSP of the same name that displays a sidebar of personal information, a navigation bar with a courses tab and logout tab, and the roadmap itself at the center. The roadmap is separated by year.

**7.3.4 Processing Detail**

Refer to 7.3.4.5

**7.3.4.1 Design Class Hierarchy**

Parent Class: Login Controller

Child Classes: Courses, Controller, Login Controller

**7.3.4.2 Restrictions/Limitations**

None

**7.3.4.3 Performance Issues**

None

**7.3.4.4 Design Constraints**

None

**7.3.4.5 Processing Detail For Each Operation**

* Retrieves user’s roadmap from database
* Allocates memory for a hashmap, where the key is the semester and the value is a list of classes pertaining to that semester
* stores hashmap in a variable to be sent to the JSP
* JSP organizes the roadmap and an algorithm separates fall and spring combinations by year
* Javascript allows a set of buttons to control each year, so the user does not need to scroll to view whole roadmap
* User shall be able to go to courses page, change photo for personalization, or logout

**7.4 Admin Home Controller**

**7.4.1 Classification**

Java servlet and JSP

**7.4.2 Processing Narrative (PSPEC)**

Refer to 7.4.4.5

**7.4.3 Interface Description**

Leads to the JSP of the same name that displays a sidebar with personal information, a navigation bar with only a logout button, and by default, it displays the courses and the ability to alter them, and a list of the students and their information

**7.4.4 Processing Detail**

Refer to 7.4.4.5

**7.4.4.1 Design Class Hierarchy**

Parent Class: Login Controller

Child Classes: Login Controller

**7.4.4.2 Restrictions/Limitations**

None

**7.4.4.3 Performance Issues**

None

**7.4.4.4 Design Constraints**

None

**7.4.4.5 Processing Detail For Each Operation**

* Retrieves courses and student info from database
* Allocates memory for two array lists, one for the courses and the other for the students
* stores lists in variables to be sent to the JSP
* Using Javascript, clicking on a course makes a hidden form pop up referring to the respective course.
* Submitting the form sends to Admin Courses Controller
* User shall be able to view student information, change photo for personalization or logout

**7.5 Admin Course Controller**

**7.5.1 Classification**

Java servlet only

**7.5.2 Processing Narrative (PSPEC)**

Refer to 7.5.4.5

**7.5.3 Interface Description**

No interface

**7.5.4 Processing Detail**

Refer to 7.5.4.5

**7.5.4.1 Design Class Hierarchy**

Parent Class: Admin Home Controller

Child Classes: Admin Home Controller

**7.5.4.2 Restrictions/Limitations**

None

**7.5.4.3 Performance Issues**

None

**7.5.4.4 Design Constraints**

None

**7.5.4.5 Processing Detail For Each Operation**

* Retrieves coordinator and class availability values passed by from previous page
* using those values, updates the database, incorporating shifting algorithm
* redirects back to the Admin Home Controller

**8. Database Design**

**Persons Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| person\_name | username | email | cin | password | is\_admin |
| stores a user’s name | stores an alias for the user | stores the email of the user | stores the cin of the user, which is already in database by the time of registration and must match | stores a non-encrypted password | a boolean that determines whether or not the user is an administrator |

**Cins Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| cin | email | gpa | is\_used |
| stores the cin of users | stores the email of users | stores the gpa of users | a boolean that determines whether or not a cin and email have been linked during registration |

**Semester Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| sem | cin | c1 | c2 | c3 | c4 |
| an int that signifies the semester | cin that links to a person | stores a class in the current semester | stores a class in the current semester | stores a class in the current semester | stores a class in the current semester |

**Courses Table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| code | name | units | coordinator | available | semester | prerequisite |
| the class code | the name of the class | how many units the class offers | the name of the coordinator teaching the course | a boolean that determines whether or not a class is available | the semesters the class is being offered in | the prerequisite of the class |

**9. User Interface**

**9.1 Overview of User Interface**

Student

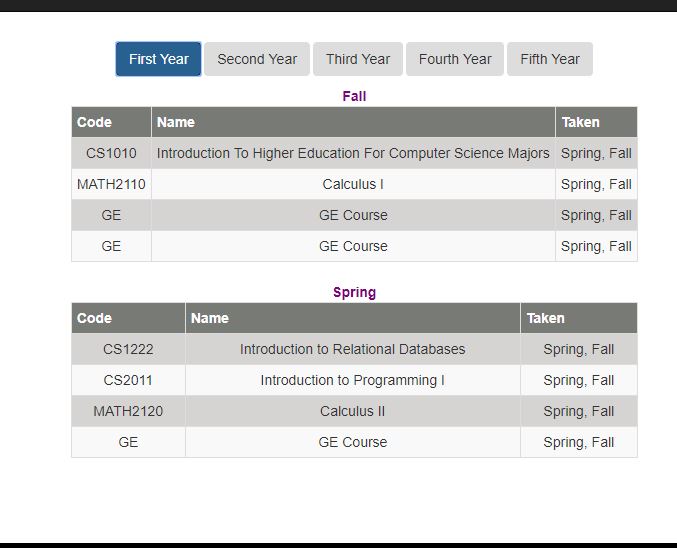
Students are allowed to register for WUMBO, which they can sign into whenever they want to view the classes they are suppose to take or view the available courses for that semester.

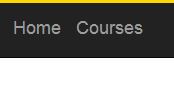
Admin

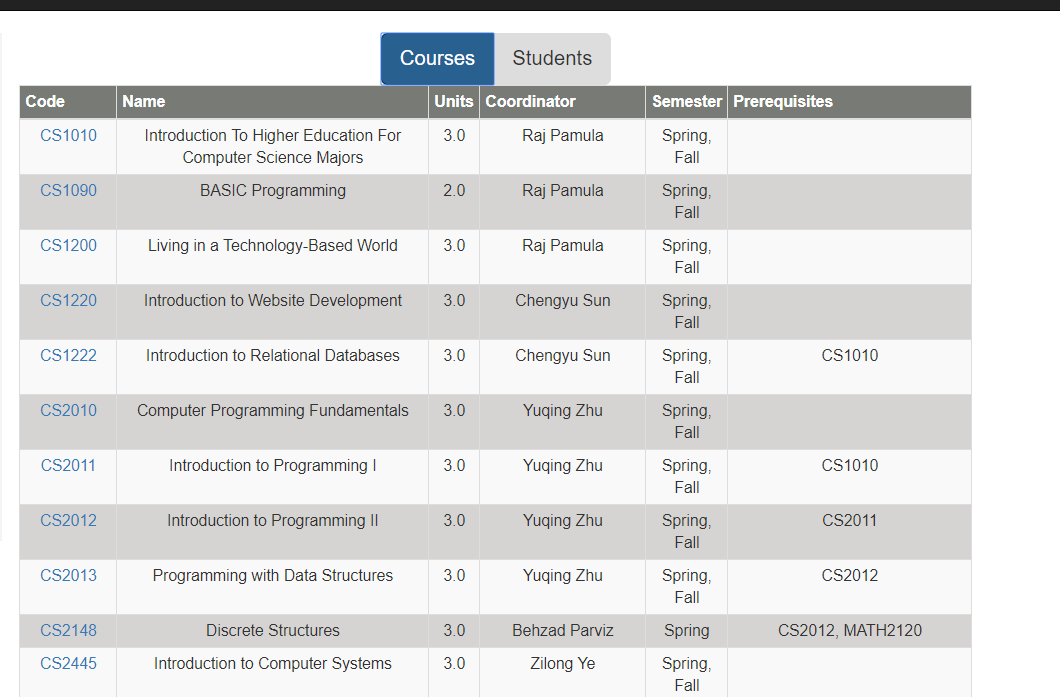
Admins are allowed to view students and course information. Also they are able to mess around with the courses values from changing coordinators to the availability of the course.

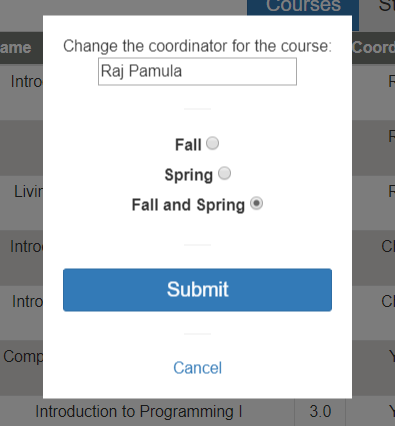
**9.2 Screen Frameworks or Images**

These can be mockups or actual screenshots of the various UI screens and popups.



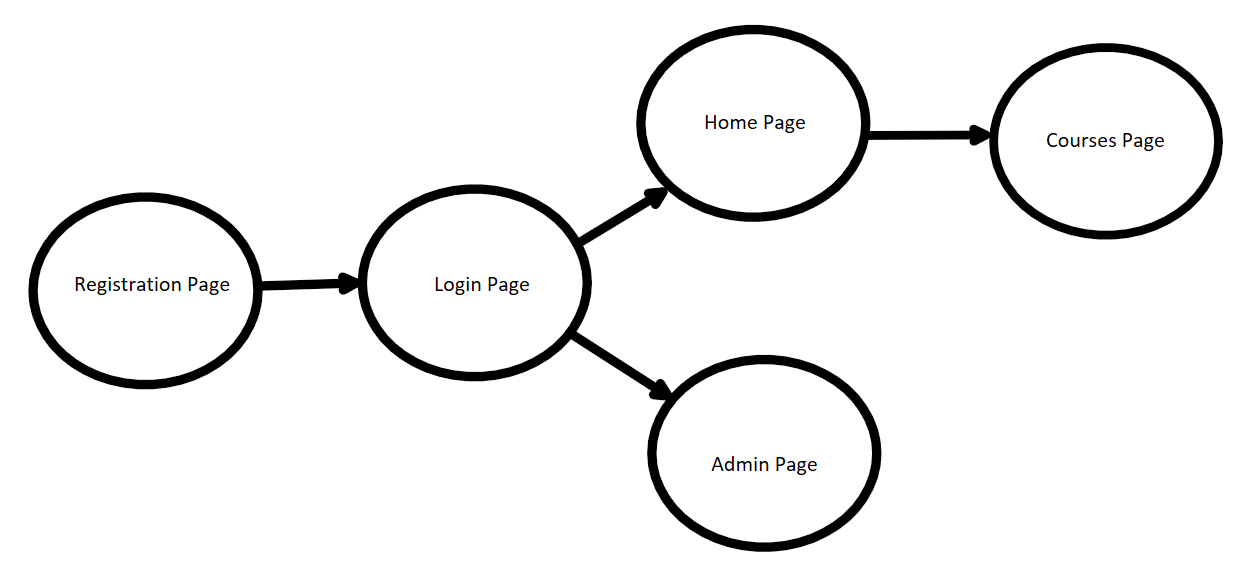






**9.3 User Interface Flow Model**

A discussion of screen objects and actions associated with those objects. This should include a flow diagram of the navigation between different pages.



**10. Requirements Validation and Verification**

|  |  |
| --- | --- |
| The system shall have a registration page for users. | Created a registration page that allows students to sign up |
| The system shall have a login page for users. | Created a login page that allows students to sign in whenever they want |
| The system shall have a home page for users. | Each students have their own home page |
| The system shall allow the users to see which courses they have taken. | Due to plans being changed, the group decided to only focus on incoming freshmen |
| The system shall allow the users to add classes. | Due to plans being changed, the group decided that the planner wouldn’t allow students to decide |
| The system shall allow the users to manage their classes. | Due to plans being changed, the group decided that the planner wouldn’t allow students to decide |
| The system shall have a webpage with a visual of the user’s IAP. | WUMBO has multiple tables that allows students to see which classes they have to take to finish within 5 years |
| The system might allow the admin to see what classes are being taken by students. | Due to time constraint we weren’t able to implement this function |

|  |  |
| --- | --- |
| The system shall validate the user’s password. | Login page checks whether the user input the right password, if not throws error and try again |
| The system shall validate the user’s username. | Login page checks whether the user input the right username, if not throws error and try again |
| The system shall verify a student has passed a course. | Due to plans being changed, the group decided to only focus on incoming freshmen |
| The system shall adjust the IAP when a student passes a course. | Due to plans being changed, the group decided to only focus on incoming freshmen |
| This system shall adjust the IAP when a student fails a course. | Due to plans being changed, the group decided to only focus on incoming freshmen |
| The system shall adjust the IAP when the admin has toggled a course available/unavailable. | In the admin page, admin has the control to change coordinator and availability |

|  |  |
| --- | --- |
| The algorithm that creates the IAP, that follows the student’s progression throughout their bachelor's, will interact with the MySQL database on the CS3 server. When the student verifies which prerequisites they have met the student IAP table will be updated on the database. When the Admin toggles courses available/not available the database will be updated. | In the admin page, admins are allowed to change the coordinator and the availability of any courses. Which then updates the student's IAP. (Individualized Assessment Plan) |
| The IAP will be visible to the user on a webpage. | Student home page displays multiple buttons that allows the user to interact with them, which shows the student IAP (Individualized Assessment Plan) in table form |

|  |  |
| --- | --- |
| student   * + - id - creates a unique id for each user     - username - a user’s unique username for the system     - email - this will be for future reference and emailing users     - cin - connects to the cins table, which already contains students from the school system     - password - a user’s password to login into the system | In the database, it shows all these information for each student. Taking up one table for all the student who will be signing up or who already signed up |
| cins   * + - cin - students cin which is already is system     - gpa - GPA of student     - is\_used - determines whether an account has already be created using the particular cin | This information is used to determine the courses each student is taking. There CIN is connected to courses table |
| student\_iap   * + - student\_id - links to the student     - course1     - course2     - course3     - course4     - course5     - course6       * each course a student takes stores the id of said course. If all courses are not currently being taken, then the remaining course cells will be nil. | In our database, there is a table that implement this information, which shows each student’s courses and their CIN |
| courses   * + - is\_available - determines if the class is being offered     - code - the class code     - name - name of the course on CSNS     - units - number of units for the class     - coordinator - the coordinator of the course on CSNS     - semester - which semester the course is being offered (season year) | In our database, we have a table that holds all these information. We also use these information to set up the student courses that they have to take. |

**11. Glossary**

* WUMBO - project name, standing for Web-User Major-Bound Organizer
* Roadmap - a guideline of the courses the students need to graduate
* IAP - Individualized Assessment Planner
* mySQL - the database being used
* Tomcat - the local web server being used
* CS3 - school server
* SQL queries - statements made to retrieve certain data from mySQL database

**12. References**

* Our own SRS document