Design Document

CSCE 361 - Summer 2017

Class Enrollment System

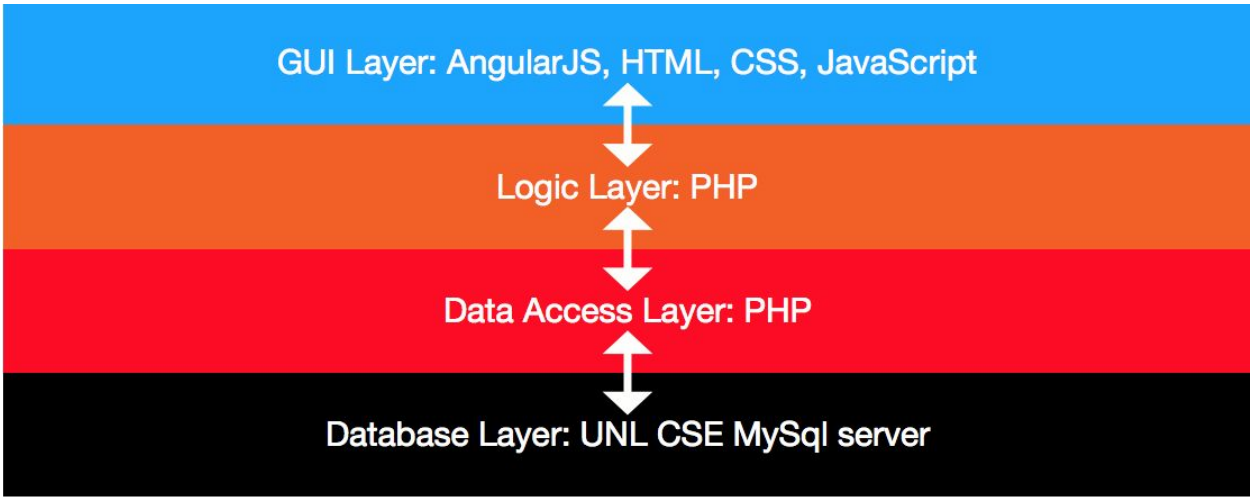
Prepared by Team 1: Edmon Adams, Josh Jones, and Trevor Fellbaum

1. Introduction

The purpose of this design document is to demonstrate the high level architecture and entity relations for the course enrollment system. This document will include architecture and class diagrams showing the relationship between different parts of the system including the database. The target audience for this document is software engineers who will be tasked with developing this design.

2. Architecture

2.1. Introduction



The top architecture of our system will be in the style of a layered system architecture. This architecture was selected because it captures the nature of our system. This is because each layer has potential for change in the future. For example, changes may be made to the GUI, and will have low cost because this architecture will allow the rest of the system to be reused.

The first layer is the GUI layer where the graphical user interface provides admins and students with the power to view and make decisions on the information presented to them. The information presented will be based on the returned decisions made by our Logic Layer that returns the information gathered from the Data Access Layer in a format understandable by the GUI Layer. The Data Access Layer handles data retrieval and insertion to the MySql database. The Database Layer is where all the pertinent information is kept on a UNL CSE hosted MySQL database.

2.2. Modules

2.2.1. GUI Layer

This layer consists of the interface that the user will interact with. This is where student users will be able to login to the system to access their course history and course checkout system. Admin users will interact with this layer to create and delete majors, minors, and courses as well as edit any information related to these objects. This layer will use the

HTML, CSS, and JavaScript (with the AngularJS framework) languages to create webpages, communicate information to the user, and allow users to interact with the system. This GUI layer will be optimized for different screen resolutions as a response to increasing mobile traffic on the Internet. The usage of the AngularJS framework will make this an easy optimization.

2.2.2. Logic Layer

This layer will perform the task of deciding what information the user needs. This information will be based on whether the user is a student or an admin. Such information includes:

1. Students would need what classes are available based on the filter criteria the student specified in the GUI Layer. Also the student will need their course history to see if they satisfy any prerequisites or college, major, or minor requirements.
2. Admins would need current information about majors, minors, college requirements and classes. This information will be gathered by passing on the necessary flags and requested information in JSON format so that the Data Access Layer can interpret the decisions made in the Logic Layer. The returned response will also be handled by translating it into JSON and sending it back to the GUI Layer for display.

This layer will also handle any data the user wants to store in the database. This layer will need to transform the inputted user data into JSON so that the Data Access Layer can interpret the decisions made in the Logic Layer.

2.2.3. Data Access Layer

This layer will run queries on our MySQL database within the Database Layer with the necessary flags that the Logic Layer decided on. This will then allow the system to store the pertinent information and retrieve information as needed. The information gathered will be returned to the Logic Layer for processing to send to the GUI Layer. Also, success and failure messages will be relayed to the Logic Layer for any update queries performed so that the GUI Layer can display feedback to the user.

2.2.4. Database Layer

The database layer is responsible for holding all the data for the system and defining the relationships between the data. The system will be using a MySQL database hosted on the UNL CSE Servers. The relationship between the data within the database is shown in section 3.1.1

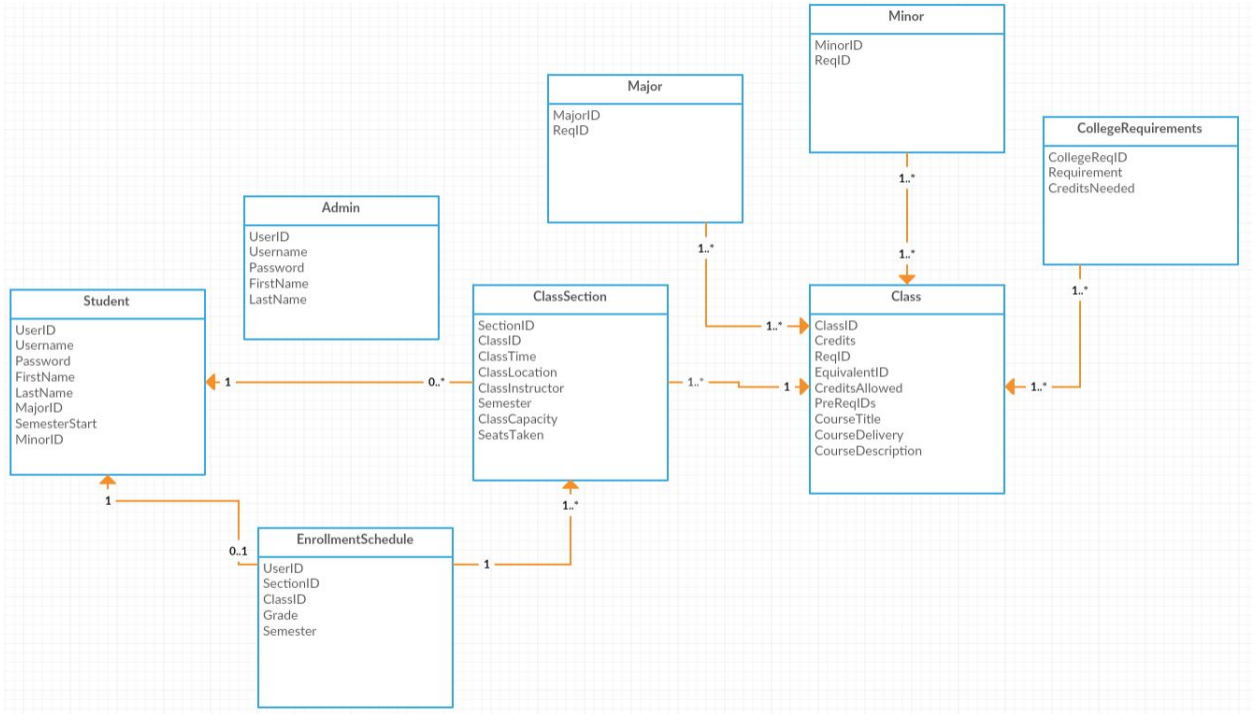
3. Class Diagrams

3.1. Data Table Classes

The system will be using a MySQL database to store all pertinent information

for the system. In section 3.1.1 a schema is shown to describe the information and the relationship between this information.

3.1.1. Database Schema



3.1.2. Schema Information

3.1.2.1. Student

Holds the data for a single student user including information such as their user ID, username, password, name, major, minor, and the semester they started.

3.1.2.2. Admin

Holds the data for a single admin user including information such as their user ID, username, password, and name.

3.1.2.3. Enrollment Schedule

This table will represent the relationship between a student and what classes they have/are currently enrolled in as well as a grade and a semester. Each record will have a student and none or a single Enrollment Schedule.

3.1.2.4. Class Section

Holds the data for a class section including information such as the Section ID, the Class ID (the class it is associated with), a Class Time, a Class Location, a Class Instructor, a Class Capacity, a current count of the Seats Taken, and the Semester the class section was held.

3.1.2.5. Class

Holds the data for a class including a Class ID, the amount of Credits the class is worth, a Requirement ID (shows what college requirement the class fulfills), an Equivalent ID (shows which classes this class is equivalent to), the Credits Allowed (how many credits of this course are you eligible to use toward your degree), the Prerequisite IDs, a Course Title, a Course Delivery, and a Course Description.

3.1.2.6. College Requirements

Holds the data for a single college requirement including a Requirement ID, the Requirement it fulfills, and the amount of Credits Needed for the requirement. This will identify what classes are required by that specific requirement.

3.1.2.7. Major

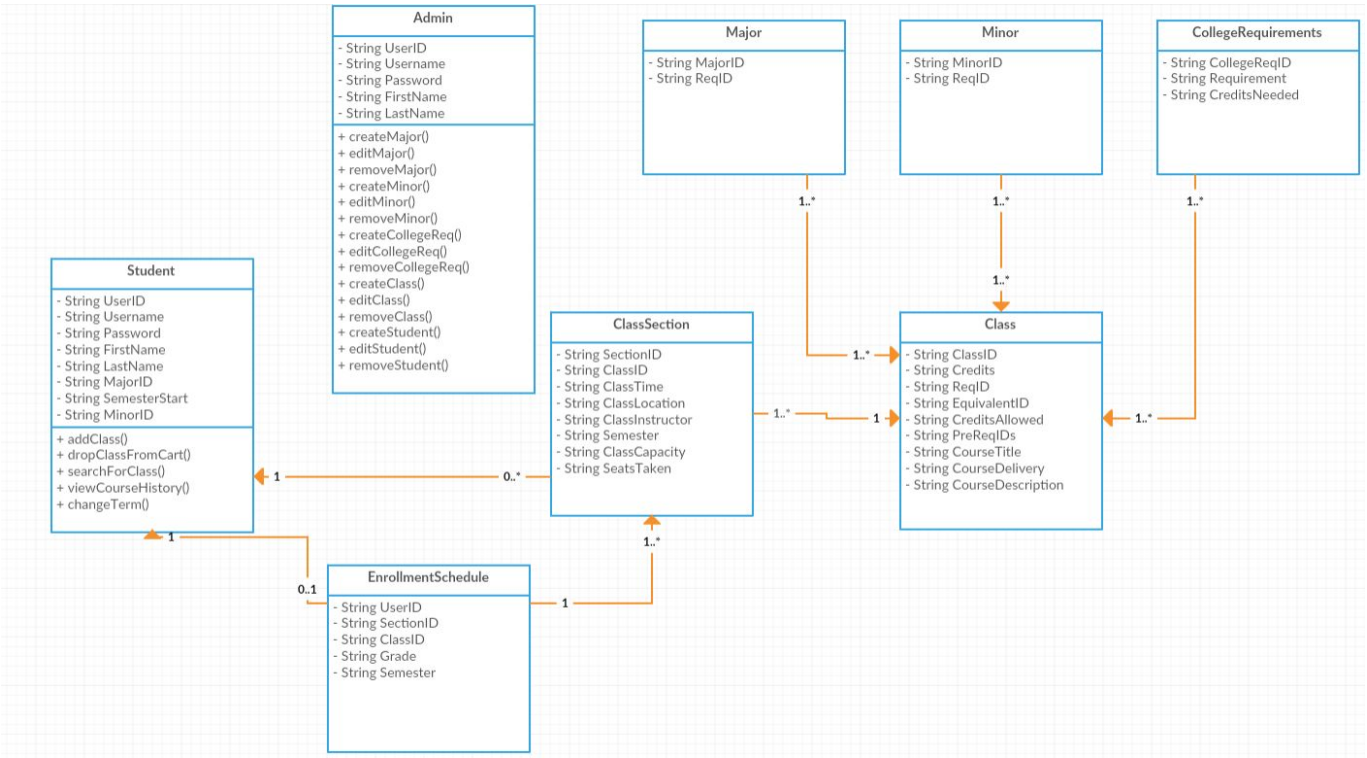
Holds the data for a single major including Major ID and Requirement ID. This will identify what classes are required by that specific major.

3.1.2.8. Minor

Holds the data for a single minor including Minor ID and Requirement ID. This will identify what classes are required by that specific minor.

3.2. Class Diagrams

Since the system is heavily database-focused, two different schemas are provided. The class diagram has the same multiplicities and relationships as the database diagram. Also, all getter and setter operations will not be specified in the class diagram.



3.2.1. Schema Method Information

3.2.1.1. createMajor()

Creates a new major with the specified major ID and list of req IDs that make up the requirements for that major.

3.2.1.2. editMajor()

Allows the addition or subtraction of req IDs to the major.

3.2.1.3. removeMajor()

Allows the removal of a major.

3.2.1.4. createMinor()

Creates a new minor with the specified minor ID and list of req IDs that make up the requirements for that minor.

3.2.1.5. editMinor()

Allows the addition or subtraction of req IDs to the minor.

3.2.1.6. removeMinor()

Allows the removal of a minor.

3.2.1.7. createCollegeReq()

Creates a new college requirement with the specified requirement ID and the name of that requirement with the credits needed to fulfill it.

3.2.1.8. editCollegeReq()

Allows the addition or subtraction of credits needed to the college requirements as well as changing the name of the requirement.

3.2.1.9. removeCollegeReq()
Allows the removal of a college requirement.

3.2.1.10. createStudent()
Creates a new student with a user ID, name, account login information, a major, possibly a minor, a semester started.

3.2.1.11. editStudent()
Allows any changes to the current student fields.

3.2.1.12. addClass()
Adds a class to a specified term's enrollment schedule.

3.2.1.13. dropClassFromCart()
Drops a class from the student's shopping cart.

3.2.1.14. searchForClass()
Searches for classes based on filter and search criteria.

3.2.1.15. viewCourseHistory()
Views all the courses the student has taken for a specified term.

3.2.1.16. changeTerm()
Changes the term to be viewed.

3.2.1.17. createClass()
Creates a new valid class with a class section for a specified term.

3.2.1.18. removeClass()
Allows the removal of a class.

3.2.1.19. editClass()
Allows the addition and removal of any class information such as, but not limited to, the professor, class ID, time of instructions, and location.

3.2.1.20. removeStudent()
Allows the removal of a student.

3.3. GUI Layer

The top layer of our system consists of 2 login portals with two paths depending on if the user is an admin or a student. These paths take you to different interfaces where users can exercise their specific privileges.

When a user first navigates with the course enrollment system, the user is prompted with a login screen asking for username and password. After entering the information, the inputted information is verified with the database. If the login information is incorrect, they will be notified of the error. If the login information is correct, they will proceed to either the student or admin side of the application depending on the login portal.

The first screen for students will display their classes for the most recent term, which will be retrieved from the database, as well as have options to change term or 'shop' for courses. Changing the term merely refreshes the course information on the initial screen with the desired term. The shop/browse courses screen will

begin with a popover that prompts the student to choose which degree requirement they want to satisfy. After choosing, the screen will be populated with classes that matched the selection criteria from the database. The student can also search and go back to the filter screen. Clicking on a class will cause it to expand and begin the cart functionality. They will be asked their section preference, professor or time of instruction, and then choose the section that they prefer most. The student can navigate back to the initial screen to enroll in their classes. The student will then see a confirmation message before they enroll in the classes in their shopping cart.

The admin users will have an initial screen that has four options, students, majors, minors, and classes, depending on what their objective is. Once they choose an option on the side, they are prompted if they want to add, delete, or edit the information of the option they chose. If they chose to add the option, they will be prompted with a bunch of text fields where the admin needs to populate with the desired information. If they chose to delete an option, they can search and select a specific student, major, minor, or class depending on the option they chose and permanently remove that object. This also erases all its information from the database. The user will be prompted with a confirmation message before they can delete the object. If they chose to edit the option, the system will display all the text fields you would see if the user chose to add the option but, the text fields will be populated with the information from the database. The user can then edit the fields and save all changes to the database. Admin users will also have the option to add other student users.