Matthew L. Miller
CS 6310
Software Architecture & Design

Overview

The purpose of the Academic Resource Management System (ARMS) is to assist students in enrolling for courses and administrators in managing the courses available. Students will be able to view which courses are being offered for the next semester, create a list of courses they would like to take and rank them according to their priority, and receive the system's recommended schedule based on demand from other students. Administrators will have the ability to manage which courses are being offered, their enrollment caps, and the number of Professors and TA's available who can be assigned to teach the courses. The system's scheduling engine will seek to produce student schedules that balance demand for courses with the Professor and TA resources available for them.

Requirements Analysis

The primary functional requirements include the ability for students and administrators to access information concerning courses offered by the university and make decisions regarding course enrollment. The system will also regulate enrollment by capping the number of students who may be enrolled in each course, restricting students from taking courses if they have not met the pre-requisite requirements, and assisting students to meet the course needs of their major.

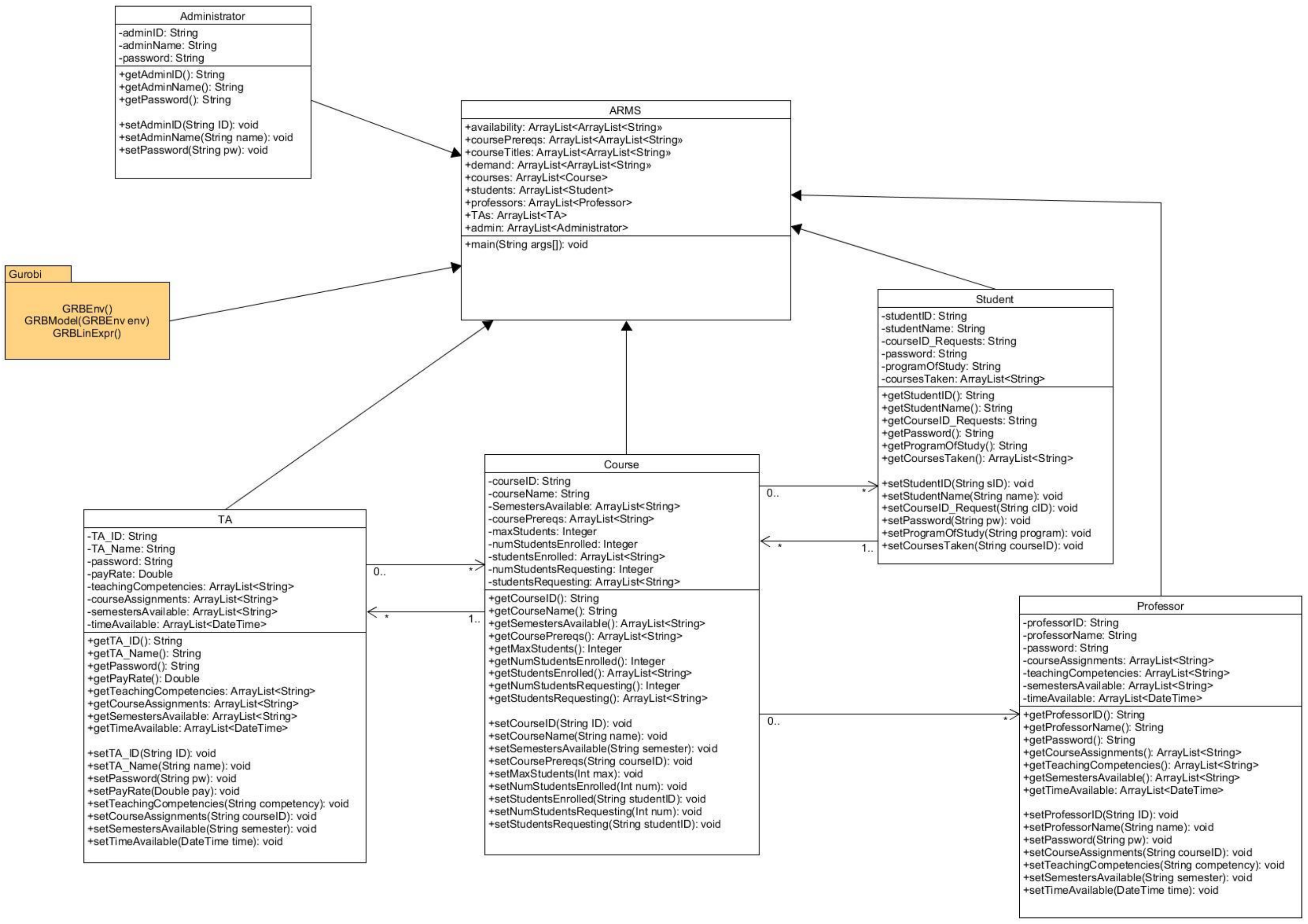
For students, the program will allow them to create a list of desired courses, rank the courses in order of the priority the student places on taking each course, and then recommend an optimal schedule for their courses, based on course availability. Students will also be able to view the courses they requested in past semesters as well as the number of students enrolled in and requesting enrollment in a particular course. The system will also ensure that students have met any pre-requisite requirements for the courses they choose and advise them as to whether the courses fill particular requirements for their major. In addition, it will advise them on what courses would be best for them to take in the future depending on their current course selections.

Administrators will have the ability to view the number of students enrolled in or requesting enrollment in a particular course as well as the ability to change enrollment caps on courses depending on demand. Administrators will also be able to maintain the overall course catalog, editing information about courses, and adding or removing courses as needed. The program will also allow administrators to view the pool of available Professors and TA's for a particular semester and assign them to particular courses based on their teaching competencies.

Non-functional requirements include the need for each user, whether student or admin, to create a unique user ID and password to guarantee security of the system. Each user will be required to enter their ID and password before accessing the system. Additionally, the program will require the development of a GUI which allows users to execute their desired actions through a series of menus. It will also need to access and maintain a database in order for students to view the history of their course requests and for administrators to maintain the course catalog.

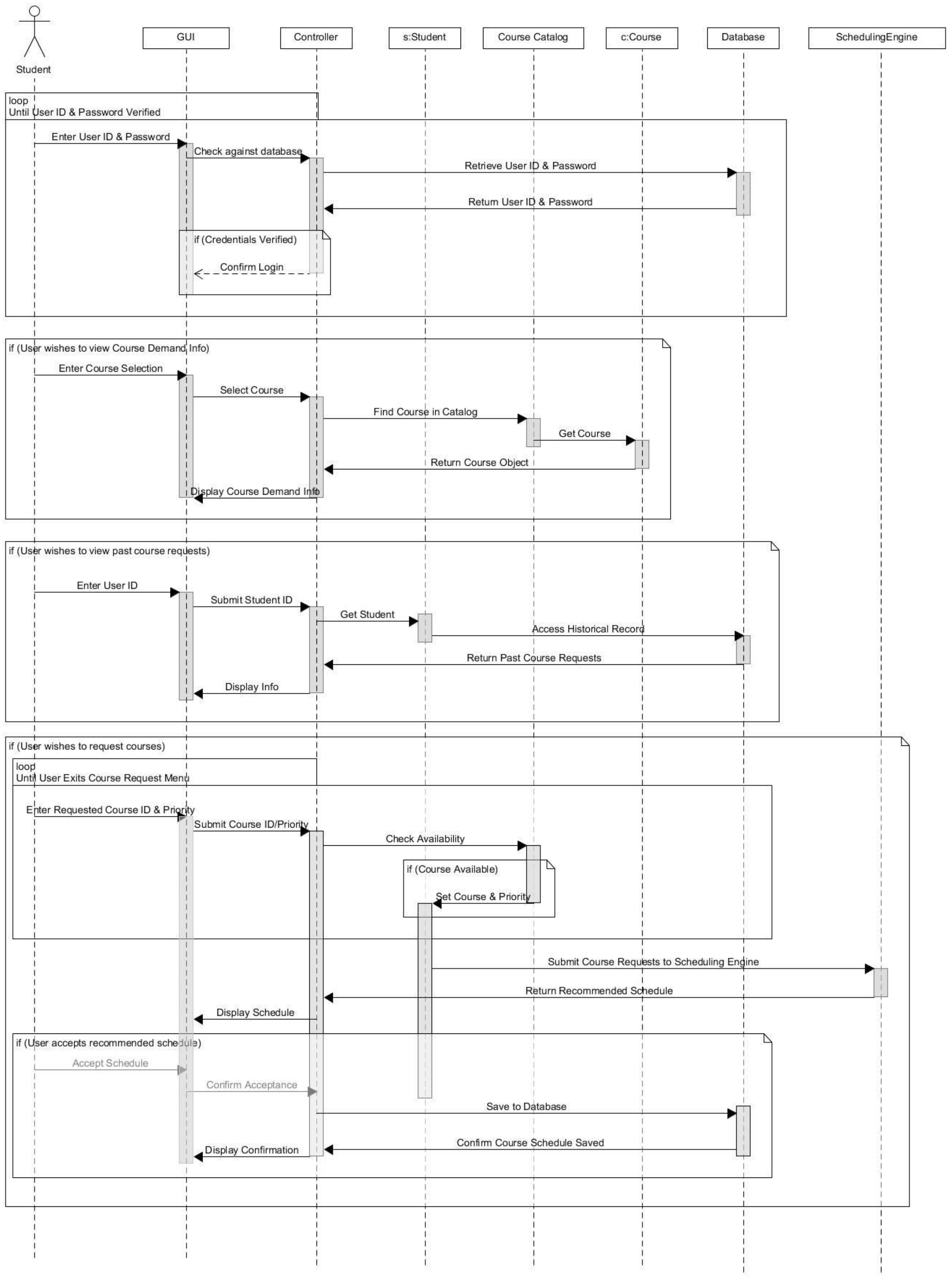
Overall, the system will seek to maximize the ability of students to enroll in their desired courses while balancing the optimal availability of the courses and the need to hire additional Professors and TA's.

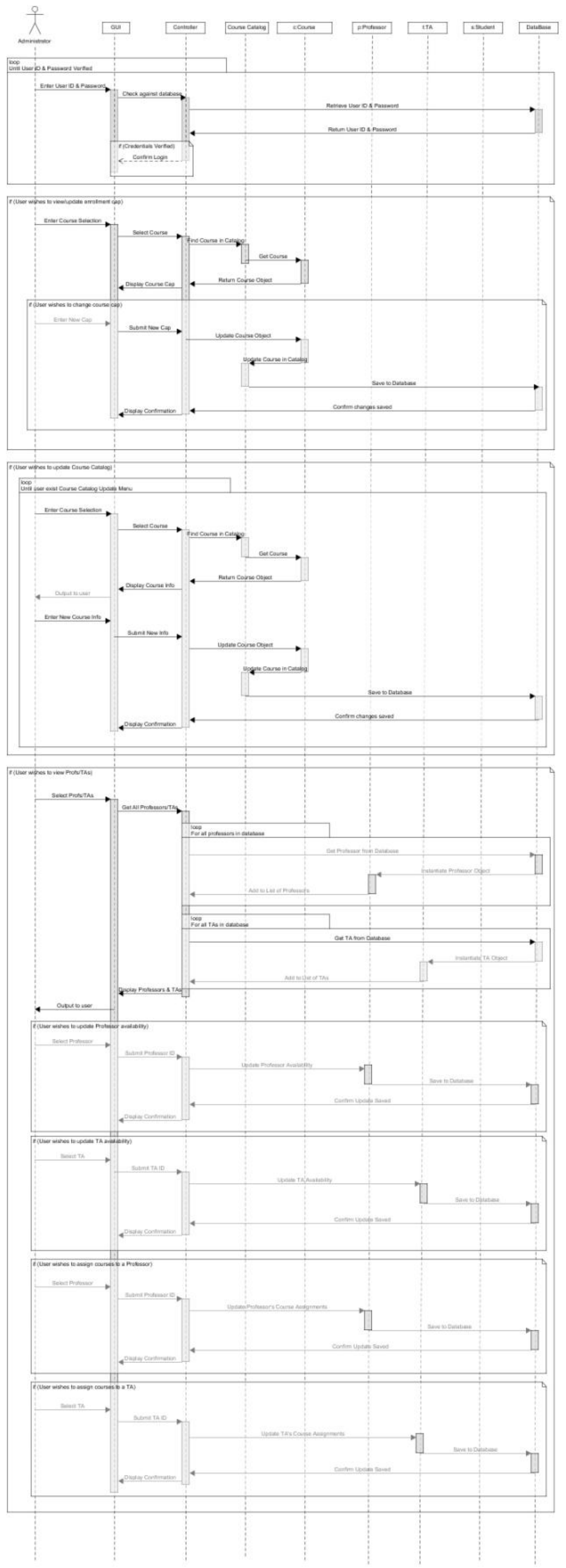
UML Class Model Diagram


Administrator
-adminID: String
-adminName: String
-password: String
+getAdminID(): String
+getAdminName(): String
+getPassword(): String
+setAdminID(String ID): void
+setAdminName(String name): void
+setPassword(String pw): void
ARMS
+availability: ArrayList<ArrayList<String»
+coursePrereqs: ArrayList<ArrayList<String»
+courseTitles: ArrayList<ArrayList<String»
+demand: ArrayList<ArrayList<String»
+courses: ArrayList<Course>
+students: ArrayList<Student>
+professors: ArrayList<Professor>
+TAs: ArrayList<TA>
+admin: ArrayList<Administrator>
+main(String args[]): void
Gurobi
GRBEnv()
GRBModel(GRBEnv env)
GRBLinExpr()
Student
-studentID: String
-studentName: String
-courseID_Requests: String
-password: String
-programOfStudy: String
-coursesTaken: ArrayList<String>
+getStudentID(): String
+getStudentName(): String
+getCourseID_Requests: String
+getPassword(): String
+getProgramOfStudy(): String
+getCoursesTaken(): ArrayList<String>
+setStudentID(String sID): void
+setStudentName(String name): void
+setCourseID_Request(String cID): void
+setPassword(String pw): void
+setProgramOfStudy(String program): void
+setCoursesTaken(String courseID): void
Course
-courseID: String
-courseName: String
-SemestersAvailable: ArrayList<String>
-coursePrereqs: ArrayList<String>
-maxStudents: Integer
-numStudentsEnrolled: Integer
-studentsEnrolled: ArrayList<String>
-numStudentsRequesting: Integer
-studentsRequesting: ArrayList<String>
+getCourseID(): String
+getCourseName(): String
+getSemestersAvailable(): ArrayList<String>
+getCoursePrereqs(): ArrayList<String>
+getMaxStudents(): Integer
+getNumStudentsEnrolled(): Integer
+getStudentsEnrolled(): ArrayList<String>
+getNumStudentsRequesting(): Integer
+getStudentsRequesting(): ArrayList<String>
+setCourseID(String ID): void
+setCourseName(String name): void
+setSemestersAvailable(String semester): void
+setCoursePrereqs(String courseID): void
+setMaxStudents(Int max): void
+setNumStudentsEnrolled(Int num): void
+setStudentsEnrolled(String studentID): void
+setNumStudentsRequesting(Int num): void
+setStudentsRequesting(String studentID): void
TA
-TA_ID: String
-TA_Name: String
-password: String
-payRate: Double
-teachingCompetencies: ArrayList<String>
-courseAssignments: ArrayList<String>
-semestersAvailable: ArrayList<String>
-timeAvailable: ArrayList<DateTime>
+getTA_ID(): String
+getTA_Name(): String
+getPassword(): String
+getPayRate(): Double
+getTeachingCompetencies: ArrayList<String>
+getCourseAssignments: ArrayList<String>
+getSemestersAvailable: ArrayList<String>
+getTimeAvailable: ArrayList<DateTime>
+setTA_ID(String ID): void
+setTA_Name(String name): void
+setPassword(String pw): void
+setPayRate(Double pay): void
+setTeachingCompetencies(String competency): void
+setCourseAssignments(String courseID): void
+setSemestersAvailable(String semester): void
+setTimeAvailable(DateTime time): void
Professor
-professorID: String
-professorName: String
-password: String
-courseAssignments: ArrayList<String>
-teachingCompetencies: ArrayList<String>
-semestersAvailable: ArrayList<String>
-timeAvailable: ArrayList<DateTime>
+getProfessorID(): String
+getProfessorName(): String
+getPassword(): String
+getCourseAssignments(): ArrayList<String>
+getTeachingCompetencies(): ArrayList<String>
+getSemestersAvailable(): ArrayList<String>
+getTimeAvailable: ArrayList<DateTime>
+setProfessorID(String ID): void
+setProfessorName(String name): void
+setPassword(String pw): void
+setCourseAssignments(String courseID): void
+setTeachingCompetencies(String competency): void
+setSemestersAvailable(String semester): void
+setTimeAvailable(DateTime time): void
0..
*
*
1..
0..
*
*
1..
0..
*

The class model diagram shows the main classes needed for the system. They include Administrator, Student, Professor, TA, and Course. The Course class has a one-to-many relationship with the Student, TA, and Professor classes since a student will likely be taking multiple classes while Professors and TA’s may be assigned to multiple classes. Each course may also have multiple TA’s as well. In addition, the diagram includes the Gurobi package used in generating optimal course schedules.

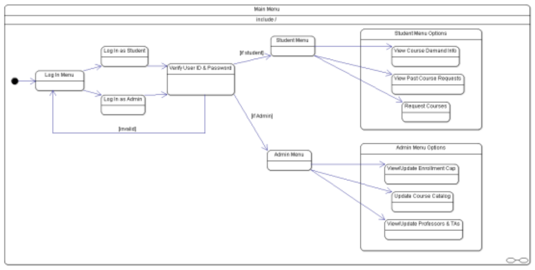
Sequence Diagrams – Student & Administrator

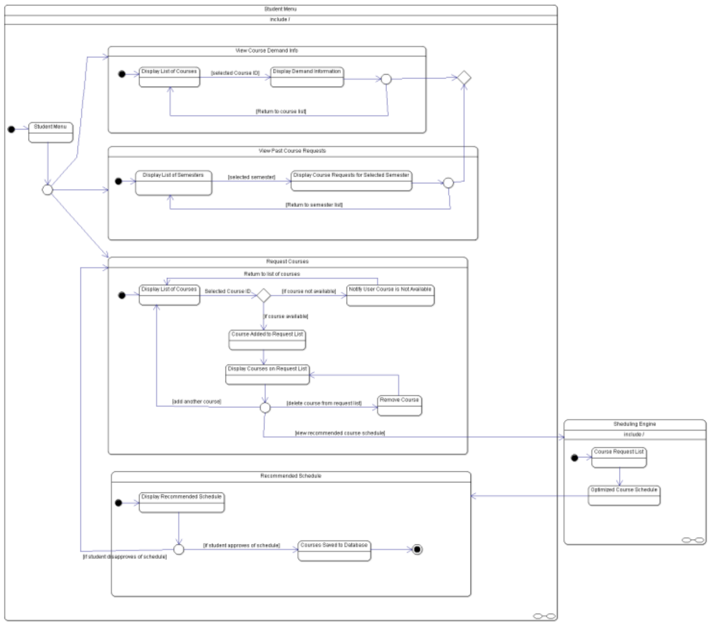

Student
GUI
Controller
s:Student
Course Catalog
c:Course
Database
SchedulingEngine
loop
Until User ID & Password Verified
Enter User ID & Password
Check against database
Retrieve User ID & Password
Return User ID & Password
if (Credentials Verified)
Confirm Login
if (User wishes to view Course Demand Info)
Enter Course Selection
Select Course
Find Course in Catalog
Get Course
Return Course Object
Display Course Demand Info
if (User wishes to view past course requests)
Enter User ID
Submit Student ID
Get Student
Access Historical Record
Return Past Course Requests
Display Info
if (User wishes to request courses)
loop
Until User Exits Course Request Menu
Enter Requested Course ID & Priority
Submit Course ID/Priority
Check Availability
if (Course Available)
Set Course & Priority
Submit Course Requests to Scheduling Engine
Return Recommended Schedule
Display Schedule
if (User accepts recommended schedule)
Accept Schedule
Confirm Acceptance
Save to Database
Confirm Course Schedule Saved
Display Confirmation

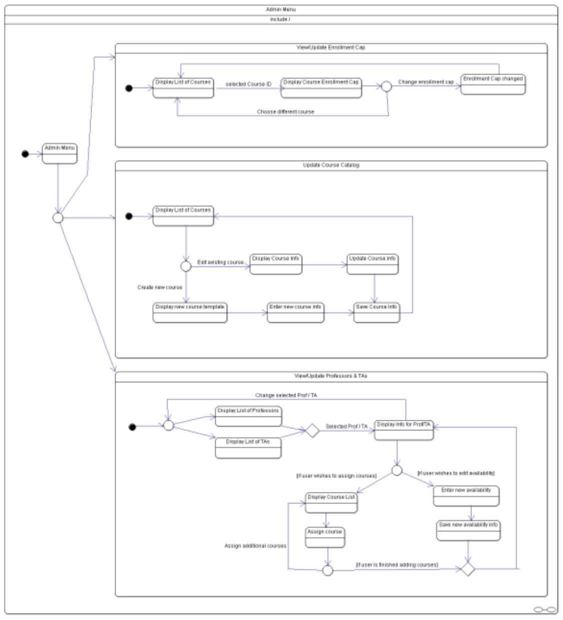

Administrator
GUI
Controller
Course Catalog
c:Course
p:Professor
t:TA
s:Student
DataBase
loop
Until User ID & Password Verified
Enter User ID & Password
Check against database
Retrieve User ID & Password
Return User ID & Password
if (Credentials Verified)
Confirm Login
if (User wishes to view/update enrollment cap)
Enter Course Selection
Select Course
Find Course in Catalog
Get Course
Return Course Object
Display Course Cap
if (User wishes to change course cap)
Enter New Cap
Submit New Cap
Update Course Object
Update Course in Catalog
Save to Database
Confirm changes saved
Display Confirmation
if (User wishes to update Course Catalog)
loop
Until user exist Course Catalog Update Menu
Enter Course Selection
Select Course
Find Course in Catalog
Get Course
Return Course Object
Display Course Info
Output to user
Enter New Course Info
Submit New Info
Update Course Object
Update Course in Catalog
Save to Database
Confirm changes saved
Display Confirmation
if (User wishes to view Profs/TAs)
Select Profs/TAs
Get All Professors/TAs
loop
For all professors in database
Get Professor from Database
Instantiate Professor Object
Add to List of Professors
loop
For all TAs in database
Get TA from Database
Instantiate TA Object
Add to List of TAs
Display Professors & TAs
Output to user
if (User wishes to update Professor availability)
Select Professor
Submit Professor ID
Update Professor Availability
Save to Database
Confirm Update Saved
Display Confirmation
if (User wishes to update TA availability)
Select TA
Submit TA ID
Update TA Availability
Save to Database
Confirm Update Saved
Display Confirmation
if (User wishes to assign courses to a Professor)
Select Professor
Submit Professor ID
Update Professor's Course Assignments
Save to Database
Confirm Update Saved
Display Confirmation
if (User wishes to assign courses to a TA)
Select TA
Submit TA ID
Update TA's Course Assignments
Save to Database
Confirm Update Saved
Display Confirmation

The sequence diagrams illustrate the steps taken by the system to perform the different events a student or administrator may wish to perform. Both sequence diagrams begin with the sequence of events required when a user enters their user ID and password. If the user is student, they may then view information about course enrollment and demand, view their history of course requests, or add courses to their course request list and have the system determine their optimal schedule. Administrators can view and change enrollment caps, make changes to the Course Catalog, view and update information about Professor and TA availability, and assign Professors and TA’s to courses.

Statechart Diagram

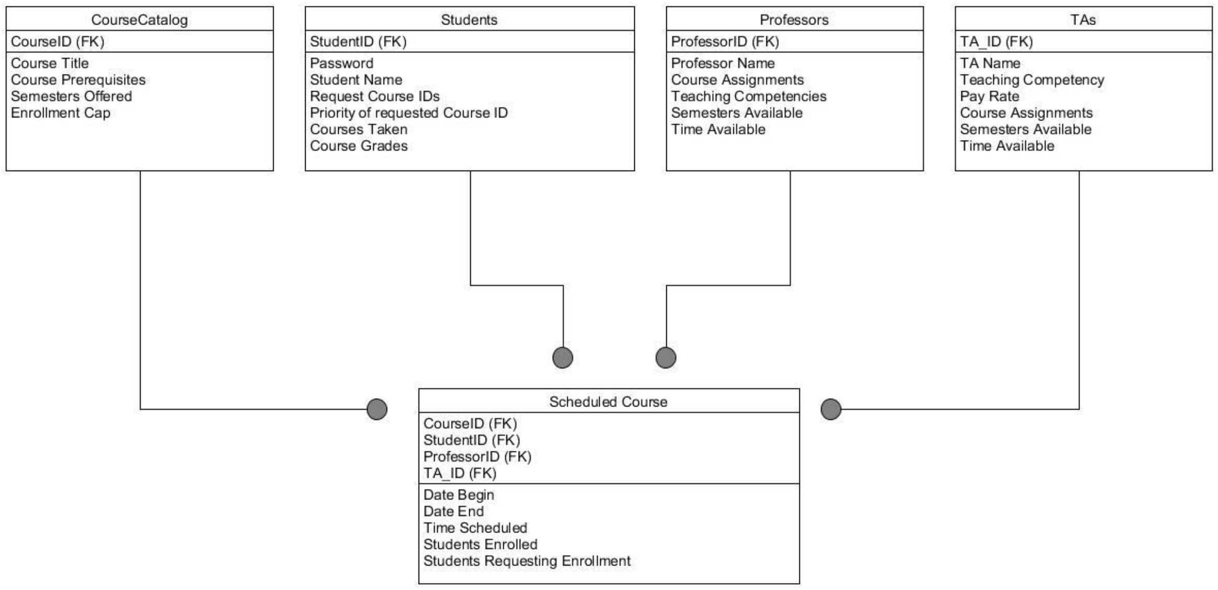



Student Menu
include /
Student Menu
View Course Demand Info
Display List of Courses
[selected Course ID]
Display Demand Information
[Return to course list]
View Past Course Requests
Display List of Semesters
[selected semester]
Display Course Requests for Selected Semester
[Return to semester list]
Request Courses
Return to list of courses
Display List of Courses
Selected Course ID
[if course not available]
Notify User Course is Not Available
[if course available]
Course Added to Request List
Display Courses on Request List
[add another course]
[delete course from request list]
Remove Course
[view recommended course schedule]
Sheduling Engine
include /
Course Request List
Optimized Course Schedule
Recommended Schedule
Display Recommended Schedule
[if student approves of schedule]
Courses Saved to Database
[if student disapproves of schedule]



The statechart diagram illustrates the state of the system from the log-in screen through the various actions a user may wish to perform. The user may log in as a student or an administrator and will be given different menu options after log in. From the initial menu screen, the system goes into different states for students and admin, allowing each to perform exclusive actions. The diagram also includes the scheduling engine, which is used by a student user to determine the optimal course schedule based on their course request list.

Logical Data Model



The logical data model illustrates the different tables held in the system's database used for storing information. It includes tables that correspond to the major classes, such as Students, Professors, and TAs, all of which include fields corresponding to each class's respective attributes. It also includes a table for the overall Course Catalog, which includes information pertaining to all courses the university offers, such as their enrollment cap, pre-requisites, and the semesters in which it is offered. If a course is being offered in the semester for which registration is taking place, then a Scheduled Course table is derived from the other four tables in which the Course ID is matched with the foreign key(s) for the Professor, students enrolled in the course and those requesting enrollment, and the TAs assigned to the course.

Architecture

The main components for the system's architecture include the GUI, the Controller, the Course Catalog, scheduling engine and the database.

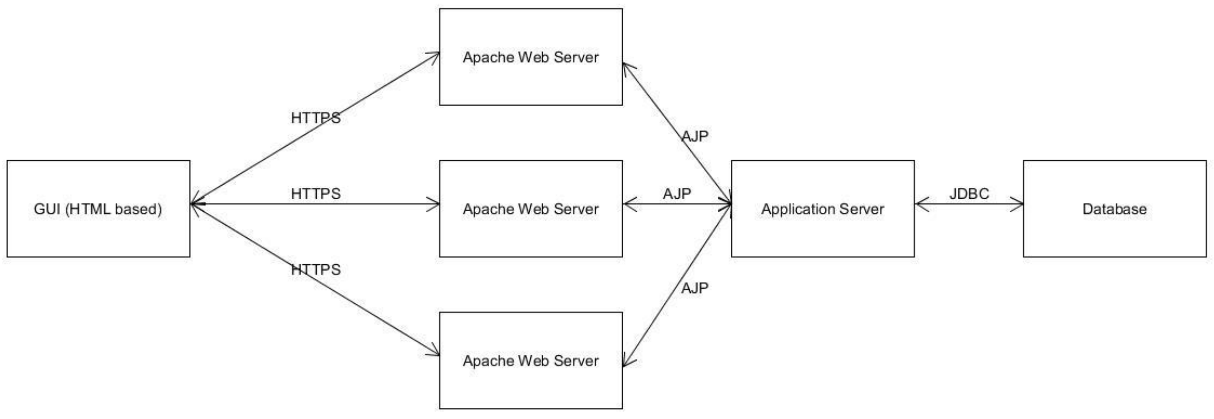
- **GUI** – The GUI provides the menus with which the users interact with the system. It would be a web-based GUI accessible through any standard web browser. Each user begins at a log in screen, and then, depending on whether the user is a student or administrator, is taken to a screen with menu options for each. The menu then allows the user to select the course of action they wish to take. The GUI interfaces with the Controller, which controls the flow of information between the GUI and the other components, including the different course, student, professor and TA objects that have been instantiated by the system. All information submitted by the user is relayed to the controller, and all resulting information is passed back from the controller to the GUI for presentation to the user.
- **Controller** – The controller maintains and directs the flow of information between the user's interaction with the GUI and the other components of the system. When the user enters their user ID and password, the controller accesses the database to confirm that the user ID exists and that its corresponding password matches that entered by the user. As the user selects menu options from the GUI, the controller accesses and updates information from the Course Catalog, any student, course, Professor, or TA objects, the scheduling engine, and the database.

- **Course Catalog** – The Course Catalog contains the complete listing of courses offered by the university. It includes information about their enrollment caps, the semesters in which each course is offered, and any pre-requisite courses that must be taken beforehand. The Course Catalog is loaded from the database when the system is initiated and course objects are instantiated from the Course class. Any changes made to a course's information by administrators are saved back to the database.
- **Scheduling Engine** – The Scheduling Engine is used to calculate the optimal schedule for students based on their course requests and priorities. It incorporates information about the number of other students who are enrolled in or requesting enrollment in a course as well as the availability of Professors and TA's to teach the course. The scheduling engine is accessed when a student has finished adding courses to their course request list and chooses to see their recommended schedule. The engine may be accessed multiple times if a student submits different sets of courses when performing a "what-if" analysis to determine their preferred set of courses to take for the next semester.
- **Database** – The database permanently stores information including the Course Catalog and information about all available courses, including the students enrolled in the course, and the Professors and TA's assigned to that course. When the system is started, the Course Catalog is loaded from the database, and when the user logs in, their user ID and password are checked against those stored in the database. Any changes made by an administrator to a course in the Course Catalog is saved to the database and, when a student accepts a course schedule from the system, their student ID is recorded in the database as being enrolled in the courses in their schedule. In addition, the database holds all historical data about course requests, allowing students to see their course

requests from past semesters, and allowing administrators to view the number of students enrolled in or requesting a particular course in past semesters.

Connectors

The main connectors for the system would link the GUI, the application controller, and the database. Since the GUI is web-based and accessible through a web browser, the GUI would communicate with the web server using HTTPS (a secure connection would be required since the system requires a log in). In the event that a large number of people are accessing the web server at once (as is likely to happen when the registration period first opens), multiple web servers can be used to distribute the user demand. The web servers would then connect to the Application Server, which would execute the application. The web servers would need to communicate through some sort of protocol, for example Apache JServ Protocol if the web servers are based on Apache HTTP Server and the application written in Java. The application controller would then use the Java Database Connectivity API to communicate with the system's database.



Architecture Style

Since the application is web-based, a traditional client-server architecture style would likely be best. The main advantage of the style would be the ability to have multiple web servers dividing the user load to prevent the system from crashing during periods of high demand, while additional servers may be added if needed. In addition, users' ability to access the system would be dependent on their own Internet connections, meaning that the system could be accessed from anywhere so long as Internet access is available. Also, the client-server architecture would keep the main database centralized, allowing for routine backups and improving security. The main disadvantages to a client-server model are the cost and human-resources needed to maintain the server.

GUI

The GUI will initially show the log-in screen to all users who access the system. It will have two options: one for logging in as a student and one for logging in as an administrator. Each option will prompt the user for a user ID and password. After a successful log-in, the user will be taken to the main menu for either students or administrators.

The student menu will have a series of buttons labeled as "View Course Information," "View Past Course Requests" and "Request Courses." If the user requests to view course information, they will be given a list of available courses to select from. If they choose to view past course requests, they will be given a list of all past semesters for which they were enrolled. If they choose to request courses, they will also be given a list of available courses, but, after requesting all their courses, they will be prompted to calculate their optimal schedule. This will prompt the scheduling engine to determine the recommended schedule based on the student's course requests. The GUI will then display the optimal schedule and prompt the user to confirm that they wish to enroll in the classes shown on that schedule.

The administrator menu will show a series of buttons labeled "View/Change Enrollment Cap," "Update Course Catalog" and "Professors/TAs". Both options to view or change enrollment caps or to update the course catalog will display a list of all available courses. When the user selects a course, the information for that course will be displayed in a series of textboxes or drop-down menus as is needed. The administrator will have the ability to change the information and will have a "save" or "cancel" button. If the user asks to view Professors and TA's, a list of all Professors and TA's will be displayed. When one is selected, that Professor or TA's availability, current course assignments, and teaching competencies will be displayed. The user will then have the ability to change the information pertaining to the Professor or TA and either save or cancel the changes.

Risk Mitigation

One of the most significant risks posed by such a system is that it may simply not be able to support a large number of users, especially at peak-demand periods. If the system were to crash during a registration period, it could leave students unable to register for the courses they need. It would also leave administrators unable to make any necessary changes such as adding new courses, or assigning Professors and TAs. This risk is mitigated, however, by the use of a scalable client-server model in which multiple web servers are used and user demand is divided between them.

Another significant risk would be a loss of the main database that stores the course information and records which students are enrolled in which courses. If the database server were to fail, or if the data were to be corrupted in some manner, the ability to plan course schedules and maintain records of enrollment could be lost. To prevent this, the system's database should be backed up regularly, preferably every day. In addition, a second cloned database could be maintained to provide redundancy in the event of data loss or corruption.

A third possibility concerns the ability of the system to properly recommend optimal schedules. The scheduling engine will need to be configured so that the recommended schedules strike a balance between course availability, the number of students enrolled in the course, and the availability of Professors and TA's with the proper teaching competencies to teach the courses. If the engine is not properly configured, situations might arise in which there are too many or too few students registering for a course, or there are too many or too few TA's with proper teaching competencies available for a course. To guard against this, it is possible that certain courses may be offered only if there is sufficient student demand as well as a sufficient number of TA's available.