Deliverable 3

SRS Document

Elite 4

Our project is to create a class scheduling application using Java, SQL, and PhpMyAdmin.

Project Goals/Main Functionality:

Add class:

Students can add classes into their schedules.

Drop class:

Students can drop a class from the schedule.

View Schedule:

Students will be able to see their schedule similar to how it appears on MyRIC

Pre-requisite courses met:

The application will use logic to make sure that students have taken required prerequisite classes.

Dates and Times:

Dates and Times of course will be visible to student.

Handle Scheduling Conflicts:

Program will handle students who make conflicting schedule additions.

Extra add Ons (If time allows)

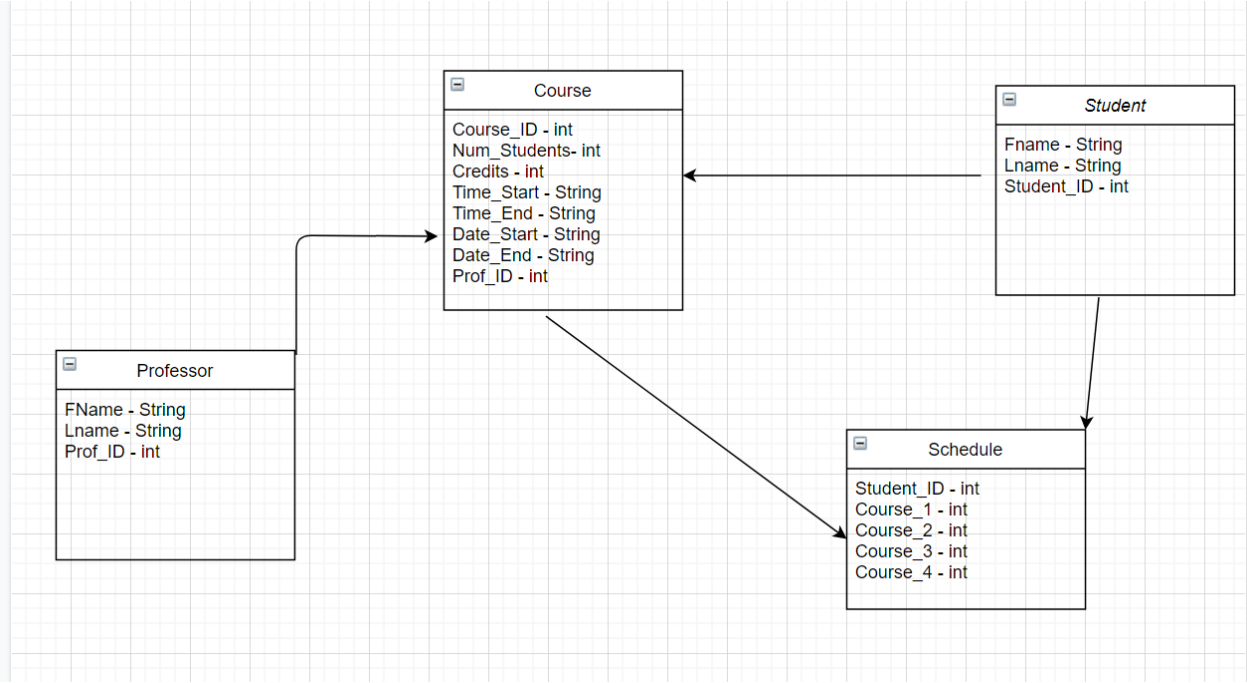
Credits:

Students will have a credit count and each course will have credits just like RIC

Built in advisor:

Recommended classes will appear for major requirements.

Class Diagram



Four Main Classes

Student:

- Fname || Stores students first name
- Lname || Stores students last name
- Student_ID || Primary Key -Stores the students unique ID Number

Professor:

- Fname || Stores professors first name
- Lname || Stores professor's last name
- Prof_ID || Primary Key - Stores the professors unique ID Number.

Course:

- Course_ID || Primary Key - Stores the courses unique ID Number.
- Num_Students || Current number of students in course
- Credits || Stores the numbers of credits the course awards.
- Time Start || Stores the start time in Hour: Min format
- Time End || Stores the end time in Hour: Min format
- Date_Start || Stores the date that the course starts.
- Date_End || Stores the date the course ends.
- Prof_ID || Stores the professor currently teaching the course.

Schedule:

- Student_ID ||Primary Foreign Key Stores the students unique ID Number
- Course_1 || Stores students' course in schedule
- Course_2 || Stores students' course in schedule
- Course_3 || Stores students' course in schedule
- Course_4 || Stores students' course in schedule

Informal Scenario 1: RIC student tries to add a class that is full.

Current State: Student logs in and is looking for a class to add. The list of classes is displayed to the student.

Informal Scenario: Student selects the desired class and clicks the add button. Student does not notice that the class is full capacity.

Next Scenario: A message pops up notifying the student that the class is full. He will be able to choose a different class to add.

Informal Scenario 2: Time conflict

Current State: The student has selected a class to add to their schedule.

Informal Scenario: The student attempts to add the class to their schedule, however there is a scheduling conflict. with a class already added to their schedule, they cannot attend two classes at the same time.

Next Scenario: An error message is displayed, indicating that there is a scheduling conflict between the class they are adding and the other conflicting class. For the student to add this class, they will have to make sure there are no scheduling conflicts with other classes.

Use Case 1: Student logging in.

Precondition: Student starts program.

Main Flow of Events: Use case starts when a student attempts to login, they input a username and password. The system then checks if the username and password are both valid. If both credentials are valid, they are allowed into the application/program, thus ending this use case.

Exceptional Flow of Events: If the student enters an invalid username. The system prompts: invalid credentials and does not allow the student to login, thus ending this use case.

Exceptional Flow of Events: If the student enters a valid username but an invalid password, the system prompts: Invalid password and does not allow login, and thus ending this use case.

Use Case 2: Student adding class.

Precondition: Student has logged in successfully. Student applies desired filters to available classes.

Main Flow of Events: Use case starts when student selects class from the filtered list. Student attempts to add desired class by clicking the add button. The desired class is added successfully to the student's schedule, thus ending this use case.

Exceptional Flow of Events: An error message is displayed telling the student that the class is already at full capacity.

Exceptional Flow of Events: An error message displays telling the user that the class that they chosen conflicts with another class they have.

Exceptional Flow of Events: An error message displays telling the user that they do not meet the requirements to take that class.

Use Case 3: Student dropping a class.

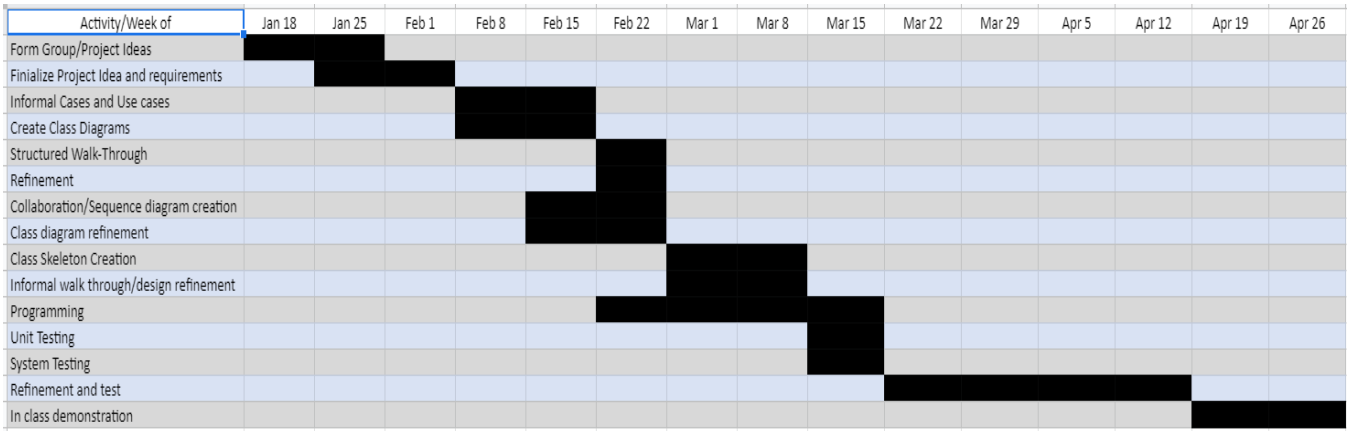
Precondition: Student has logged in successfully.

Main Flow of Events: User wants to remove a class from their schedule, they navigate to the Drop class section and selects the class they want to drop, the system confirms that they would like to drop a class. The student confirms and the class is removed from the schedule, thus ending the use case.

Exceptional Flow of Events: Student tries to remove a class, and there is no class in the schedule, thus ending the use case.

Exceptional Flow of Events: Student cancels the request to drop the class, thus ending this use case.

Time Frame.


Activity/Week of
Jan 18
Jan 25
Feb 1
Feb 8
Feb 15
Feb 22
Mar 1
Mar 8
Mar 15
Mar 22
Mar 29
Apr 5
Apr 12
Apr 19
Apr 26
Form Group/Project Ideas
Finialize Project Idea and requirements
Informal Cases and Use cases
Create Class Diagrams
Structured Walk-Through
Refinement
Collaboration/Sequence diagram creation
Class diagram refinement
Class Skeleton Creation
Informal walk through/design refinement
Programming
Unit Testing
System Testing
Refinement and test
In class demonstration