Concordia University

**Department of Computer Science**

**and Software Engineering**

**Software Process**

**SOEN 341/4 H --- Fall 2016 --- Section H**

**Project Scope and Plan Document**

|  |  |
| --- | --- |
| **Team members information** | |
| **Name** | **SID** |
| **Ayan Bikalapov** | **27028741** |
| **Alexandre Pelletier** | **27259492** |
| **Shenghong Yan** | **40021616** |
| **Claudia Chaillan** | **40037878** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Grading Sheet**

|  |  |  |
| --- | --- | --- |
| **Section** | **Evaluation criteria (see instructions in the template for details)** | **Grading** |
| all | 10 marks are allocated for excellence, professionalism and quality of work above and beyond the correct meeting of specifications.. | /10 |
| 1 | Completeness and clarity of the presentation of this document | /1 |
| 2 | Completeness and accuracy with regard to initial project description | /1 |
| 3.1 .  .  3.2  3.3 | Completeness and accuracy of the project functional requirements expressed as formal use cases, including difficulty and importance indicators  completeness and accuracy of the diagram and description of the domain model  completeness and accuracy with regard to initial project description accuracy with regard to initial project description, difficulty and importance ratings | /14  .  /2 .  /1 |
| 4.1  4.2 | Description of all team members’ capacities and schedule restrictions  Description of technical resources. | /1  /1. |
| 5 | List of goals removed from the project.  For each goal removed, give justifications in light of the resources available | /1 |
| 6.1 .  6.2 | Clarity of textual description, validity of rationale, clarity and appropriateness of diagram, list of modules responsibilities  List of technologies used, validity of rationale | /2 .  /1 |
| 7.1 .  7.2 .  7.3 .  7.4  7.5  7.6 | Completeness of list of activities, clarity of their stated purpose, as well as statement of what artifacts they are producing  Completeness of list of artifacts to be produced during the project, validity of roles description of each artifact  Cost estimation of each individual artifact, validity of explanation of cost estimation, total cost estimate  Mapping of activities to individual project members  Accurate and complete presentation of milestones  *Assessment of risks `* | /1 .  /2 .  /3 .  /1  /1  /1 |
| 8 | Early Prototyping | /3 |
| 9 | Management | /3 |
| Total |  | /50 |

**DO NOT REMOVE THIS PAGE WHEN SUBMITTING YOUR DOCUMENT**

# *Table of Contents*

[*Table of Contents* 3](#_Toc463980892)

[1. Presentation 3](#_Toc463980893)

[2. Project Description 4](#_Toc463980894)

[3. Goals and Constraints 6](#_Toc463980895)

[3.1 Functional Requirements 6](#_Toc463980896)

[3.2 Domain Model 18](#_Toc463980897)

[3.3 Constraints and Qualities 20](#_Toc463980898)

[**3.1.1** **Requirements and Performance** 20](#_Toc463980899)

[**3.1.2** **Usability and Compatibility** 20](#_Toc463980900)

[**3.1.3** **Reliability and Scalability** 20](#_Toc463980901)

[**3.1.4** **Interoperability and Maintainability** 20](#_Toc463980902)

[**3.1.5** **Security and Session Retrieval** 21](#_Toc463980903)

[4. Resource Evaluation 21](#_Toc463980904)

[4.1 Human Resources 21](#_Toc463980905)

[4.2 Technical Resources 23](#_Toc463980906)

[5. Scoping 23](#_Toc463980907)

[5.1 Included features 24](#_Toc463980908)

[5.2 Absent features 24](#_Toc463980909)

[6. Solution Sketch 25](#_Toc463980910)

[6.1 Architecture 25](#_Toc463980911)

[6.2 Technologies in use 25](#_Toc463980912)

[7. Plan 26](#_Toc463980913)

[7.1 Activities 32](#_Toc463980914)

[7.2 Artifacts 32](#_Toc463980915)

[7.3 Project Estimates 32](#_Toc463980916)

[7.4 Activities Assignments 32](#_Toc463980917)

[7.5 Schedule 32](#_Toc463980918)

[7.6 Risk 34](#_Toc463980919)

[**7.6.1** **Time Constraint** 34](#_Toc463980920)

[**7.6.2** **Team Cohesion and Communication** 34](#_Toc463980921)

[**7.6.3** **Team Experience** 34](#_Toc463980922)

[**7.6.4** **Uncooperative Members** 34](#_Toc463980923)

[**7.6.5** **Poor Implementation** 34](#_Toc463980924)

[**7.6.6** **Unforeseen Risks** 35](#_Toc463980925)

[8. Prototyping 35](#_Toc463980926)

[9. Management 40](#_Toc463980927)

[9.1 Code management 40](#_Toc463980928)

[9.2 Team communication 40](#_Toc463980929)

[9.3 Ticketing System 41](#_Toc463980930)

[9.4 Milestones 41](#_Toc463980931)

# Presentation

As the application of various modern computer technologies surges, students’ ability to create their schedules online is a feature widely used in the various universities of the world. However, there still exist some obstacles for students when producing their course schedules through the university websites. Several key elements like prerequisites, personal constraints and time availabilities are expected to be kept in mind while the students navigate their university curriculums and plan for their schedules using outdated tools. Unfortunately, there are always certain conditions that are forgotten during the process, which contributes to the failure of the student’s course registration. That is very frustrating to the the students. From all this we can conclude that manual schedule building is kind of tedious and errors emerge easily. Our product, however, will benefit both administrators and students a lot by automating the process using an online registration service.

The goal of the project is to create a web-based application that will help the student manage their course schedule easily and quickly. Fate Generator is the application we have developed to facilitate each student’s personal course schedule. Throughout this document, we detail the software development process used to create the web application. The project is divided into five deliverables that will follow the construction of the application and the progress of the project as a whole.

# Project Description

Fate Generator is a web-based schedule generation system that can create schedules based on the preferences specified by a student. The Student user can access course descriptions and use the schedule generator. The generator uses Concordia University’s list of available classes and the undergraduate and the graduate calendar descriptions to create a complete schedule of classes for one or more semesters. The student can input the following preferences in the system: preference to take morning or evening classes, preferred teachers for a particular course, full time or part time availability… Once the student completes their preferences, the generator selects the best sections, courses and days from the database to propose a schedule. In addition to the student’s preferences the Fate Generator also takes into account the student’s current situation in the university, such as the program followed, the core courses and the failed courses, to adapt the schedule for a successful degree completion. Students are the primary users of the Fate Generator.

In addition, a Registrar user can also be created. The registrar possesses specific privileges, such as viewing the student’s background, adjusting the available courses and granting existing students course exemptions and equivalents. This system will be designed and implemented over the course of thirteen weeks and will be divided into five deliverables.

|  |  |
| --- | --- |
| **Deliverable 0**  System overview  Description | Brief Introduction  Team members  Domain Model |
| **Deliverable 1**  Requirements, scope and plan | Introduction  Project Description  Goals and Constraints  Human and Technical Resources  Scope and Plan  Prototyping  Management |
| **Deliverable 2** | Architecture and design |
| **Deliverable 3** | Implementation and test results |
| **Deliverable 4** | Final submission and demo |

# Goals and Constraints

## Functional Requirements

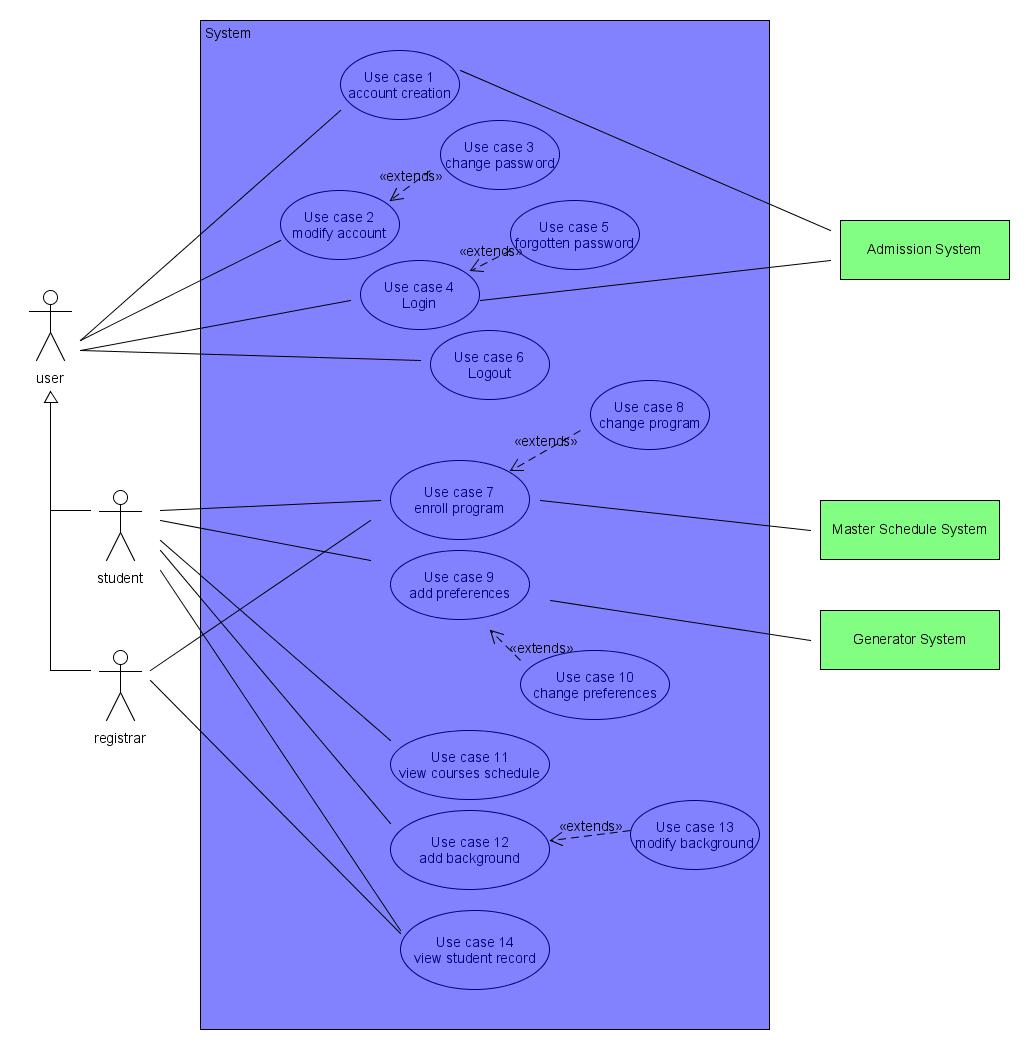


Figure 3.1: Use Case Diagram for the Users and the Student

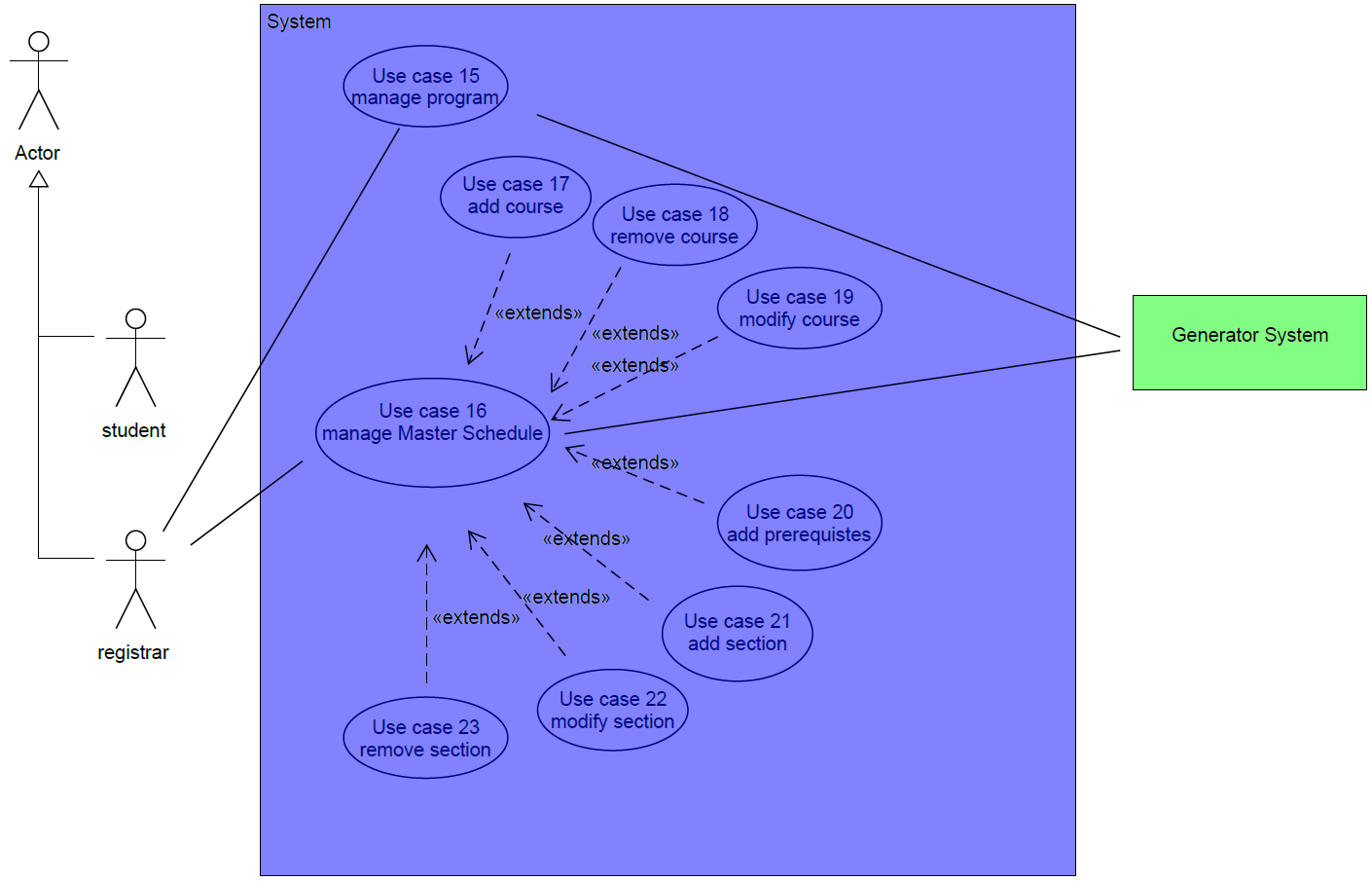


Figure 3.2: Use Case Diagram for the Registrar

|  |  |
| --- | --- |
| **Identifier/Name** | **UC1-Account Creation** |
| Importance | 5/5 |
| Difficulty | 1/5 |
| Actor(s) | Users (Student and Registrar) |
| Goal | To create an account for people who aren’t users. |
| Preconditions | The user is at the main page of schedule generator. |
| Related Use Case | N/A |
| System | Administration System |
| Summary | The new user needs to create an account for the first time to log in the schedule generator. |
| Basic flow | 1. The user selects the option to create a new account. 2. The system requests the user to provide personal information. 3. The user enters the information as well as set up a password. 4. The system asks for confirmation. 5. The user confirms the information. 6. The system directs the user to the main page. |
| Post-conditions | Success: The users have their own accounts and automated emails will be sent to the users’ email addresses for confirmation  Failure: The users fail to create accounts and an error message displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC2-Modify Account** |
| Importance | 1/5 |
| Difficulty | 2/5 |
| Actor(s) | Users (Student and Registrar) |
| Goal | The users can change their profile to match their new personal information. |
| Preconditions | The users are logged in  The user is on their profile page |
| Related Use Case | N/A |
| System | N/A |
| Summary | The users will modify their personal information. |
| Basic flow | 1. The users modify their personal information. 2. The systems asks for confirmation from the user. 3. The user confirms the modification 4. The system displays the final profile. |
| Post-conditions | Success: The user’s personal information is changed.  Failure: The users’ personal information remains unchanged and an error message displayed |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC3-Change Password** |
| Importance | 1/5 |
| Difficulty | 2/5 |
| Actor(s) | Users (Student and Registrar) |
| Goal | The user’s password is reset and emailed to them |
| Preconditions | The user is at the password resetting page |
| Related Use Case | UC2 |
| System | N/A |
| Summary | The user can request a password reset by providing their username to the system. The system will send an automated email to the user with instructions on how to reset the password. |
| Basic flow | 1. The user requests a password reset. 2. The system prompts for username and old password. 3. The user puts in the username and old password 4. The system prompts for new password. 5. The user puts in new password. 6. The system confirms the password change and directs users to main page. |
| Post-conditions | Success: Email is sent to user’s email address with message that they have successfully reset the password.  Failure: User is informed that their password reset request failed and the email is not sent. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC4-Login** |
| Importance | 5/5 |
| Difficulty | 1/5 |
| Actor(s) | Users (Student and Registrar) |
| Goal | To allow the user to access the system. |
| Preconditions | The user is not logged onto the system  The user is on the login page of the schedule generator |
| Related Use Case | N/A |
| System | N/A |
| Summary | The system will validate the username and password and subsequently give them access to the website. |
| Basic flow | 1. The system prompts for username and password. 2. The user provides a valid username and password. 3. The system directs the user to the main page of the website. |
| Post-conditions | Success: The user is logged in and directed to the website’s home page.  Failure: The user remains on the login page (denied access) and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC5-Forgotten Password** |
| Importance | 1/5 |
| Difficulty | 4/5 |
| Actor(s) | Users (Student and registrar) |
| Goal | To allow the user to retrieve password. |
| Preconditions | The user is at the login page.  The user has left a valid contact information (email or phone number) previously. |
| Related Use Case | UC4 |
| System | Admission system |
| Summary | The user will retrieve the password through email with a link to the password retrieving page. |
| Basic flow | 1. The user activates the password retrieving feature. 2. The system prompts for an email address. 3. The user puts in the email address 4. The system sends an email. 5. The user receives an email with a link to the password retrieving page. 6. The user sets up for a new password and confirms it. 7. The system displays successful password reset |
| Post-conditions | Success: The user gets an email and retrieves his password.  Failure: The password hasn’t been retrieved and an error message is displayed. There is no message sent to the user’s email address. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC6-Logout** |
| Importance | 5/5 |
| Difficulty | 1/5 |
| Actor(s) | Users (Student and Registrar) |
| Goal | The user terminates the session. |
| Preconditions | The user is logged onto the system. |
| Related Use Case | N/A |
| System | N/A |
| Summary | The user logs out of the session by disconnecting them from the system. |
| Basic flow | 1. The user activates the logout feature. 2. The system acknowledges the logout request. 3. The system redirects the user to the login page. |
| Post-conditions | Success: The user is logged out.  Failure: The user remains logged in and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC7-Enroll Program** |
| Importance | 5/5 |
| Difficulty | 1/5 |
| Actor(s) | Student |
| Goal | The student can choose which program they want to study. |
| Preconditions | Students are logged onto the system.  Students haven’t chosen their program. |
| Related Use Case | UC8 |
| System | N/A |
| Summary | The student chooses their program from a list provided by the system. |
| Basic flow | 1. The system displays a list of programs. 2. The students chooses a program. 3. The system prompts for confirmation. 4. The student confirms the request. 5. The system indicates that a choice has been successfully made. |
| Post-conditions | Success: The student is enrolled in their program.  Failure: The students remains on the enroll page and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC8-Change Program** |
| Importance | 5/5 |
| Difficulty | 1/5 |
| Actor(s) | Student |
| Goal | The students can change their program |
| Preconditions | The student is logged onto the system.  The student is enrolled in a program. |
| Related Use Case | UC7 |
| System | N/A |
| Summary | The student can choose a different program from the list provided. |
| Basic flow | 1. The student requests to change their program. 2. The system prompts for confirmation. 3. The student confirms. 4. The system displays the list of programs. 5. The student chooses a program. 6. The system prompts for confirmation. 7. The student confirms. 8. The system displays that a choice has been successfully made. |
| Post-conditions | Success: The student’s program is changed.  Failure: The student keeps the original program and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC9-Add Preferences** |
| Importance | 5/5 |
| Difficulty | 1/5 |
| Actor(s) | Student |
| Goal | The student can add their preferences. |
| Preconditions | The student is logged onto the system.  The student is enrolled in a program. |
| Related Use Case | UC10 |
| System | N/A |
| Summary | The student can choose their own preferences (number of credits, sections preferred, morning/afternoon courses…) |
| Basic flow | 1. The system displays a list of preferences. 2. The students fills in his preferences. 3. The systems prompts for confirmation 4. The student confirms the request. 5. The system indicate that preferences have been successfully made. |
| Post-conditions | Success: The student’s preferences are added.  Failure: The student remains on the add preferences page and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC10-Change Preferences** |
| Importance | 3/5 |
| Difficulty | 1/5 |
| Actor(s) | Student |
| Goal | The student can change their preferences. |
| Preconditions | The student is logged onto the system.  The student is enrolled a program.  The student has added his preferences. |
| Related Use Case | UC9 |
| System | N/A |
| Summary | The student can renew their own preferences when needed. |
| Basic flow | 1. The student requests to change his preferences. 2. The system prompts for confirmation. 3. The student confirms. 4. The system displays the list of preferences. 5. The student fills in his preferences. 6. The system prompts for confirmation. 7. The student confirms. 8. The system indicates that changes to the preferences have been successfully made. |
| Post-conditions | Success: The student’s preferences are changed.  Failure: The student’s preferences remain the same and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC11-View Course Schedule** |
| Importance | 5/5 |
| Difficulty | 1/5 |
| Actor(s) | Student |
| Goal | The student can view the course schedule provided by the system. |
| Preconditions | The student is logged onto the system.  The student is enrolled in a program.  The student has added his preferences. |
| Related Use Case | N/A |
| System | N/A |
| Summary | The student can check the result generated by the system. |
| Basic flow | 1. The student makes a request to view the course schedule. 2. The system displays the generated courses schedule |
| Post-conditions | Success: The student can view the courses schedule.  Failure: The student remains on the main page and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC12-Add Background** |
| Importance | 3/5 |
| Difficulty | 1/5 |
| Actor(s) | Student |
| Goal | The student can add his educational background. |
| Preconditions | The student is logged onto the system.  The students is enrolled in a program. |
| Related Use Case | UC13 |
| System | N/A |
| Summary | The student can add his educational background (credits, courses taken…) |
| Basic flow | 1. The system displays a list of questions. 2. The student fills in his answers. 3. The system prompts for confirmation. 4. The student confirms. 5. The system indicated that background has been successfully added. |
| Post-conditions | Success: The student adds his background.  Failure: The student remains on the add background page and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC13-Modify background** |
| Importance | 3/5 |
| Difficulty | 1/5 |
| Actor(s) | Student |
| Goal | The student can change their educational background. |
| Preconditions | The student is logged onto the system.  The student is enrolled in a program.  The student has added their background. |
| Related Use Case | UC12 |
| System | N/A |
| Summary | The student can upload and renew his educational background. |
| Basic flow | 1. The student requests to change their background. 2. The system prompts for confirmation. 3. The student confirms 4. The system displays a list of questions. 5. The students fill in his answers. 6. The system prompts for confirmation. 7. The student confirms. 8. The system indicates that background changes have been successfully made. |
| Post-conditions | Success: The student’s background is changed.  Failure: The student’s background remains the same and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC14-View Student Record** |
| Importance | 5/5 |
| Difficulty | 1/5 |
| Actor(s) | Registrar |
| Goal | The registrar can view the student record. |
| Preconditions | The registrar is logged onto the system.  The student is enrolled in a program.  The student has added his preferences.  The student has added his background. |
| Related Use Case | N/A |
| System | N/A |
| Summary | The registrar can check the background uploaded by the student. |
| Basic flow | 1. The registrar requests to view the student background. 2. The system displays the student’s background. |
| Post-conditions | Success: The registrar views the student background.  Failure: The registrar remains on the main page and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC15-Manage program** |
| Importance | 5/5 |
| Difficulty | 3/5 |
| Actor(s) | Registrar |
| Goal | The registrar can manage the programs. |
| Preconditions | The registrar is logged onto the system.  Different programs exist. |
| Related Use Case | N/A |
| System | Generator System |
| Summary | The registrar can make changes to the different programs available. |
| Basic flow | 1. The registrar requests to manage the programs.  2. The system displays the different programs available.  3. The registrar makes the changes the wants on the programs.  4. The system asks for confirmation for the changes made.  5. The registrar confirms the changes.  6. The registrar requests for the updated program. |
| Post-conditions | Success: The registrar makes changes to the program.  Failure: The program is not changed and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC16-Manage Master Schedule** |
| Importance | 5/5 |
| Difficulty | 3/5 |
| Actor(s) | Registrar |
| Goal | The registrar can manage the Master Schedule. |
| Preconditions | The registrar is logged onto the system.  The Master Schedule exists.  The courses and sections are added to the Master Schedule. |
| Related Use Case | N/A |
| System | Generator System |
| Summary | The registrar can manage the master schedule by adding, modifying or removing courses and sections. |
| Basic flow | 1. The registrar requests to manage the Master Schedule.  2. The system displays the Master Schedule.  3. The registrar makes the changes he wants on the master schedule.  4. The system asks for confirmation for the changes made.  5. The registrar confirms the changes.  6. The registrar requests for the updated Master Schedule. |
| Post-conditions | Success: The registrar makes changes to the Master Schedule.  Failure: The Master Schedule is not changed and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC17-Add course** |
| Importance | 4/5 |
| Difficulty | 2/5 |
| Actor(s) | Registrar |
| Goal | The registrar can add courses to the Master Schedule. |
| Preconditions | The registrar is logged onto the system.  The Master Schedule exists. |
| Related Use Case | UC16 |
| System | N/A |
| Summary | The registrar wants to add a course to the Master Schedule. |
| Basic flow | 1. The registrar requests to add courses to the Master Schedule.  2. The system displays the courses available to be added.  3. The registrar selects the course to add.  4. The system asks for confirmation to add the course.  5. The registrar confirms the addition of that course to the schedule.  6. The registrar requests for the updated Master Schedule. |
| Post-conditions | Success: The course is added to the Master Schedule.  Failure: The course isn’t added to the Master Schedule and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC18-Remove course** |
| Importance | 4/5 |
| Difficulty | 4/5 |
| Actor(s) | Registrar |
| Goal | The registrar can remove a course from the Master Schedule. |
| Preconditions | The registrar is logged onto the system.  The course exists in the Master Schedule. |
| Related Use Case | UC16 |
| System | N/A |
| Summary | The registrar wants to remove a course from the Master Schedule. |
| Basic flow | 1. The registrar requests to remove courses from the Master Schedule.  2. The system displays the courses on the Master Schedule.  3. The registrar selects the course to remove from the schedule.  4. The system asks for confirmation to remove the course.  5. The registrar confirms the removal of the course from the schedule.  6. The registrar requests for the updated Master schedule. |
| Post-conditions | Success: The course is removed from the Master Schedule.  Failure: The course isn’t removed from the Master Schedule and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC19-Modify Course** |
| Importance | 4/5 |
| Difficulty | 1/5 |
| Actor(s) | Registrar |
| Goal | The registrar can modify a course. |
| Preconditions | The registrar is logged onto the system.  The Master Schedule exits.  The course has been added on the Master Schedule |
| Related Use Case | UC17, UC18 |
| System | Generator System |
| Summary | The registrar wants to modify the course on the Master Schedule based on the student program, and the course is modified completely. |
| Basic Flow | 1. The registrar requests to modify course on the Master Schedule. 2. The system asks for confirmation. 3. The registrar confirms the course modification information. 4. The system indicates that the course has been successfully modified. |
| Post-conditions | Success: The new course information updated on the Master Schedule.  Failure: The course information is not updated and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC20-Add prerequisites** |
| Importance | 3/5 |
| Difficulty | 3/5 |
| Actor(s) | Registrar |
| Goal | The register can add prerequisites based on the student program. |
| Preconditions | The registrar is logged onto the system.  The student’s background is uploaded on the system, and the registrar can view the record.  The Master Schedule exists.  The core courses are been added into Master Schedule. |
| Related Use Case | UC12, UC13, UC14, UC17 |
| System | Generator System |
| Summary | After viewing the student record, the register can choose to add prerequisites to the Master Schedule. |
| Basic flow | 1. The registrar requests to add prerequisites on the Master Schedule. 2. The system asks for the information to add. 3. The registrar puts in prerequisites. 4. The system asks for confirmation. 5. The registrar confirms the prerequisites information. 6. The system displays the prerequisite information. |
| Post-conditions | Success: The prerequisite are added and the student can see the prerequisites based on background and core courses.  Failure: The prerequisite are not added and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC21-Add Section** |
| Importance | 2/5 |
| Difficulty | 2/5 |
| Actor(s) | Registrar |
| Goal | The registrar wants to add section for a course. |
| Preconditions | The registrar is logged onto the system.  The Master Schedule exists. |
| Related Use Case | N/A |
| System | N/A |
| Summary | The registrar wants to add a section to the course. |
| Basic flow | 1. The registrar requests to add section to a course. 2. The system displays a list of courses. 3. The registrar selects the course to add section. 4. The system asks for confirmation 5. The registrar confirms the addition of a section. 6. The system displays the new section added successfully for a course. |
| Post-conditions | Success: A new section is added to the course on the Master Schedule.  Failure: The section is not added and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC22-Modify Section** |
| Importance | 2/5 |
| Difficulty | 4/5 |
| Actor(s) | Registrar |
| Goal | The registrar can modify the section of a course |
| Preconditions | The registrar is logged onto the system.  The Master Schedule exists.  A section for the course exists. |
| Related Use Case | UC21 |
| System | Generator System |
| Summary | The registrar wants to modify the section of a course |
| Basic flow | 1. The registrar requests to modify course section. 2. The system displays a list of sections. 3. The registrar selects the section to modify and changes the information. 4. The system asks for confirmation of the modification. 5. The registrar confirms the modifications. 6. The system displays the modified information of the section. |
| Post-conditions | Success: The section information is modified.  Failure: The section information is not modified and an error message is displayed. |

|  |  |
| --- | --- |
| **Identifier/Name** | **UC23-Remove Section** |
| Importance | 2/5 |
| Difficulty | 1/5 |
| Actor(s) | Registrar |
| Goal | The registrar can remove section of a course. |
| Preconditions | The registrar id logged onto the system.  The Master Schedule.  The section for the course exists. |
| Related Use Case | UC21, UC22 |
| System | Generator System |
| Summary | The registrar wants to remove the section of a course |
| Basic Flow | 1. The registrar requests to remove course section. 2. The system displays a list of sections. 3. The registrar selects the section to remove. 4. The system asks for confirmation for the removal. 5. The registrar confirms the removal. 6. The system displays the updated version of the Master Schedule. 7. The Master System sends a message that the registrar confirms to remove section of a course. |
| Post-conditions | Success: The section is removed from the Master Schedule.  Failure: The section is not removed from the Master Schedule and an error message is displayed. |

## 3.2 Domain Model

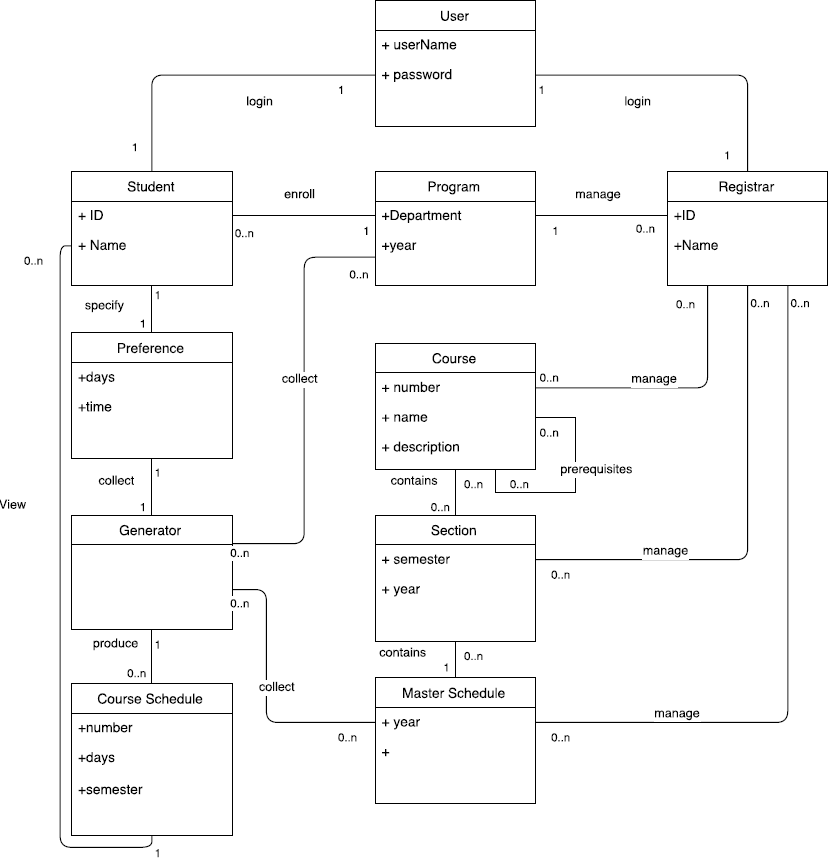


Figure 3.3: Domain Model Diagram

This domain model represents the Fate Generator system overview, incorporating the following domain level objects (DLO) and their relations:

User: the core DLO that is extended to a Student or Registrar DLOs, the starting point of the system access. Requires registration with username and password.

Student: a type of User DLO that requires additional details such as Student ID and the first and last names. Student DLO has relationship with various other DLOs in the system.

Registrar: another type of User DLO. This type requires employee ID of the registrar person, as well as first and last names. The validity of the ID can be potentially checked against a database of valid employee IDs, subject to change.

Program: DLO that stores Program requirements such as required courses, required number of credits, etc. This DLO is used by Generator to determine what courses a student needs to complete.

Preference: DLO that stores student preferences to be used by the Generator

Course: DLO for all available courses in the University that contains the essential information about the course

Section: related to the Course DLO as it stores the information on the sections of a particular course that are available

Master Schedule: a list of Sections available for the calendar year, main reference point of the Generator DLO

Generator: core DLO of the system, collects information about the students and the courses to generate a Course Schedule

Course Schedule: output DLO that is the final result of a Generator. Many schedules can be created with one Generator.

## 3.3 Constraints and Qualities

### **Requirements and Performance**

Fate Generator is a web application, and as such, a server is required to host its domain. To provide optimal performance it is required to be up at all times and handle requests at sufficient speeds. A system with standard hardware, protection against DDOS attacks and efficiently written code should suffice.

### **Usability and Compatibility**

Fate Generator has been implemented by prospective users and therefore the design of its interface has been developed with the user experience in mind. It is easy to navigate and understand the flow of the application and has been designed according to industry standards, including norm features of a website. The web application is also compatible and operable on all systems and web browsers (including Internet Explorer).

### **Reliability and Scalability**

Fate Generator has been tested throughout its implementation and has undergone acceptance testing to ensure that the generated schedule takes into account all the necessary user and course information providing reliable data. It should be able to support increasing amounts of traffic without decreasing response time when handling requests. If this traffic exceeds a threshold users will be pushed into a queue in order to log in.

### **Interoperability and Maintainability**

Fate Generator has been designed to generate a suggested schedule by gathering necessary data from the university website itself therefore ensuring that the information is relevant. A team of developers will maintain the application on a yearly basis to ensure it coheres/adapts to specifications and requirements.

### **Security and Session Retrieval**

Fate Generator users' passwords will be encrypted to prevent/dissuade hackers from accessing accounts they do not own. After being idle for a period of ten minutes a session time out will occur to prevent unwanted access by individuals on the same system. The web application will store information from the last session of the user and up to five generated schedules.

# Resource Evaluation

## Human Resources

|  |  |
| --- | --- |
| **Name** | **Ayan Bikalapov** |
| Skill | Java, Perl, Assembly, C |
| Experience | QA for Testing Automation tools at Nuance INC |
| Role | Team Leader |
| Availability | 6 hours/week |

|  |  |
| --- | --- |
| **Name** | **Yefei Xue** |
| Skill | PHP, CSS, Java, JavaScript, HTML, Ruby |
| Experience | Web development project from SOEN 287 class |
| Role | Mix |
| Availability | 6 hours/week |

|  |  |
| --- | --- |
| **Name** | **Alexandre Pelletier** |
| Skill | Front-end and Back-end development, database management |
| Experience | Junior Developer at Skyfold Inc. |
| Role | Developer Leader, Repository/Server maintainer, Back-end developer |
| Availability | 6 hours/week |

|  |  |
| --- | --- |
| **Name** | **Yu Luo** |
| Skill | C++, Java, Thinkphp, background programming, web development |
| Experience | Sotware engineer at Tecent Inc. |
| Role | Coder |
| Availability | 6 hours/week |

|  |  |
| --- | --- |
| **Name** | **Yongann Khoo** |
| Skill | HTML, PHP, CSS, Java, JavaScript |
| Experience | Web design project from SOEN 287 |
| Role | Developer |
| Availability | 12 hours/week |

|  |  |
| --- | --- |
| **Name** | **Claudia Chaillan** |
| Skill | C, C++ |
| Experience | Team projects |
| Role | Documentation |
| Availability | 6 hours/week |

|  |  |
| --- | --- |
| **Name** | **Dandan Zhang** |
| Skill | Java, Android, VB |
| Experience | Data Structure & Algorithm from SOEN 352 class (java experience and the algorithm application in programs) |
| Role | Developer, Documentation |
| Availability | 6 hours/week |

|  |  |
| --- | --- |
| **Name** | **Duo Liang** |
| Skill | Java, C++, Word, Excel, Powerpoint |
| Experience | A Stimulating System of Library based on DOS (A project from Data Structure Course) |
| Role | Documentation |
| Availability | 8 hours/week |

|  |  |
| --- | --- |
| **Name** | **Shenghong Yan** |
| Skill | UML, Java |
| Experience | Data Structure & Algorithm from SOEN 352 |
| Role | Documentation |
| Availability | 6 hours/week |

|  |  |
| --- | --- |
| **Name** | **Konstantinos Evagelidis** |
| Skill | Java, C++, HTML |
| Experience | Personal Projects |
| Role | Mix |
| Availability | 5 hours/week |

## Technical Resources

During the development stage of the project some form of technology will be used for any given task.

The simplest form of hardware necessary will be computers for the individual coders to implement and test features. A live environment will be necessary as well; thus a web server is needed to view the project from a user perspective.

As far as software is concerned, a simple text editor to edit source code and a browser to perform tests will be necessary.

|  |  |
| --- | --- |
| **Hardware** | Personal Computers, Concordia lab computers, web server (for live environment testing) |
| **Personal Computer Specifications** | Windows Operating System  At least 512MB of RAM |
| **Server Specifications** | Compatible with chosen back-end language (Node.js)  512MB of RAM  1 CPU Share  Row limit of 10K for database  4 hours of downtime per month |
| **Software** | Text editors, web browsers (development will be focused primarily for Firefox and Chrome) |

# Scoping

Our web application will include a number of features. However, due to lack of time and technical constraints, some of these features will be omitted.

## Included features

Some components of high importance are necessary to maintain the application’s basic function. Listed below are these features with according to their level of importance in each category:

|  |  |  |
| --- | --- | --- |
|  | **Features** | **Use case(s)** |
| Core component | Login/logout | UC4, UC6 |
| Enroll/change program | UC7,UC8 |
| View course schedule | UC11 |
| Account creation | Account creation | UC1 |
| Academic record | View student record | UC14 |
| Course planner | Add preference | UC9 |
| Manage program | UC15 |
| Manage master schedule | UC16 |
| Course modification | Add/ Modify background | UC12, UC13 |
| Change preference | UC10 |
| Add/remove/modify course | UC17, UC18, UC19 |
| Add prerequisites | UC20 |
| Add/modify/remove section | UC21, UC22, UC23 |
| Other features | Modify account | UC2 |
| Change password/ forgotten password | UC3, UC5 |

As listed above in sequence, the **core component and the account creation** has the highest priority in terms of implementation. The course planner and course management are in second place in terms of priority to make the process of planning schedules more flexible. The course modifications and other features have the lowest priority as they are not essential for the implementation of the project.

## Absent features

Some features are unnecessary to achieve the goal of this project which is why they have been removed. Listed below are the absent features and their reasons:

Note: currently all features are considered to be within scope, however as the project advances, the de-scoped features will be represented in the table below.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

\*if time permits, some features could be implemented

# Solution Sketch

## Architecture

Our course scheduler, Fate Generator, is based on the Model View Controller (MVC) architecture. This type of architecture is widely used for web applications and is the best suited for the development phase of the project. It separates the work into three different classes, so that they can be worked on independently and simultaneously.

The Model will manage the data and logic of the application, by retrieving and storing information in the database. The View will display information from the model data. The controller will handle the user’s inputs and responses.

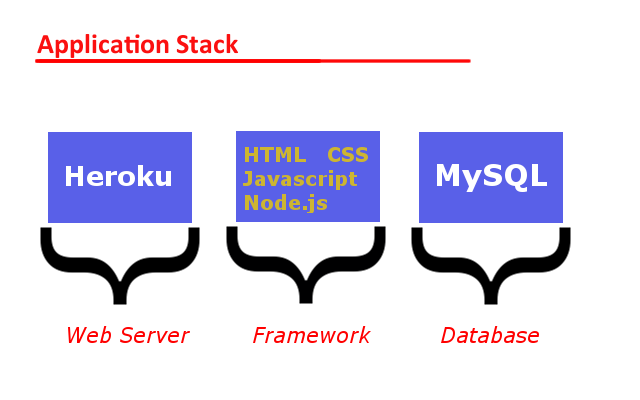


Figure 6.1: Representation of the application stack for the project

## Technologies in use

As defined in the application stack diagram – this project will be focused on combining three integral components. Namely: the web server which allows for a live environment and user experience, a framework of languages with which to implement the system, and a database to store user and course data.

* Web Server – Heroku
* Framework – HTML, CSS, Javascript, Node.js
* Database - mySQL

Other vital services that will be required are a VCS (version control system), repository, and TMS (task management system). Github provides all three of these services and is to be used to store the project’s source code and documents, provide developers with distributed environment to allow for parallel development, and an issue tracker to monitor the project’s work flow.

# Plan

|  |  |
| --- | --- |
| **Activity** | **System overview** |
| Purpose | To define the project as a whole, its purpose, its function. |
| Participants | Ayan, Alex, Claudia, Dandan, Kosta, Liang Duo, Shenghong Yan, Yu Luo, Ye Fei |
| Duration(Time period) | 7th-14th September |
| **Artifact** | **Description of project** |
| Description | A brief description of the system to be developed, how it will work and how it will be used. |
| Estimated hours | 1 |
| Estimated cost | 20$ |
| **Artifact** | **Domain model** |
| Description | Domain model of the system, with the principal entities and their relationships. |
| Estimated hours | 4 |
| Estimated cost | 80$ |

|  |  |
| --- | --- |
| **Activity** | **Role distribution** |
| Purpose | To ensure every team member has clear understanding of their own duty/ |
| Participants | Ayan, Alex, Claudia, Dandan, Kosta, Liang Duo, Shenghong Yan, Yu Luo, Ye Fei |
| Duration(Time period) | 7th-14th September |
| **Artifact** | **List of team members** |
| Description | A list with all team members’ contact information, including e-mail address and phone number. |
| Estimated hours | 0.5 |
| Estimated cost | 10$ |
| **Artifact** | **Role of each member** |
| Description | Dividing the team into Coding Sub-team and Documentary Sub-team, depending on each person’s intrest. |
| Estimated hours | 0.5 |
| Estimated cost | 10$ |

|  |  |
| --- | --- |
| **Activity** | **Define project** |
| Purpose | To understand the background and goals and to give a brief description of our project. |
| Participants | Ayan, Dandan, Shenghong Yan |
| Duration(Time period) | 14th-20th September |
| **Artifact** | **Introduction** |
| Description | Description of the current background, reasons for creating the application, final goals of the project. |
| Estimated hours | 1 |
| Estimated cost | 20$ |
| **Artifact** | **Project description** |
| Description | Brief introduction of what users can do in the application (and how different users achieve different functions), listing the specific content of each deliverable. |
| Estimated hours | 1 |
| Estimated cost | 20$ |

|  |  |
| --- | --- |
| **Activity** | **Goals and constraints** |
| Purpose | To list both the functional or non-functional features we want to achieve in the project. |
| Participants | Claudia, Dandan, Liang Duo, Shenghong Yan, Kosta |
| Duration(Time period) | 21th-27th September |
| **Artifact** | **Use Case diagrams & Use Cases** |
| Description | Drawing the Use Case diagrams, completing the Use Cases based on Use Case diagrams. |
| Estimated hours | 5 |
| Estimated cost | 100$ |
| **Artifact** | **Constraints and Qualities** |
| Description | To understand the limitations of the project, what risks we will meet, deciding on the implementation of qualities. |
| Estimated hours | 2 |
| Estimated cost | 40$ |

|  |  |
| --- | --- |
| **Activity** | **Resource evaluation** |
| Purpose | To collect all the resources we already have for further planning and estimation. |
| Participants | Alex, Shenghong Yan |
| Duration(Time period) | 14th-20th September |
| **Artifact** | **Human resources** |
| Description | A collection of information of each team member including name, experience, skills, role and available time per week. |
| Estimated hours | 1 |
| Estimated cost | 20$ |
| **Artifact** | **Technical resources** |
| Description | A list of both hardware and software resources going to be used. |
| Estimated hours | 1 |
| Estimated cost | 20$ |

|  |  |
| --- | --- |
| **Activity** | **Scoping and Prototyping** |
| Purpose | To readjust our scope and to make a preliminary prototype, depending on the resources and the cost of the project. |
| Participants | Dandan, |
| Duration(Time period) | 27th September - 10thOctober |
| **Artifact** | **Features** |
| Description | A list of features that we can or cannot achieve. |
| Estimated hours | 3 |
| Estimated cost | 60$ |
| **Artifact** | **Prototype** |
| Description | A prototype in an early phase, including the organization of database, the design of our UI… |
| Estimated hours | 15 |
| Estimated cost | 300$ |

|  |  |
| --- | --- |
| **Activity** | **Solution sketching** |
| Purpose | To decide what kind of solution we are going to apply to our project. |
| Participants | Alex, |
| Duration(Time period) | 20th-27th September |
| **Artifact** | **Technologies used** |
| Description | List all the specific technologies we will use in our project. |
| Estimated hours | 1 |
| Estimated cost | 20$ |

|  |  |
| --- | --- |
| **Activity** | **Plan** |
| Purpose | To make a periodic evaluation of our work. |
| Participants | Claudia, Kosta, Liang Duo |
| Duration(Time period) | 5th - 10thOctober |
| **Artifact** | **Activities artifacts estimates** |
| Description | A record of all the artifacts we are going to have over the course of the project and an estimation of time and cost. |
| Estimated hours | 2 |
| Estimated cost | 40$ |
| **Artifact** | **Schedule** |
| Description | A table which indicates our process in the project and to record the milestones. |
| Estimated hours | 2 |
| Estimated cost | 40$ |
| **Artifact** | **Risks** |
| Description | An estimation of risks that we have or will encounter in our project. |
| Estimated hours | 2 |
| Estimated cost | 40$ |

|  |  |
| --- | --- |
| **Activity** | **Define Architectural Design** |
| Purpose | To define the high-level architecture of the system like design, features implemented and interactions. |
| Participants | Alex |
| Duration(Time period) | 3rd October – 10th October |
| **Artifact** | **Architecture design** |
| Description | A diagram of high-level architecture and an explanation of why we chose this solution. |
| Estimated hours | 5 |
| Estimated cost | 100$ |
| **Artifact** | **Class Diagram** |
| Description | A diagram showing the connection between different objects and the attributes and methods they provide. |
| Estimated hours | 3 |
| Estimated cost | 60$ |

|  |  |
| --- | --- |
| **Activity** | **Define Subsystems Interfaces** |
| Purpose | To define the interactions between different components of the system. |
| Participants | Alex, Yu Luo |
| Duration(Time period) | 17th October – 24th October |
| **Artifact** | **Module Interface Specifications** |
| Description | The specifications of how modules will cooperate with one another |
| Estimated hours | 6 |
| Estimated cost | 120$ |
| **Artifact** | **Unit Descriptions** |
| Description | A description of each class of subsystem. |
| Estimated hours | 2 |
| Estimated cost | 40$ |

|  |  |
| --- | --- |
| **Activity** | **Implementation of requirements** |
| Purpose | Implementation of requirements defined for the project. |
| Participants | Alex, Yu Luo, Alvis, Ye Fei |
| Duration(Time period) | 25th October – 1st November |
| **Artifact** | **Configured Server, Database** |
| Description | Implemented database which will contain information about the courses, the professors and the students. |
| Estimated hours | 20 |
| Estimated cost | 400$ |
| **Artifact** | **Implement View** |
| Description | Implemented design of the project like the user interface. |
| Estimated hours | 20 |
| Estimated cost | 400$ |
| **Artifact** | **Implement Model** |
| Description | Implemented model of the system like the different functions. |
| Estimated hours | 15 |
| Estimated cost | 300$ |

|  |  |
| --- | --- |
| **Activity** | **Define Dynamic Design Scenarios** |
| Purpose | To simulate potential scenarios our system would encounter. |
| Participants | Yu Luo, Alvis |
| Duration(Time period) | 17th October – 24th October |
| **Artifact** | **System Sequence Diagrams** |
| Description | A diagram indicating the interactions between a user and the system when an event is raised. |
| Estimated hours | 6 |
| Estimated cost | 120$ |

|  |  |
| --- | --- |
| **Activity** | **Develop Unit tests** |
| Purpose | To develop Unit test that will verify the functionality of each code. |
| Participants | Alvis, Yu Luo |
| Duration(Time period) | 2nd November – 9th November |
| **Artifact** | **Tested items** |
| Description | Test cases for the Unit tests. |
| Estimated hours | 20 |
| Estimated cost | 400$ |
| **Artifact** | **Untested items** |
| Description | Untested cases for the Unit tests. |
| Estimated hours | - |
| Estimated cost | - |

|  |  |
| --- | --- |
| **Activity** | **Develop Requirements tests** |
| Purpose | To develop tests on the Requirements to verify the features. |
| Participants | Donna, Lucas |
| Duration(Time period) | 2nd November – 9th November |
| **Artifact** | **Tested items** |
| Description | Test cases for the Requirements tests. |
| Estimated hours | 5 |
| Estimated cost | 100$ |
| **Artifact** | **Untested items** |
| Description | Untested cases for the Requirements tests. |
| Estimated hours | - |
| Estimated cost | - |

|  |  |
| --- | --- |
| **Activity** | **Develop Stress Tests** |
| Purpose | To develop Stress tests that verify extreme situations of system usage. |
| Participants | Ye Fei |
| Duration(Time period) | 10th November – 14th November |
| **Artifact** | **Tested items** |
| Description | Test cases for the Stress tests. |
| Estimated hours | 8 |
| Estimated cost | 160$ |
| **Artifact** | **Untested items** |
| Description | Untested cases for the Stress tests. |
| Estimated hours | - |
| Estimated cost | - |
| **Activity** | **Write Manuals** |
| Purpose | To write supporting documentation to explain the installation and the way the system works. |
| Participants | Duo Liang, Claudia, Kosta |
| Duration(Time period) | 14th November – 21st November |
| **Artifact** | **Installation Manual** |
| Description | A document explaining the way to install the system. |
| Estimated hours | 20 |
| Estimated cost | 400$ |
| **Artifact** | **User Manual** |
| Description | A document explaining the way to use the system. |
| Estimated hours | 20 |
| Estimated cost | 400$ |

## Activities

See the table in 7

## Artifacts

See the table in 7

## Project Estimates

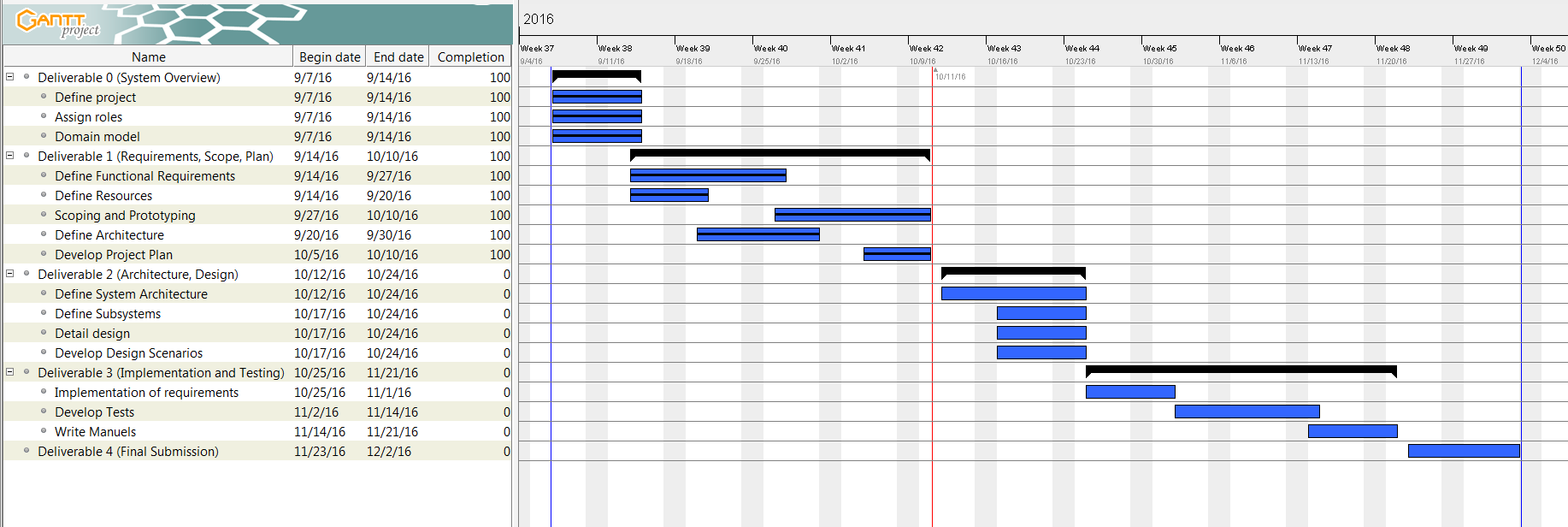
See the table in 7

## Activities Assignments

See the table in 7

## Schedule

See next page



## Risk

### **Time Constraint**

Team members may be unable to deliver tasks which they chose to undertake or were designated to do, due to limiting factors such as work, studying for other courses, lack of discipline, dedication or motivation. To help overcome this problem prioritization, team inspiration and mutual assistance is key to drive each other forward.

### **Team Cohesion and Communication**

Team members are not necessarily affiliated with each other and depending on their personalities may find it hard to work with one another. Team members that are international students may complicate communication due to language barriers. It may also be hard to arrange a good time for the members of the team to meet on a regular basis. To tackle these problems the team can meet on occasions that do not regard work, a team member that is proficient in both languages can help relay communication between members and the team can split into smaller groups which have similar schedules and overlapping work on the project.

### **Team Experience**

Team members may not be proficient or confident in using the resources, frameworks or languages that have been chosen to implement the project. This may lead to time being allocated in getting familiar with these technologies or to the uneven distribution of the workload. In order to avoid this the members that are unfamiliar should be assigned alternative tasks or should work alongside others in order to learn.

### **Uncooperative Members**

A team member may consciously and deliberately avoid working on the project, dragging back the rest of the team, causing tensions and impeding work. In this case the professor should be notified promptly in order to diffuse and rectify the situation.

### **Poor Implementation**

The team may not be adequately equipped and experienced ti handle a project of this level and therefore as a consequence the final product may be susceptible to undesired behavior due to bad implementation of certain modules. The design of the web application may also lack certain desired features that were scoped out or not taken into consideration due to lack of time or the complexity of their implementation. The number of unmet specifications due to bad source code can be minimized through adequate testing both during the implementation and acceptance phase.

### **Unforeseen Risks**

A team member that was highly valued may be unable to contribute to the project anymore due to dropping out or for other reasons. To stop this from impeding the project other members of the team should be able to comprise the knowledge necessary to complete the tasks the lost team member was implicated in. Data may be lost or tampered with. To avoid data loss files should be saved regularly and uploaded to a repository.

# Prototyping

Generally, we divided our coder team into two small sections, one is for a front-end interface, and another is for a back-end database.

All course information will be stored in the database, in addition to other data necessary for the application.

The focus of the back-end developers is to bridge the application and the information contained in the database. This is to say that during login the application must reference existing accounts in the database to validate information entered by the user and during sequence/schedule generation the database must reference the user’s information with the course information contained in the database. The database will also store preferences that the user has already entered.

*Figure 8.1: User Table Prototype*





*Figure 8.2: Course Table Prototype*

As for the front-end interface, we plan to have four web pages, including a home (log in) page, a registration page, a course sequence page, and a schedule page. Therefore, we have to code in HTML, PHP, CSS, and JavaScript.

Home (log in) page is for user to log in, as the *Figure 8.3* shows. It also contains a registration link directory on the right top.

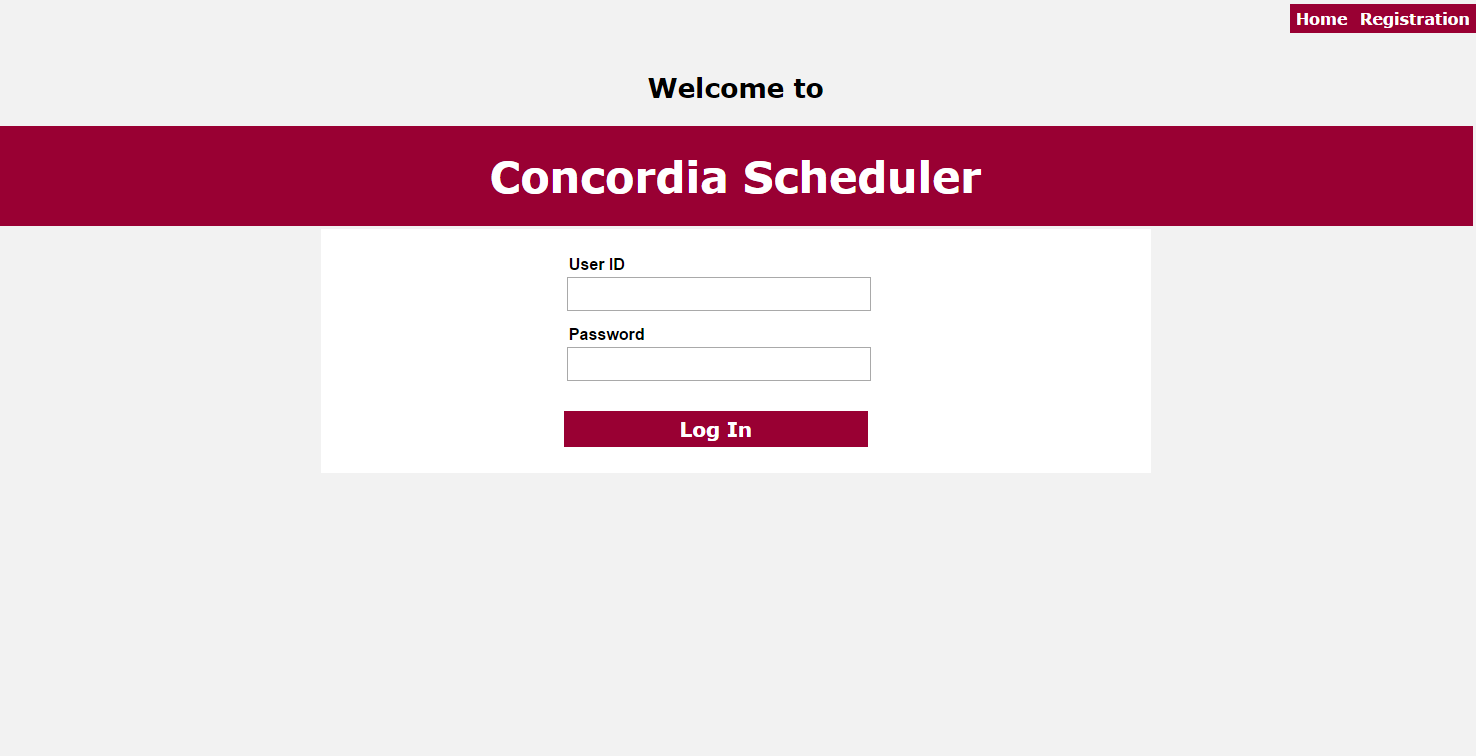


Figure 8.3 : home page

The registration page is used for registration, and requires users (students) to fill a form of personal information in order to register a new account, shown as *Figure 8.4*.

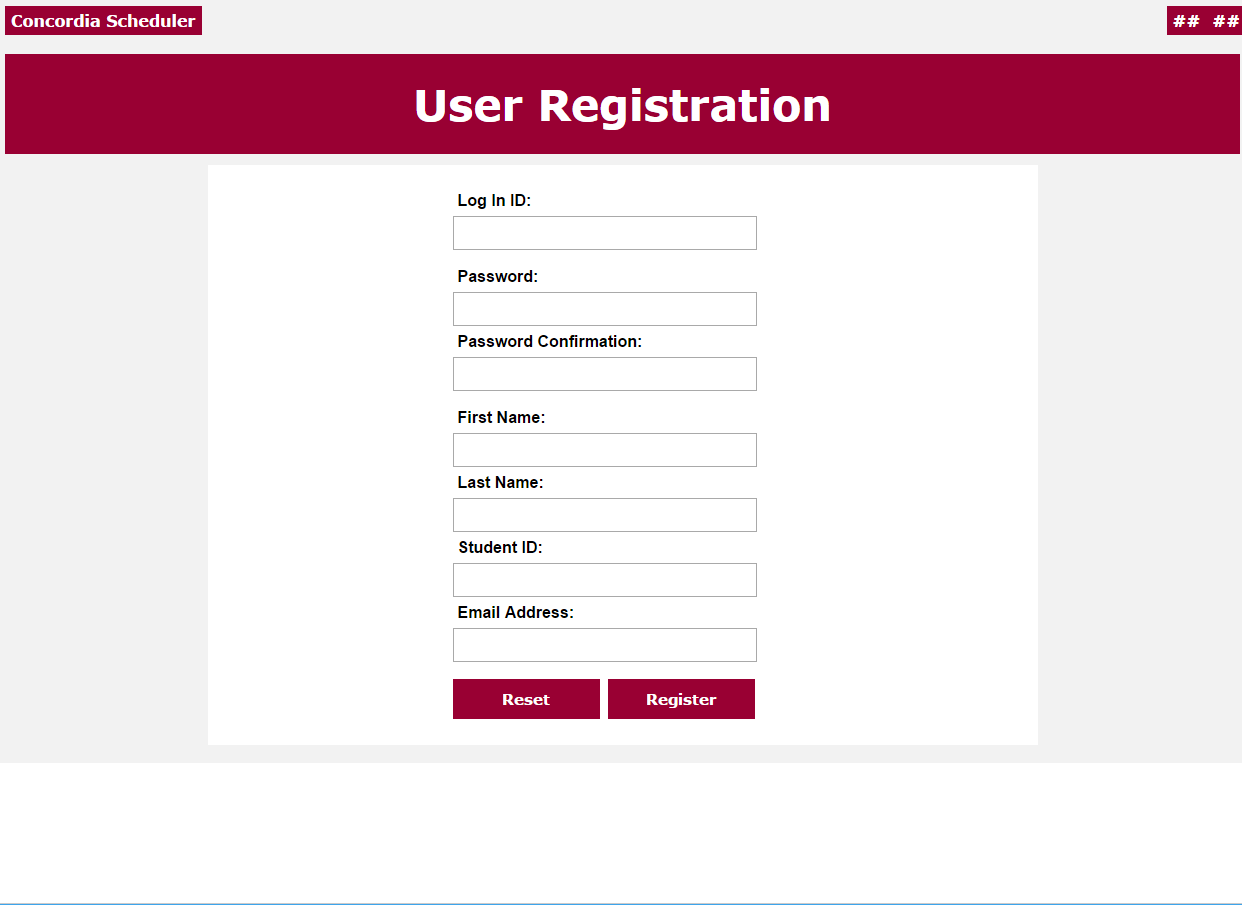


Figure 8.4 : Registration Page

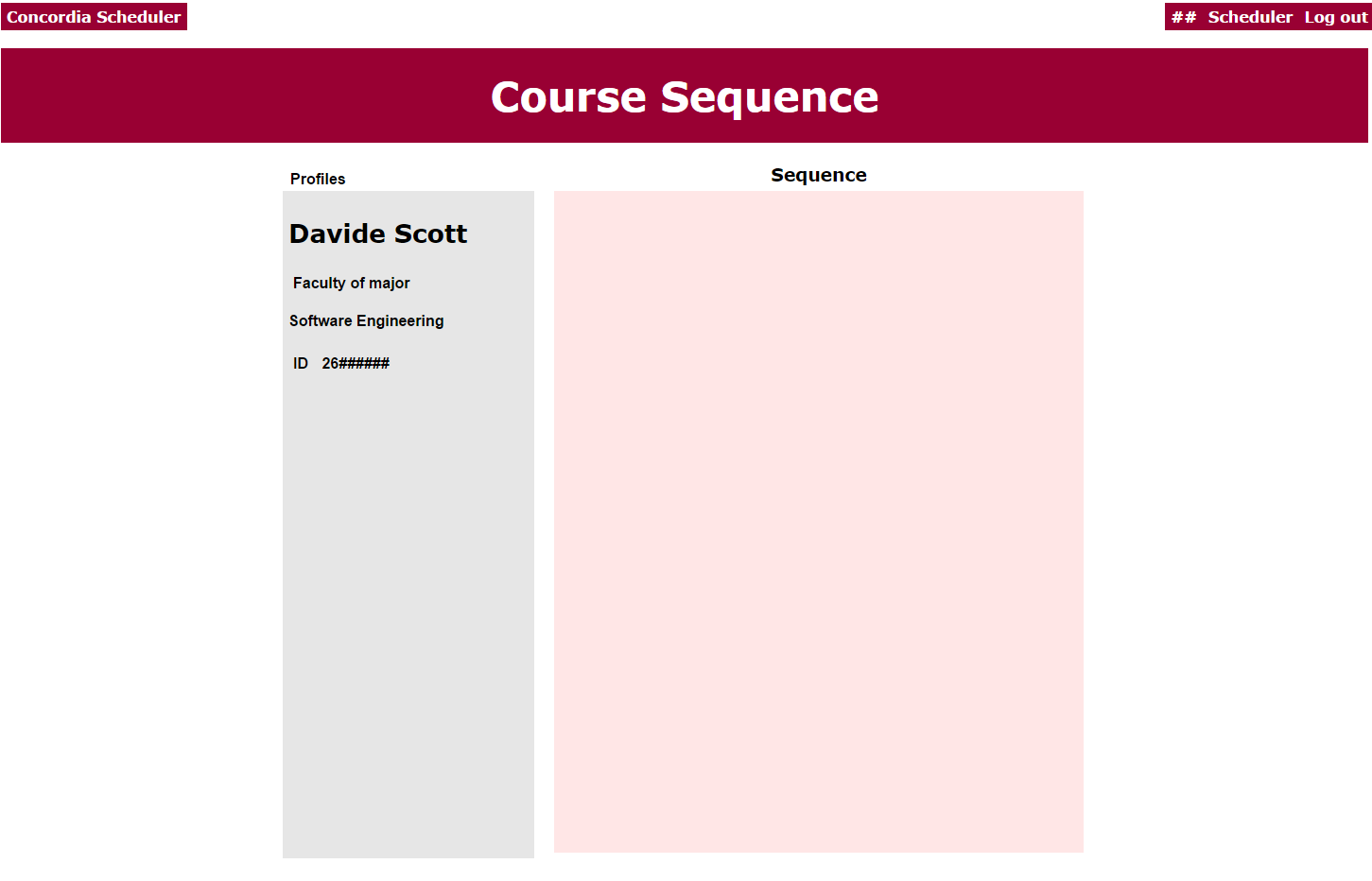


Figure 8.5 : Course Sequence

Course Sequence page (shown as *Figure 8.5*) would contain brief individual information on the left side and the course sequence on the right side. As for the appearance of the sequence, it would be list basically (shown as *Figure 8.6*); if we get enough time will might add a grid view of the sequence. Also, it contains a top-right navigation bar with a directory towards course schedule.

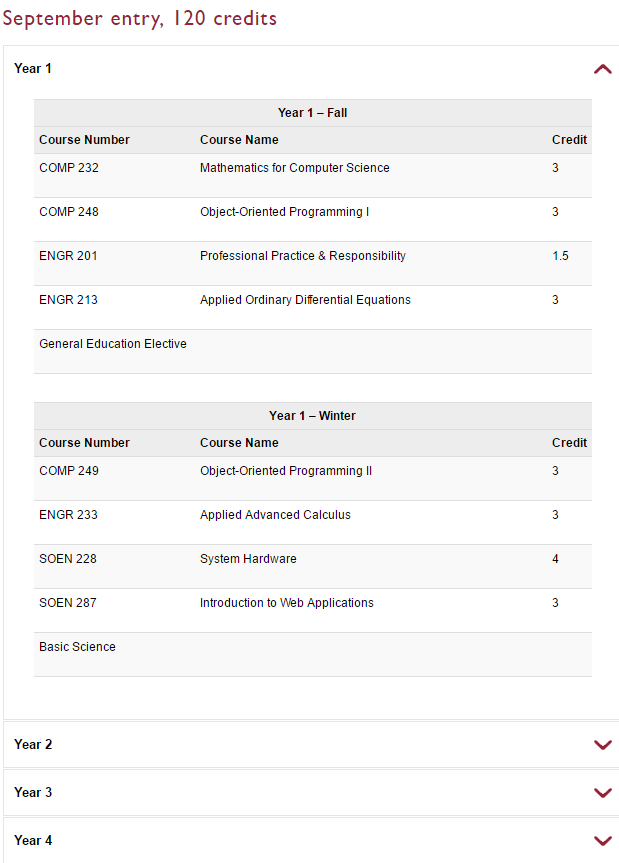


Figure 8.6 : Sequence List – Retrieved from https://www.concordia.ca/encs/computer-science-software-engineering/students/course-sequences/sept-soen-general.html

Course Schedule page would have similar layout with the sequence page (shown as *Figure 8.7*). Within the schedule box, it would be like *Figure 8.8*.

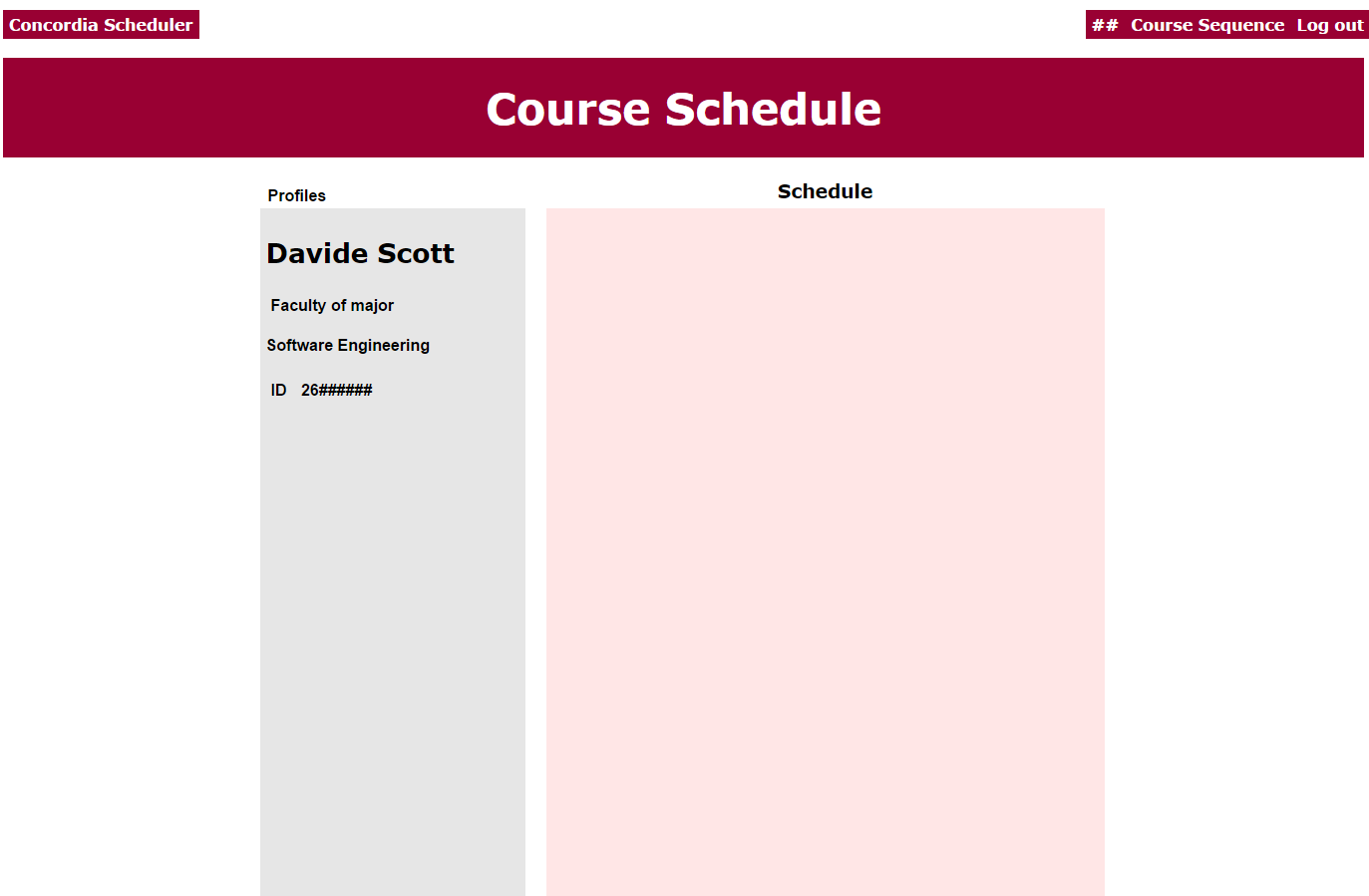


Figure 8.7 : Course Schedule

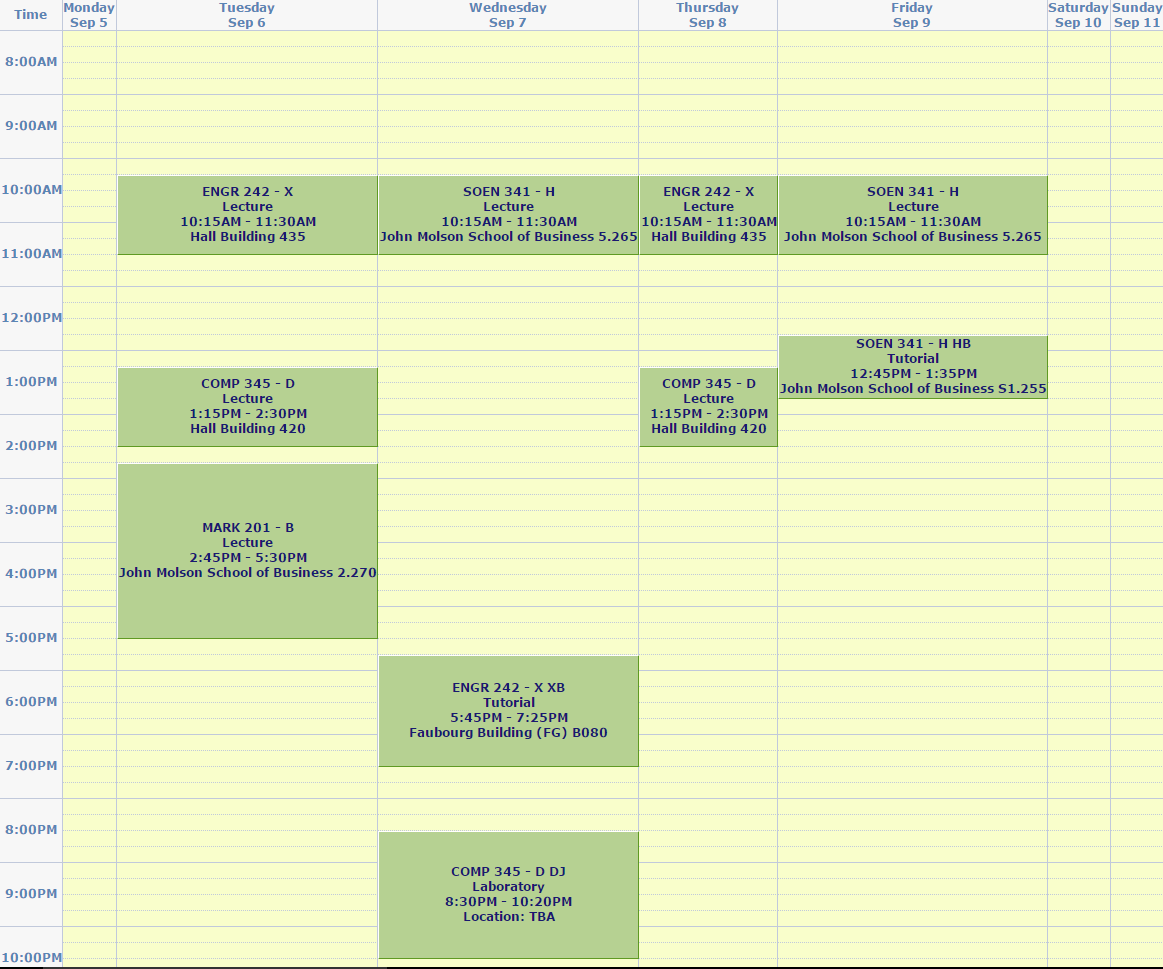


Figure 8.8 : Schedule Example – Retrieved from Concordia student weekly schedule

# Management

The management of the project is split into 2 Subteams, Documentation and Coding, as well as the Group lead to oversee the project in its entirety; Design is included in the Coding group. The Documentation and Coding groups have a group leader for that particular group.

## Code management

GiHub is used to maintain the code required for the scheduler application. There is a main branch, code from which is run on the server with the final product, a dev branch where untested code is stored and updated, and a feature branch which includes all the potential features that could be added, but are currently descoped. The master branch has a strict modification policy, where the commit to it has to be approved by the leading coder of the group. The other two branches are laxer in that regard allowing for experimentation.

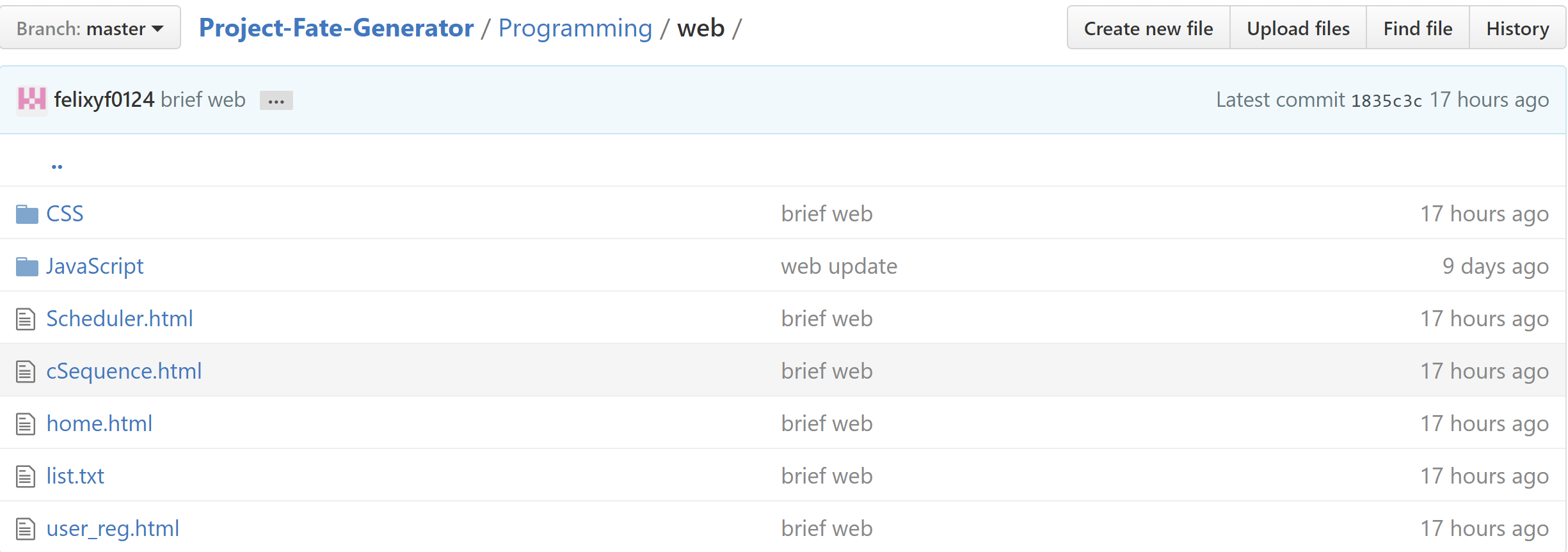


Figure 9.1: GitHub master branch for the web part of the project

## Team communication

Slack is the main communication tool for the team, used to set up meetings, discuss potential issues and the development for the documentation team. It is heavily used by the documentation team to share their drafts, receive and give out feedback and review meeting results. Our Slack channel consists of three different sub-channels: General, Coding and Documentation. Every member can post, access channels and upload or download files.

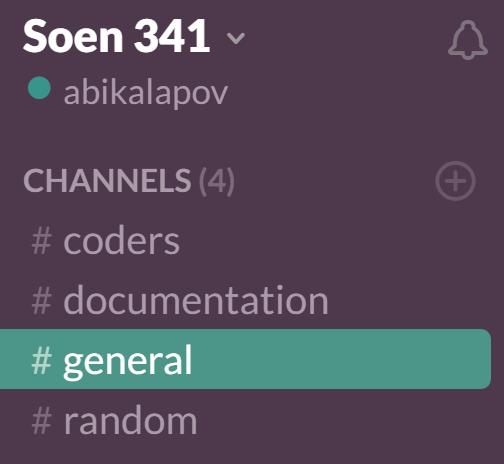


Figure 9.2: Slack channels used by the team

## Ticketing System

For issue tracking we have also resorted to GitHub, where the issues can be organised using labels and milestones. Coding, Documentation and the management teams have all used the issue tracker.

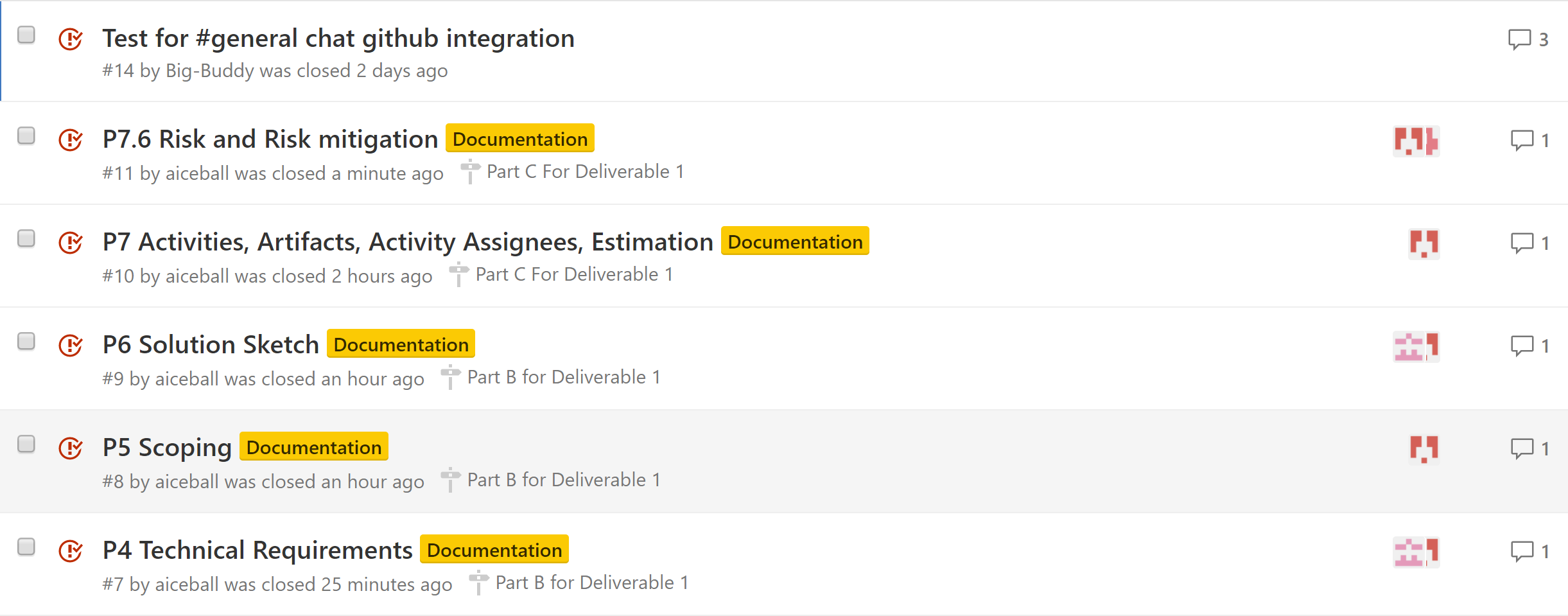
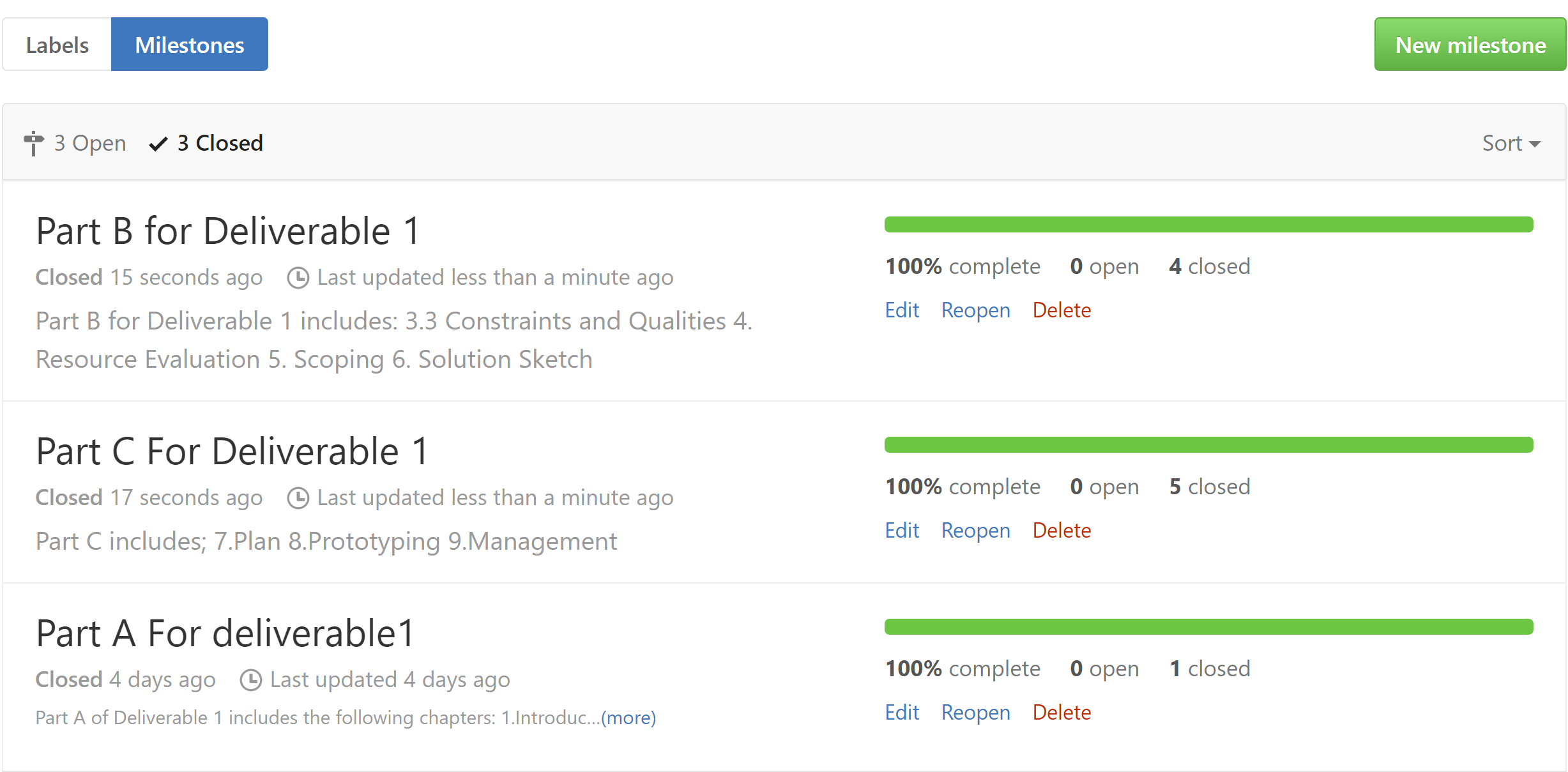


Figure 9.3: GitHub Issues page used as ticketing system

## Milestones

Milestones were designed by the management team to track down completion rate of the project and balanced task delegation. For example, Deliverable 1 was split into parts A,B and C and each part was assigned a milestone. This helped the project to stay on track and motivated the team to participate.



*Figure 9.4: Completed milestones for Deliverable 1*