Half Moon Studio

Study Scheduling App - PrioritU

Scrum Team:

Dominic Smart 816007605 (Scrum Master)

Shereece Victor 816001671 (Product Owner)

Jerrel Williams 816009134

Selah Lewis 816012230

COMP 3613

Software Engineering 2

Contents

[Introduction 2](#_Toc21943794)

[Purpose 2](#_Toc21943795)

[Scope 2](#_Toc21943796)

[Project Timeline 3](#_Toc21943797)

[Design Elements (Based on Requirements) 4](#_Toc21943798)

[Outline of Use Cases 5](#_Toc21943799)

[Extended Use Cases 6](#_Toc21943800)

[Use Case 3: Student Adds new course 6](#_Toc21943801)

[Use Case 4: Student Adds new class 8](#_Toc21943802)

[Use Case 6, 8: Student Adds new exam/assignment 10](#_Toc21943803)

[Use Case 5, 7, 9: Student modifies course/class/assignment/exam 12](#_Toc21943804)

[Use Case 11: Student marks assignment as complete 14](#_Toc21943805)

[Use Case 12: Student changes sorting order for tasks due list 16](#_Toc21943806)

[Class Diagram 18](#_Toc21943807)

# Introduction

The main goal of this project is to develop a working prototype of the PrioritU Scheduling app outlined in the requirements document. Initially the project was intended to be an improved version of the My study Life App available on Android devices and through a web application. However, further refinement of the group’s ideas created a simpler web interface aimed at making it easier for students to manage their time.

The purpose of this design document is to illustrate and provide details about the architectural design, interface design and core functionalities via sequence diagrams. This document will also define the projected timeline for implementation of the various design elements using a scrum approach.

## Purpose

This purpose of this document is to outline the approach decided upon by Team Half Moon Studio to implement the PrioritU application according to the requirements specified. To date, no particular architecture pattern has been chosen for this project.

## Scope

This document provides design specifications of the PrioritU application, including high level explanations of how certain use cases will function and how it will be implemented. This will be done using use cases of unique functions and a class diagram and sequence diagrams.

## Project Timeline

Development period

Week 1 - Oct 13th - 19th

Set up database, website, establish link

Week 2 - Oct 20th - 26th

Add APIs and Middleman classes

Week 3 Oct 27th- Nov 2nd

Test Implemented functions, refine code

Week 4 Nov 3rd -9th **MILESTONE 4 DUE**

Test and refine GUI

Week 5 Nov 10th - 16th

Add and Test secondary functionality, prepare presentation

Week 6 Nov 17th - 23rd

Finalise Documentation, present project

# Design Elements (Based on Requirements)

* Database / Server for storing user information, settings and configuration
  + Includes manager for information requests, additions, updates, deletions changes.
  + User information
    - Name
    - Email
    - Number of Courses
    - Courses information
  + Courses (Course code, name, duration, colour?)
    - Classes (times, days, type, teacher?, room)
    - Exams (time, date, duration, type - final/coursework, weight)
    - Assignments (time, date, name, complete/ incomplete, weight)
* Front end Dashboard and user interface
  + Settings?
  + Calendar (Google api)
  + Recommendations
  + Lists of upcoming tasks and exams
  + Sign out button
  + FAQ tab or menu option (I don’t think we should do a menu, that implies several pages. Would a dashboard be easier to make than several pages?)
* Middleman/ Processing server
  + Sorting Classes for displaying the tasks and assignments due in order.
  + Display manager for applying colours and themes to displays
  + Reminder plugin
  + Calendar plugin
  + Admin access or control portal? (ignore this)
  + Login plugin and authentication
  + Add or modify exams
  + Add or modify assignments

## Outline of Use Cases

1. Student Logs in
2. Student signs out
3. Student adds New Course
4. Student adds new Class (extending: add new course)
5. Student modifies class
6. Student modifies course
7. Student adds new exam
8. Student modifies exam
9. Student adds new Assignment
10. Student modifies assignment
11. Student marks assignment as complete
12. Student changes sorting order for tasks due list
13. Student chooses colour scheme

# Extended Use Cases

### Use Case 3: Student Adds new course

|  |  |
| --- | --- |
| Description | A student adds one of their courses to the app |
| **Actors** | Student |
| **Type** | Primary, Essential |
| **Pre-Conditions** | - The Student has logged into the application |
| **Flow of Events** | |
| Basic Path | |
| Actor Action | System Response |
| 1. The use case begins when the student selects the option to add a new course | 1. The system displays the add course form |
| 1. The student enters the course code. | 1. The system checks if the course code already exists. |
| 1. The student enters the course name |  |
| 1. The student selects a colour for the course |  |
| 1. The student selects the duration during which the course runs. |  |
| 1. The student may enter the lecturer's name. |  |
| 1. The student submits the form. | 1. The system acknowledges the completion of the form and updates the database and dashboard. |
| Alternative Paths | |
| Line 2: The course already exists. (see Student modifies course) | |
| Post-Conditions | A new course has been added to the app |
| **Related Use Cases** | Student modifies course |
| Used Use Case | None |
| Extending Use Cases | Student adds new class |

### Use Case 4: Student Adds new class

|  |  |
| --- | --- |
| Description | A student adds one of their classes to the app |
| **Actors** | Student |
| **Type** | Primary, Essential |
| **Pre-Conditions** | - The Student has logged into the application  - The Student has created a course |
| **Flow of Events** | |
| Basic Path | |
| Actor Action | System Response |
| 1. The use case begins when the student selects the option to add a new course | 1. The system displays the add class form |
| 1. The student selects whether the class is a lecture, tutorial or lab. |  |
| 1. The student enters the class information. | 1. The system checks if the class already exists |
| 1. The student submits the form. | 1. The system acknowledges the completion of the form and updates the database and dashboard. |
| Alternative Paths | |
| Line 2: The course already exists. (see Student modifies class) | |
| Post-Conditions | A new class has been added to the app |
| **Related Use Cases** | Student modifies class |
| Used Use Case | None |
| Extending Use Cases | None |

### Use Case 6, 8: Student Adds new exam/assignment

|  |  |
| --- | --- |
| Description | A student adds one of their exams or assignments to the app |
| **Actors** | Student |
| **Type** | Primary, Essential |
| **Pre-Conditions** | - The Student has logged into the application  - The Student has created a course |
| **Flow of Events** | |
| Basic Path | |
| Actor Action | System Response |
| 1. The use case begins when the student selects the option to add a new exam or assignment | 1. The system displays the add exam form or add assignment form |
| 1. The student enters the relevant information into the form. | 1. The system checks if the exam or assignment already exists. |
| 1. The student submits the form. | 1. The system acknowledges the completion of the form and updates the database and dashboard. |
| Alternative Paths | |
| Line 2: The exam or assignment already exists. (see Student modifies course/assignment/exam) | |
| Post-Conditions | A new exam or assignment has been added to the app |
| **Related Use Cases** | Student modifies exam/assignment |
| Used Use Case | None |
| Extending Use Cases | None |

### Use Case 5, 7, 9: Student modifies course/class/assignment/exam

|  |  |
| --- | --- |
| Description | A student modifies the entered information for a course, assignment or exam, from now referred to as an entry |
| **Actors** | Student |
| **Type** | Primary, Essential |
| **Pre-Conditions** | - The Student has logged into the application  - A course assignment or exam has been entered by the Student |
| **Flow of Events** | |
| Basic Path | |
| Actor Action | System Response |
| 1. The use case begins when the student selects the option to modify an existing entry | 1. The system displays the modify entry form |
| 1. The student updates the displayed form with the new values specific to the entry type. |  |
| 1. The student submits the form. | 1. The system acknowledges the completion of the form and updates the database and dashboard. |
| Alternative Paths | |
| Line 1: The Use case begins when the user enters an entry that already exists. (See Student adds new course/exam/assignment) | |
| Post-Conditions | The entry has been modified to match the values entered by the user. |
| **Related Use Cases** | None |
| Used Use Case | None |
| Extending Use Cases | None |

### Use Case 11: Student marks assignment as complete

|  |  |
| --- | --- |
| Description | Student marks an assignment as complete |
| **Actors** | Student |
| **Type** | Primary, Essential |
| **Pre-Conditions** | - The Student is already logged into the application  - The Student has already added an assignment task to the app |
| **Flow of Events** | |
| Basic Path | |
| Actor Action | System Response |
| 1. The use case begins when the student selects the option to view assignments | 1. The system displays all assignments the student has entered and not completed |
| 1. The student selects an assignment or multiple assignments and clicks the ‘Mark as complete’ option | 1. The system removes the assignment(s) from the incomplete assignment page and enters it into the completed assignments page |
|  | 1. The system removes the assignment(s) from the student’s calendar |
|  | 1. The system notice is sent to the student acknowledging the assignments as completed |
| Alternative Paths | |
| None | |
| Post-Conditions | - The student calendar is clear of the assignment  - The assignment is now listed under ‘completed assignments’ |
| **Related Use Cases** | Student adds an assignment  Student modifies an assignment |
| Used Use Case | None |
| Extending Use Cases | None |

### Use Case 12: Student changes sorting order for tasks due list

|  |  |
| --- | --- |
| Description | Student changes sorting order and criteria for tasks due list |
| **Actors** | Student |
| **Type** | Primary, Essential |
| **Pre-Conditions** | - The Student is already logged into the application  - The Student has already added tasks to the app |
| **Flow of Events** | |
| Basic Path | |
| Actor Action | System Response |
| 1. The use case begins when the student selects the option to view tasks | 1. The system displays all tasks the student has entered and not completed, by default in order by due date |
| 1. The student selects the option ‘Sort by’ | 1. The system displays the options to view tasks by weighting or due date |
| 1. The student selects the option to sort by weighting instead of due date | 1. The system displays the tasks due in order of weighting instead of due date. |
| Alternative Paths | |
| None | |
| Post-Conditions | - The list of tasks is now ordered by weighting |
| **Related Use Cases** | None |
| Used Use Case | None |
| Extending Use Cases | None |

# Class Diagram

