|  |
| --- |
|  |
|  |
| November 30 |

A close up of a logo

Description automatically generated

816012230 816007605 816001671 816009134

Acknowledgements

Half Moon Studio would like to express gratitude to our project supervisor, Inzamam Rahaman, for his guidance and support during the development of this application. We would also like to acknowledge those who kept us sane during the long hours of coding and bickering. No height is ever achieved without some sacrifices made, and it is here where we owe a special debt to our loved ones for showing their generous affection and care throughout the entire development period. Finally, we would like to acknowledge us, the creators of this app. We worked hard and put our best foot forward. It was worth it.

# Abstract

University students are known to have hectic schedules, and often the workload can become overbearing. Time management is a skill that is key to managing this workload. PriorityU is an application developed using Python, Flask, and SQLAlchemy, designed to aid university students in keeping track of courses, assignments, and exams. The easy to use interface and clean design make the application usable for all levels of computer literacy. The goal of this report is to detail the development and production of PriorityU.

**Table of Contents**

[Abstract 2](#_Toc25969061)

[Introduction 4](#_Toc25969062)

[Purpose of Project 4](#_Toc25969063)

[Project Description 4](#_Toc25969064)

[Tools and Technology 5](#_Toc25969065)

[Requirements 6](#_Toc25969066)

[Functional Requirements: 6](#_Toc25969067)

[Non-functional Requirements: 7](#_Toc25969068)

[User Stories 8](#_Toc25969069)

[Outline of Use Cases 10](#_Toc25969070)

[Extended Use Cases 11](#_Toc25969071)

[Use Case 3: Student adds new course 11](#_Toc25969072)

[Use Case 5, 7: Student Adds new exam/assignment 13](#_Toc25969073)

[Use Case 4, 6, 8: Student modifies course/assignment/exam 15](#_Toc25969074)

[Use Case 11: Student marks assignment as complete 17](#_Toc25969075)

[Use Case 10: Student deletes an assignment 19](#_Toc25969076)

[Design 21](#_Toc25969077)

[Classes and Methods 22](#_Toc25969078)

[Forms.py 22](#_Toc25969079)

[Models.py 23](#_Toc25969080)

[Routes.py 24](#_Toc25969081)

[Class Diagram 26](#_Toc25969082)

[Sequence Diagrams 28](#_Toc25969083)

[Future Works 31](#_Toc25969084)

# Introduction

## **Purpose of Project**

The purpose of this project work is to provide students with a proper method to organize and visualize assignment deadlines and exam dates. When work starts to pile up, students can become stressed and forget about other assignments or exams, which causes further stress. There exists a need for a simple, usable scheduling application. Thus, we developed PriorityU which will prove to be more user friendly and effective than other methods of recording deadlines.

## **Project Description**

PriorityU is intended for university students, more specifically the University of The West Indies’ students. In detail, the main function of the system is to record and remind students of assignments and exams. Once an account is made, users can enter course details into a form which will be saved and can be edited and/or deleted. Users are also able to add assignment and examination details, including course codes and weightings. Assignments can be marked as complete or deleted once finished. If marked complete, it will be visible in a separate tab. The application sends an alert to the user five days before an assignment is due if it has not been marked as complete. Each instance of the application is unique to the user.

# Tools and Technology

The following technologies were used:

* HTML5/Bootstrap framework This was used for the client side of the application
* Python3.6/Flask This is used as the backend of the application functionality
* SQLAlchemy Database This is where all data will be stored

# Requirements

## **Functional Requirements:**

|  |  |
| --- | --- |
| **Requirement ID** | **Requirement Description** |
| FR001 | The system allows users to enter course details, such as names, date/time, room, lecturer. |
| FR002 | The system allows users to enter assignment details, such as course, names, due dates, content. |
| FR003 | The system prioritises assignments based on due dates |
| FR004 | The system periodically reminds user of assignment deadlines |
| FR005 | The system allows users to enter examination details, such as course code/name, date/time |

## **Non-functional Requirements:**

|  |  |
| --- | --- |
| **Requirement ID** | **Requirement Description** |
| NF001 | The system should be able to utilise the Google calendar API |
| NF002 | The system interface should be simplistic yet aesthetically pleasing |
| NF003 | The system should provide a smooth and easy to use experience to first-time users |

## **User Stories**

As a student I want to be able to access this application from a browser so that it doesn't take up valuable space on my devices

As a student I want to be able to login before viewing my information so that it is only accessible by me

As a student I want to be able to add courses and their details so that I can be more organised visually

As a student I want to be able to add assignments and their details so that I can be more organised visually

As a student I want to be able to add exams and their details so that I can be more organised visually

As a student I want to be able to previously added courses, assignments and exams so that when I login in another time so that I don't have repeatedly add them

As a student I want to be able to delete previously added courses, assignment, exams so that they don't remain on the website indefinitely

As a student I want to be able to easily visualize my exams and assignment deadlines so that I don't miss any of them

As a student I want to be able to modify details on courses, assignments and exams added so that they can reflect real life changes, and I can correct errors.

As a student I want to be able to mark my assignments as completed so that my to do list reduces when I have completed a task

As a student I want to be able to be reminded of upcoming deadlines so that I don't forget them

## Outline of Use Cases

1. Student logs in
2. Student signs out
3. Student adds new course
4. Student modifies course
5. Student adds new exam
6. Student modifies exam
7. Student adds new assignment
8. Student modifies assignment
9. Student marks assignment as complete
10. Student deletes assignment

## **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 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 | None |

### Use Case 5, 7: 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 4, 6, 8: Student modifies course/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 9: 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 notice is sent to the student acknowledging the assignments as completed |
| Alternative Paths | |
| None | |
| Post-Conditions | - 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 10: Student deletes an assignment

|  |  |
| --- | --- |
| Description | Student deletes an assignment |
| 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  - The Student has marked an assignment as complete |
| Flow of Events | |
| Basic Path | |
| Actor Action | System Response |
| 1. The use case begins when the student selects the option to view completed assignments | 1. The system displays all assignments the student has entered and completed |
| 1. The student selects an assignment and clicks the ‘Delete’ option | 1. The system removes the assignment from the database |
|  |  |
| Alternative Paths | |
| None |  |
| Post-Conditions | - The assignment is no longer listed under “Completed assignments” and is removed from the database |
| Related Use Cases | Student adds an assignment  Student modifies an assignment |
| Used Use Case | None |

# Design

**Design Elements (Based on Requirements)**

- Database / Server for storing user information, settings and configuration

- User information

- Name

- Email

- Number of Courses

- Courses information

- Courses (Course code, name, duration)

- Exams (time, date, duration, type - final/coursework, weight)

- Assignments (time, date, name, complete/ incomplete, weight)

- Front end dashboard and user interface

- Lists of upcoming tasks and exams

- Sign out button

- Middleman/ Processing server

- Sorting classes for displaying the tasks and assignments due in order.

- Reminders

- Login and authentication

- Add or modify exams

- Add or modify assignments

# Classes and Methods

## Forms.py

class LoginForm(FlaskForm) – form used to accept user login information

class RegisterForm(FlaskForm) – form used to accept user signup information

def validate\_email(self, email) – ensures that input, email, is a valid email address for RegisterForm

def validate\_username(self, username) – ensures that input, username, is a valid username and not taken address for RegisterForm

class NewCourseForm(FlaskForm)

class NewExamForm(FlaskForm) – form used to accept new exam information

def validate\_course\_code(self, course\_code) – ensures that input, course\_code, is a valid course code in the system for NewExamForm

class NewAssignmentForm(FlaskForm) – form used to accept new assignment information

def validate\_course\_code(self,course\_code) – ensures that input, course\_code, is a valid course code in the system for NewAssignmentForm

## Models.py

class User(UserMixin, db.Model) – class that defines User structure and behaviour

class Courses(db.Model) – class that defines User structure and behaviour

class Assignment(db.Model) – class that defines Assignment structure and behaviour

class Exam(db.Model) – class that defines Exam structure and behaviour

## Routes.py

def load\_user(user\_id) – loads user data if a returning user

def index() – loads index page

def login() – validates user data for logging in

def signup() – creates a new user if given valid data

@login\_required : these functions require users to be logged in so they can be accessed

def dashboard() – loads user dashboard data from database

def courses() – loads course data from database

def addCourse() – adds a course to database

def deleteCourse(c\_id) – deletes a course from database

def exams() – loads exam data from database

def updateExam(e\_id) – edits exam data

def deleteExam(e\_id) – deletes an exam from database

def addExam() – adds an exam to database

def assignments() – loads assignment data from database

def addAssignments() – adds an assignment to database

def updateAssignment(a\_id) – edits assignment data

def markCompleteAssignment(a\_id) – sets complete to true, moves data

def deleteAssignment(a\_id) – deletes an assignment from database

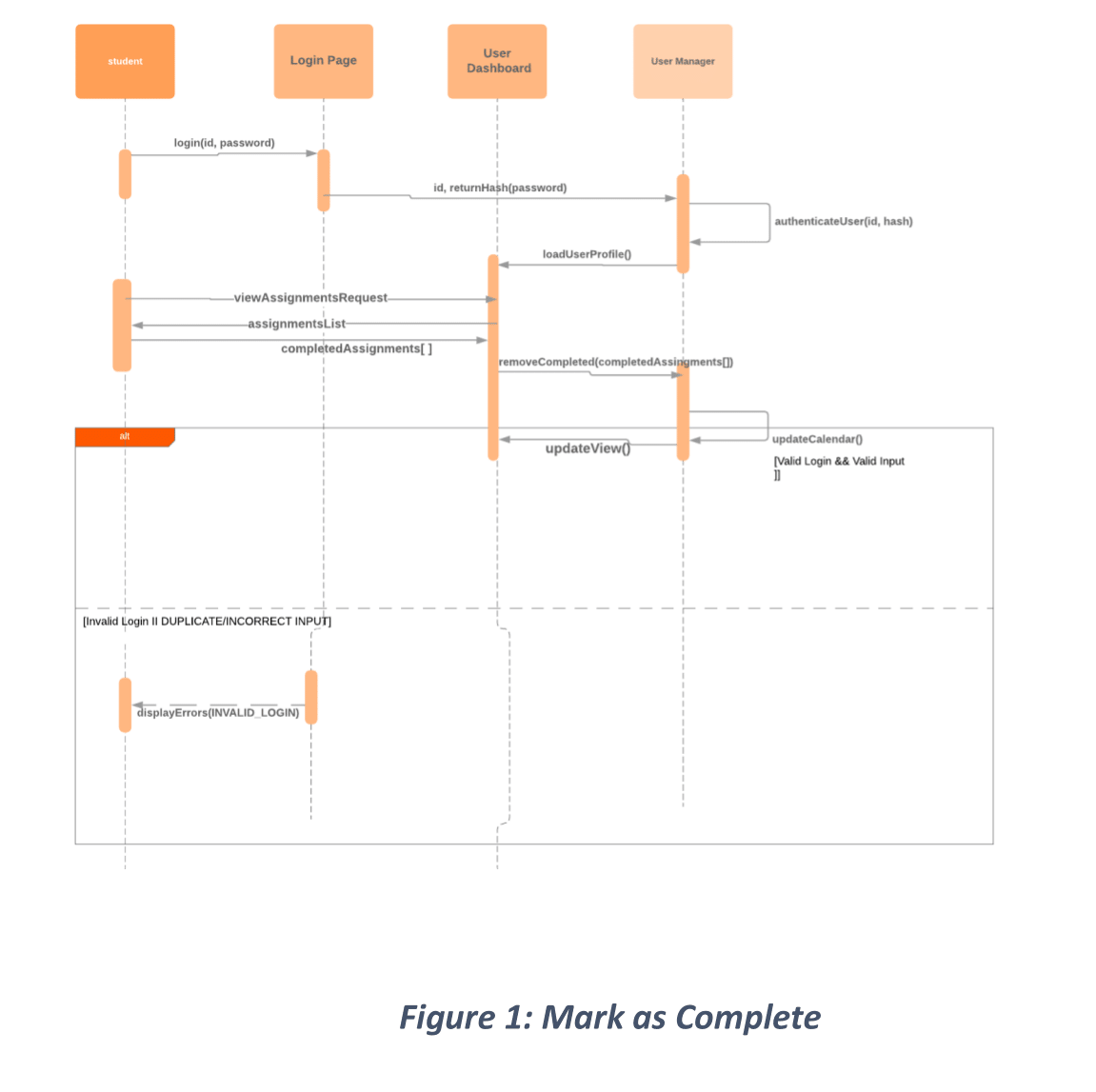
def completed() – loads assignments marked as read from database

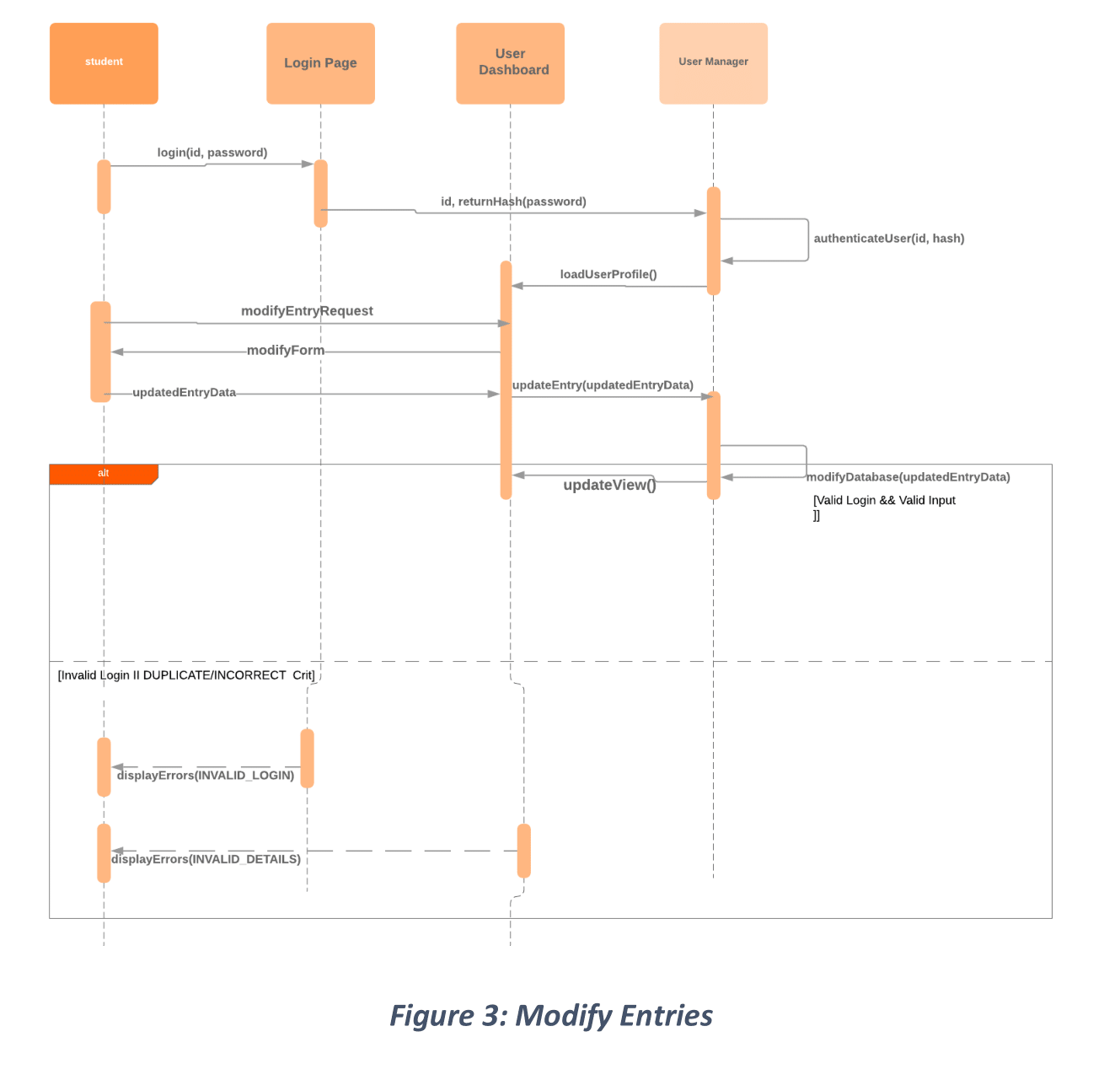
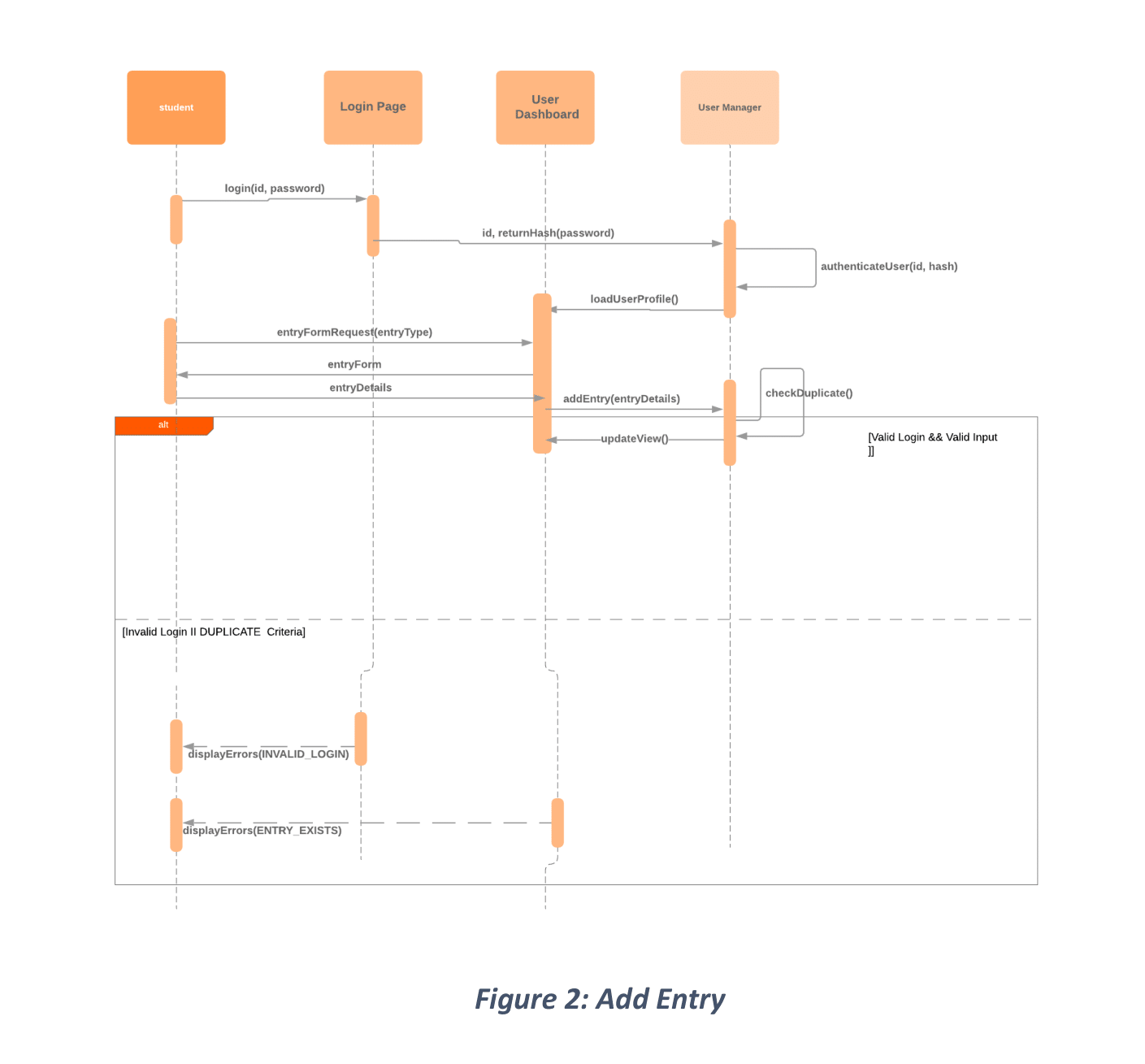
def deleteCompleted(a\_id) – deletes an assignment marked as complete

def logout() – logs the user out from the app

## **Class Diagram**

## **Sequence Diagrams**





Future Works

Due to time restraints, the team had to limit the application to a subset of functions we deemed most necessary. Reflecting on our backlog, there is room for refinements. Some of the features to come include:

1. More granular control over the alert system. User should be able to decide when they would like to be updated about their tasks.
2. Colour Coding of Tasks. This design choice will play a vital role in improving the user interface/experience by granting the user the ability to add familiar visual cues to their tasks.
3. Single Sign On. Users will be given the ability to log in via their Google accounts.
4. Google API. We will continue with our plan to implement Google’s calendar so that our app provides a seamless experience for those that have already found a home in Google’s software environments.
5. Sharing. A future sprint will also see the implementation of a sharing feature that allows users to share their task management activities with friends/colleagues.

## **Burn Down Chart**

## **Testing Report**

During the development of the requirements and design for the proposed PriorityU app the strategy to be used for testing was decided upon based on knowledge of python and the projected timeline.

The proposed plan for testing involved ensuring that each new feature implemented is continuously tested and revised. This will be done on a weekly basis using a mixture of functional, black box, and white box testing to ensure that at the result of every individual and group coding activity is a working program.

The features tested would be selected based on the contents of the product backlog and the progress in terms of its implementation. Testing is to occur continuously through the project duration.

The following is an overview of the testing done throughout the project. It was mostly high level and done with the aim of continuously error checking and refactoring code to ensure that the functionality was correct and the app was useable.

The **Sign-In**, **Sign-Up** **pages** were tested along with the **Log-out** **feature**.

1. The sign-up page was used to sign up using simple values for username, email, university, and password. They were only accepted when they were of the appropriate length and format. This was deemed satisfactory.
2. The sign-up button was selected without any values being entered. An error message was displayed stating that values had to be entered before submission. This was deemed satisfactory.
3. After signing up the login in page denied entry to the website without the previously entered values for username and password. This is displays correct functionality.
4. The sign in page allows access to he website using the previously entered values for password and username.
5. The Sign-out button exited the website. And pressing the browsers back button reloaded the sign in page instead of the website. This is satisfactory functionality.

The contents of the **database** were checked after logging in to see if the credentials were being stored, I they were saved until the database was wiped clean and reset.

The **Add Courses feature** was tested in various stages after every step in its implementation.

1. The add courses form was checked to see if empty fields could be successfully submitted. They could not.
2. The form was tested with values of various lengths, and formats, it was found that the course code field only accepted the UWI format for course codes despite the app being designed for any university student. That was changed.
3. The course time and day fields were deemed unnecessary for this project and were eliminated. However, a location field was added.

The **Exam** and **Assignment features** were tested similarly since they contain almost identical implementations and contents.

1. The date and time fields were tested with random values. An error was displayed stating that the format was incorrect. It was decided that placeholders were needed to ensure that the correct format is known to users.
2. It was found that courses which were not previously entered could be added to the assignment and exam forms. This is in violation of our functional requirements since the databases course table should have a comprehensive record of courses added to the user’s page. This was later changed by adding a validator which ensures that exams and assignments cant be added with new courses.

The **exams and assignment pages** were tested to ensure that they contain all the assignment and exam information entered during the tests and stored on the database.

1. There were a few instances when the information did not appear on their respective pages. These errors were fixed. Thus, the functionality was deemed satisfactory.
2. Each page was tested to see if the courses, exams, and assignments were removed after the delete function was added. The database was also checked to ensure that the information matched.
3. The pages were again verified after the modify feature was added.

The **Mark Complete Feature** was added to the assignment pages.

1. The function was tested to see if marking ‘complete’ removed the listings from the assignment page. This feature failed at first but was later changed.
2. The database was checked to see if the ‘complete’ value for each marked assignment was changed when marked on the webpage. This feature was satisfied.
3. The completed assignment page was checked to see if it listed the completed assignments after they were marked. This feature failed several times but was later correctly implemented.

The **Dashboard** was tested to check if it was being populated correctly

1. Several courses, exams and assignments were added then assignments were marked completed. The database was verified to see if it contained the entered values and the dashboard page was then visited to check if it contained the entered information. Several inconsistencies were discovered especially with the completed assignments section. They were later resolved and the requirement deemed fulfilled.

The **Alerts feature** was the last to be tested.

1. It was checked to see if it contained only events occurring within the next five days. That was deemed the case.
2. Then it was checked to ensure that the information displayed belonged to the correct classes. This requirement was not satisfied and was later fixed.