**Software Requirements and Design Document**

**For**

**Group <3>**

Version 1.0

**Authors**:

Luke B

# Overview (5 points)

The goal of the system is to be a hosted website acting as a student centric dual calendar and planner. It will be hosted on a website featuring several pages including a sign in page and the main page featuring the calendar/planner. The website’s will be made using HTML and CSS and will be connected to the backend using the Flask API, which will link to the backend database. The website is intended to be student centric displaying times for classes, upcoming homework assignments, study time for users, and more features. Along with this it will be connected to Canvas via the Canvas API to provide up to date information and get live updates.

# Functional Requirements (10 points)

1. Host a webpage for users to view a calendar/ planner. High.
2. Have a sign in page for users. High
3. Connect to the Canvas API to get users information for classes and upcoming assignments. High
4. Have a page for user settings. High.
5. Have a page for groups/messaging. High.
6. Have a database for storing user information. High.
7. Host webpage. High.
8. Host database. High.
9. Support scheduling/ adding events to calendar. High.
10. Change view level of calendar, ie Day, Week, and Month level. Medium.
11. Have a button to automatically find and schedule study time. Medium.
12. Display important upcoming assignments. Medium.
13. Display recent messages on main page. Low.
14. Implement a texting notification service. Low.
15. Implement a notepad/journaling page. Low.
16. Import schedule from myFSU. Low.

# Non-functional Requirements (10 points)

1. Securely store user information on backend database.
2. Ensure webpage is fast and reliable.
3. Try to ensure all backend features such as saving user settings and importing data to and from database is quick and consistent.

# Use Case Diagram (10 points)

For this project currently the only type of users are generic “Student Users”.

A current first iteration use case diagram is given below:

Diagram

Description automatically generated

# Class Diagram and/or Sequence Diagrams (15 points)

A sequence diagram works better for this project as we are not using a class structure, but what is more important is the order of events for each use case. Currently still in early stages so have not fully developed a sequence diagram. Most use cases would likely consist of a process like pushing a button on a webpage, the webpage interacts with the Flask API, which then either sends or requests data from the database.

# Operating Environment (5 points)

For increment one the webpage is just using HTML and CSS which should be supported by every web browser and each of these are supported by the common operating systems. While the webpage is currently designed for viewing on a laptop/ desktop computer, it would be able to be viewed on a phone, though it would not look nice.

# Assumptions and Dependencies (5 points)

As mentioned in part 6 for increment one the webpage is currently designed for being viewed on a laptop/desktop and would not look nice when viewed on a phone.