WASHINGTON STATE UNIVERSITY
VANCOUVER

Software Requirements Specification

for

<Team Pi >

Version <1.0>

Prepared by

Group Name: *<Team Pi >*

<Caylin Seitz> **<caylin.seitz@wsu.edu >**
<Ramsay Fencl> **<ramsay.fencl@wsu.edu>**

Date: **<12/16/2020 >**

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Caylin Seitz Ramsay Fencl	First draft, includes the fundamental design details for the project	11/06/2020
1.1	Ramsay Fencl, Logan Tan	Final submission edits to the Project SRS, to update out of date information.	12/16/2020

1 Introduction

Introduction:

This is an application that is designed to help students stay organized with their coursework throughout college. The goal being two main tools

1. The application helps students manage their time better.
2. By organizing which assignments will take longer or shorter amounts of time. The time taken can be used as a measure of where a student needs to study more.

The application is solely meant to help indicate where a student might need to apply more time in a subject than another. It is also meant to help a student stay organized around other social events and work.

1.1 Document Purpose

The purpose of this document is to layout the features of the program as well as how to operate the application. This also contains what the functions do in the program/global variables for later troubleshoots. This also provides information on minimum hardware requirements.
The follow sections will contain everything known about the application. Later versions will be appended to this document and added features as well.

1.2 Product Scope

Students who struggle with time management would greatly benefit from this application. This product is meant to help students ~~organize their time into blocks of 15-minute~~ day long increments. The student can put in their work, school, and other obligations for the week to have a simplified timetable to follow. In the application they can add their homework/test/quizzes ~~due dates/times~~ as well as notes about the assignment.

~~When a student creates a new event, they can add start/end time and date. They can also add where the event is at if they wish. In the documentation for schoolwork the student can record how long it took them to complete the assignment. The program can add those average times for the week and can give feedback on where to focus studying. If there is improvement in the amount of time taken on an assignment it can give positive feedback.~~

1.3 Intended Audience and Document Overview

The intended audience is mainly students in school. However, this could be used for general use as well to organize personal appointments. This application could also be used by teachers to set a benchmark for how long assignments should take someone to do if they know the material well. Teachers could also make their own calendars with all the due dates for the semester so students can plan out their homework schedule.
In this document there is sections about use, known issues and documentation of this project. There also notes about internal structure to better understand code written besides the comments inside the code.

1.4 Definitions, Acronyms and Abbreviations

TT: Time Taken

Obligation: A time frame that should not be scheduled over

1.5 Document Conventions

Document is written with 1” all around margins in 12-point Arial font.

Sections that have been struck though are goals that weren’t able to be implemented before launch. ~~*Like this*~~

Italicized words are comments

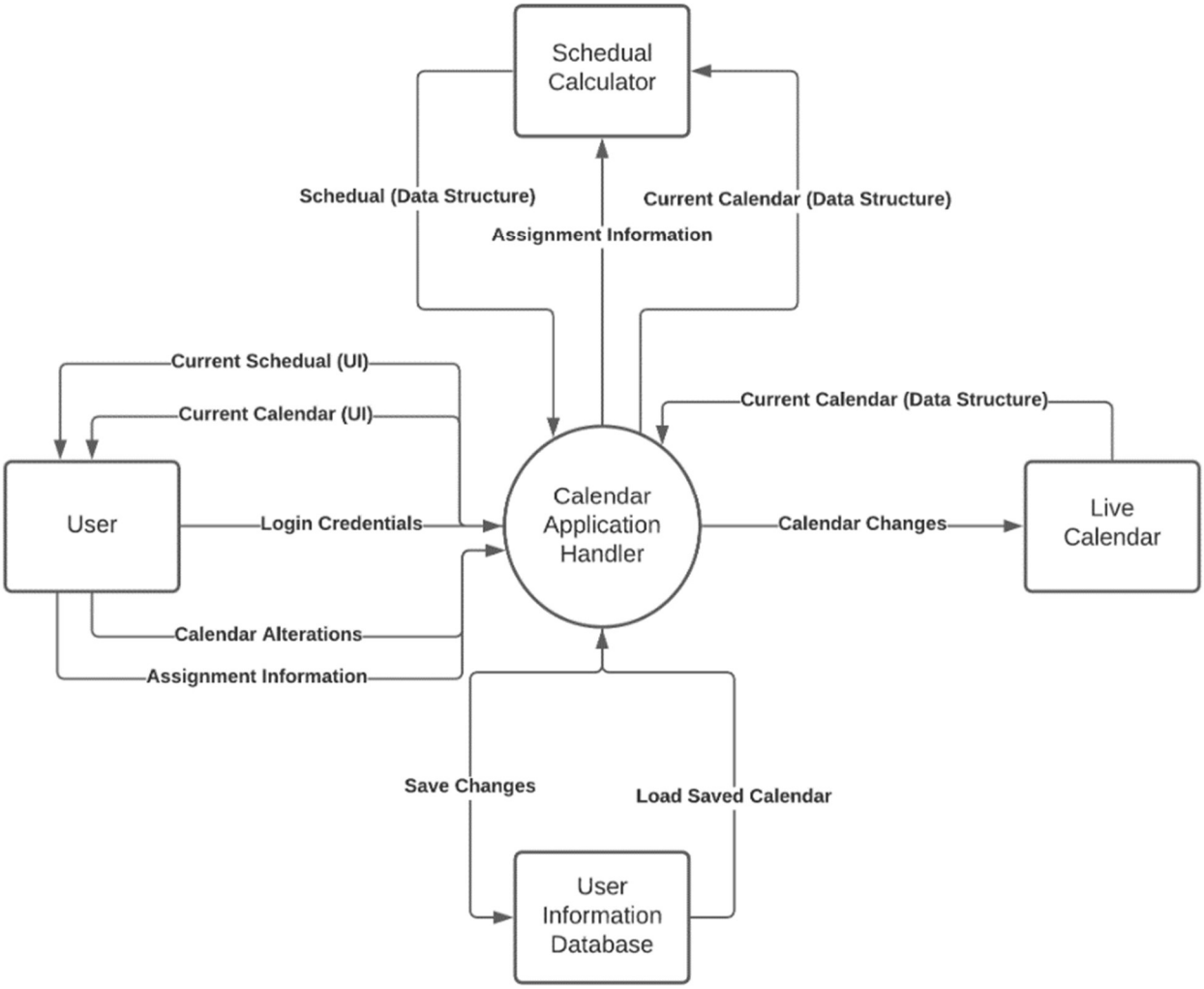
1.6 References and Acknowledgments

Functions pertaining JavaScript were found in Mozilla MDN web documentations found at: https://developer.mozilla.org/en-US/docs/Web/javascript

2 Overall Description

2.1 Product Perspective

The application will be designed as a standalone free use application for the primary userbase of students. The application is not designed to replace or be an extension of any preexisting products. The primary requirements for the functionality of the application will be supplied by the components of the system that store user's current calendar data, store and maintain the current live calendar, ~~and to generate a suggested schedule based on the given user data~~. The components will be managed through a handler that also supplies the requirement of generating a visual interface for the user to interact with the application through.



The diagram above is no longer indicative of the final product. There is no User Information Database, Schedule calculator, or login credentials.

2.2 Product Functionality

- ~~User Account Functions~~
  - ~~Create Account~~
  - ~~Delete Account~~
- Calendar Functions
  - Store events~~, event types, and due dates~~
  - ~~Store "Obligations" that are mandatory block out times~~
- ~~Schedule Functions~~
  - ~~Receive Current Calendar Information to Generate Schedule~~
  - ~~Receive Assignment Information to Discern Priorities~~
- Handler Functions
  - Receive User Calendar Alterations
  - Generate Visual Representations of Current Calendar for User
  - Receive Calendar Load/Save Requests

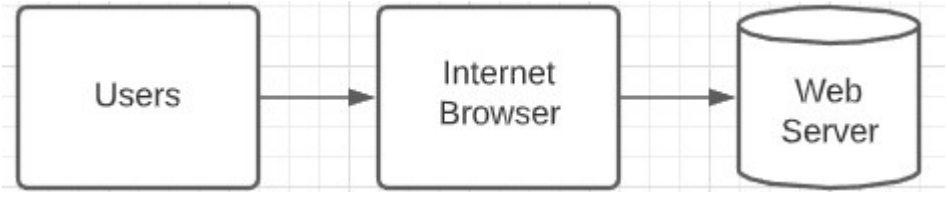
2.3 Users and Characteristics

1. Students:
   The most important users for the application to meet the needs of are students of any level, as such the application will need to be manageable to allow students at any point in their academic career to easily generate a visual representation of their school term to help them succeed. Specific functionalities are designed around the specific needs of different types of students.
   - University Students: Often have busier lives, as such will need to block out certain times in their daily lives for obligations and may not have as much time to thoroughly plan out their schedule. This is the ideal user for the application as they will most likely make up most of the user base and have the highest need for its functionalities.
   - Other Students: Have more free time and fewer obligations, they will demand less from the applications total functionalities and may not need its use at all. However, the application still can offer value from its ability to show representation for any tasks or deadlines that need to be completed.

2. Employees/Workers:
   These users will most likely not need the application due to more strict schedules within their daily routines. However, the application can be used to maintain a healthy work/life balance.

2.4 Operating Environment

The system will be a JavaScript program ~~that will be maintained and run through a web server,~~ as such it will be able to be accessed from most modern web browsers, given that they have the capability to present and run the UI the application will be generating. This will most likely be in the form of HTML. *This project had to shift from a web server to local storage.*



2.5 Design and Implementation Constraints

The main constraint for the project will be time. As the developers of the application will be learning to interact with the technologies that will need to be implemented to achieve the end goal as the application is being developed. These technologies include HTML, ~~web browser functionality, JavaScript server functionality~~, etc.

~~A secondary constraint will be that of server hosting, as the user information will need to be stored on an external host in the form of a database, there can be difficulty in finding a host that suits the needs of the project.~~

2.6 User Documentation

There will be a basic user tutorial ~~upon account creation~~, as well as a user manual that can be consulted at any time within the application itself. These will cover the basics for interacting with the applications functionalities from adding events and their information to ~~saving the calendar to the database~~. *Saving is done automatically and Locally*

2.7 Assumptions and Dependencies

The main assumptions for the project are that each user will have at the minimum an empty calendar being stored in their ~~user account~~ *Local storage*, however the amount of events being stored within a single calendar can be infinitely scalable, as such the calendar will most likely need to only keep track of upcoming dates when it comes to scheduling, as well as other user information that is applicable. Past events should be simplified to reduce their complexity when being viewed later.

Users will be using a current web browser, such as chrome or edge.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Upon the application being started, the user will be prompted with ~~a login screen~~ *Landing page with instructions*, ~~this will ask for a username and a password to be filled in within their respective text fields, additionally there will be a button that allows the user to create an account with the same text fields.~~

Once the user has logged into the application, they will be presented with their most current calendar that was saved to ~~the database~~ *Local storage* in the center of their screen. They will have options to generate a schedule, view the tutorial, add an event, ~~or change the current viewed timeframe along an options bar that will be on the edge of the screen.~~

Upon generating a schedule, the calendar will present the given ~~week~~ *Month* with any available times blocked out and color coded based on the given events and deadlines as well as an option to go back to the main application view.

The tutorial will ~~bring up a series of text boxes~~ *be filled* with useful information to manipulate the application.

Adding an event to the calendar will allow the user to input general information about the event through a series of checkboxes, sliders, and textboxes. This includes whether the event is an ~~obligation or deadline, its priority~~, and any notes on the event the user would like stored.

3.1.2 Hardware Interfaces

Depending on whether the user is logged into the application through a device with a touchscreen or mouse and keyboard, the application will need to be able to receive both inputs through means of basic click functionality within HTML.

3.1.3 Software Interfaces

~~The main software interface will be that of the database used for storing the user accounts and data, this will most likely be run from a private Linux host.~~
On the user's end, the applications software interface will be limited to that of the web browser and loading the html visual representation ~~for the deliverables from the server.~~

3.1.4 Communications Interfaces

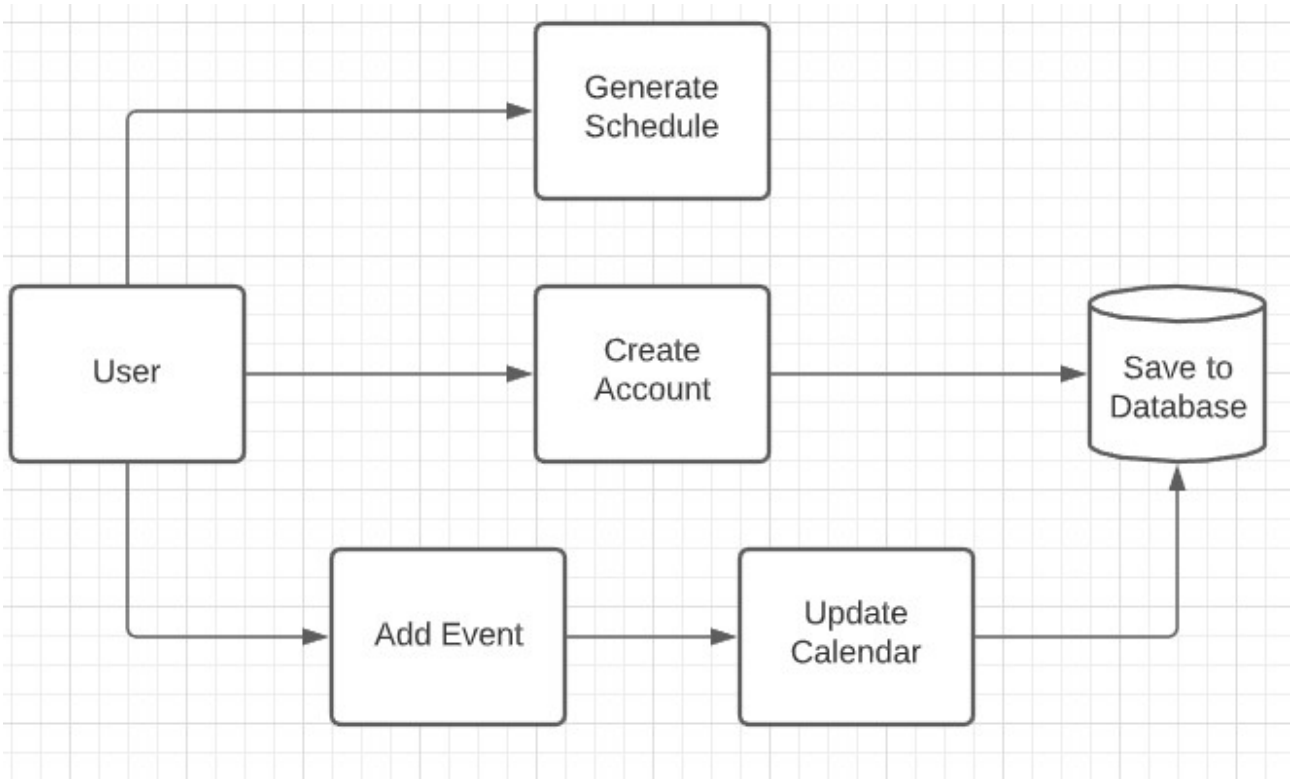
The communication requirements associated with the application will be the web browsers ~~connection to the server~~.

3.2 Functional Requirements

- ~~User Account Functions~~
  - ~~Create Account~~
    - ~~name, classes, start/ end dates, work information~~
  - ~~Delete Account~~
    - ~~Remove data in create account~~
- Calendar Functions
  - Store events, ~~event types, and due dates~~
    - Store date, time~~, locations~~
    - ~~Assigns level of priority, based on what user defined in creating an account.~~
    - Store dates for homework/quizzes
  - ~~Store "Obligations" that are mandatory block out times~~
    - ~~Define what is a block out time~~
      - ~~I.e. what time work is/ what time class is~~
- ~~Schedule Functions~~
  - ~~Receive Current Calendar Information to Generate Schedule~~
    - ~~When an event is created, it generates the list at the same time, with all events in chronological order.~~
  - ~~Receive Assignment Information to Discern Priorities~~
    - ~~Puts priorities in order based on the order defined in create account.~~
- Handler Functions
  - Receive User Calendar Alterations
    - If there is an alteration event gets removed or added, then the calendar will adapt.
  - Generate Visual Representations of Current Calendar for User
    - There will be a function that handles assigning icons to each category if the user specified an icon.
  - Receive Calendar Load/Save Requests
    - If there is a load/save request, returns request.

3.3 Behavior Requirements

3.3.1 Use Case View



In the diagram the user is the sole actor, they have the following use cases:

- ~~Create an account: creates an account and adds it to the database~~
- Add an event: adds an event that updates the calendar, ~~pushing all changes to the database~~
- ~~Generate schedule: this generates a schedule on the users end~~

4 Other Non-functional Requirements

4.1 Performance Requirements

- Times recorded should not overlap each other, for example two assignments that are being recorded for time should not be in the same category.
- The program should not take more than 2ms to start a timer.
- There should not be an excess number of bytes for one event. I.E 2 bytes is acceptable, 100 is not for one calendar event.
- It should not take more than 2 seconds to input a new calendar event into the main calendar.

4.2 Safety and Security Requirements

~~A password will be needed to access the calendar when logging in. This would be to ensure that a student maintains their own privacy. This also gives a piece of mind for the student knowing no one can access the calendar but them.~~

4.3 Software Quality Attributes

The system will be maintainable by using functions that are simple and not super complex. This allows for future changes to be made easier because the function calls are precise and has the purpose in the function name. An example would be eventCreate; this would have calls within this function to add details to the event. This makes it easier to see what function is causing an error using print statements. There will be print statements after each call, commented out, for ease of seeing input/output of functions.

5 Other Requirements

Appendix A – Data Dictionary

<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>

Appendix B - Group Log

First meeting notes 10/21/2020

Specs-

- Users can set blackout times on their calendar
  - Set events that conflict with blackout times are given +1 priority
    - This is so assignments can be completed before a work shift/class/etc...
- Tracks set events
  - Events can be categorized by class/subject
  - Events can be created/deleted/marked as complete
  - Events will have priority levels
  - Events that conflict will be marked
- Creates an itinerary
  - Organizes events based on importance/priority/completion time
  - Schedules around blackout times
- Calculates completion times
  - Upon completing an event, prompts for time spent
  - Takes average times for event type/subject into account
    - Math assignments may be quicker than physics, etc...

Second meeting notes 11/30/2020

Specs –

Changed the scope of the project due to new time constraints.

Third meeting notes 12/07/2020

Specs –

Decided on Time of Presentation