## **SW Engineering CSC 648/848 Section 4**

## **Milestone 2 Submission Form**

**October 17, 2022**

**MyDay**

**Team 7**

**Nya Avelina Bautista – Team Lead**

**Paula Abigail Tam – Front End Lead**

**Christine J’Ursey – Back End Lead**

**Henry Cai – Github Master**

**Francis Ranallo – Scrum Master**

**History Table**

| **Revision ID** | **Revision Date** | **Revised By** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Content**

[1. Data Definitions V2](#_cdlp049j9vt7)

[2. Functional Requirements V2](#_rn6ehdpzifgo)

[3. UI Mockups and Storyboards (high level only)](#_re1cwf5ppswj)

[4. High-Level Architecture, Database Organization](#_ci51j566sahq)

[5. High-Level UML Diagrams](#_5xt9bhdeljhb)

[6. Key Risks](#_l5yrj2y0mxt9)

[7. Project Management](#_ihq1pljytl1d)

# 1. Data Definitions V2

| Primary Data | Sub-Data | Usage |
| --- | --- | --- |
| Users | * *sfsu id* * *password* * *first name* * *last name* * *major* * *minor* * *events* * *goals* | The user is the person using the app, we want to let users have a personalized MyDay so we need to keep track of different users. A password is set to stop other users from editing or viewing other’s MyDay. Events that are created by the user, such as class, work, or other, are used to populate their schedule and display it for them in an organized way. If an event has a location on the SFSU campus MyDay will show where their current event is located on the map and where their next event will be. Goals are what a user wants to do but doesn’t have a specific time or date they want to do them. All these attributes will be used to make suggestions for the user to help them through their day or week. |
| Time mapping of SFSU locations to other locations (i.e. business building to student center 5-minute walk) | -start location  -end location  -time | This is used to track how long it takes to get from one place to another in order to make sure that suggestions to a user are reasonable and fit a user’s schedule. For example, if a user has a half-hour break between classes a suggestion requires going to a building that is thirty minutes away from the first class and ten minutes away from the second class the student would be late for class following this suggestion. A list of locations will be kept for users to select from and the Time Mapping will keep the time it takes to go from one location to another. |
| Places of Interest | * *name* * *location* * *price* * *type* (food, club, event…) * *average time to complete* (how long it takes to order…) * *open time* * *close time* * *date* | Things that may be suggestions for a user. Location and average time to complete is used to make sure a user has the time to visit a place of interest. Price to show how expensive it is to do this for tailoring suggestions. The type will help tailor suggestions to match a user’s preferences. Open and close time so a user doesn’t take go when it is not open. Sometimes the places of interest will only be available on a certain day, which is why a date is included. |
| Events | * *name of event* * *start time* * *end time* * *days* * *start date* * *end date* * *location* * *tags* * *preferences* | Events, classes and work schedules may change, so they need to be adjustable. Start time, end time, and date show the time slot that the event takes up on that date. Location: an on-campus location or off-campus. Time to reach is for off-campus locations to make sure suggestions will fit a user’s schedule. Preferences help tailor suggestions to a user. |
| Preferences | * *amount* (amount of time: “2” hours, “7” minutes) * *timeUnits* (minutes, hours, days) * *relative* (before or after the related event) * *eventid* (The other event that is being referenced) * *location* (The other location that is being referenced: coffee shop) * *repeat* (boolean value: yes or no) * *days* (What days does this repeat: MTWThFSSu * *numberOfTimes* (how many times should this happen: “2” times a week) * *everyTimeUnit* (every week, month, year: 2 times a “week”) * *what* (coffee, pasta, gym, see event, see location) * *when* (how long, before, after this event) * *repeat* (never, always, once, on Mondays) | Preferences are a way to tailor suggestions to a user’s preference. They are a part of events so a user can decide what they want to do before or after that event. The user can specify what is it they wanted to do relative to the event the preference is tied to, like a general desire such as coffee or food, or a more specific preference like what events should be scheduled near this event. After specifying what it is they want to have near the event, they need to input when that should happen relative to the event. Repeat is how often you want to repeat a type of suggestion relative to a class. |
| Goals | * *type* * *priority* * *event* * *recurring* * *status* | Goals are for certain things that you need or want to accomplish but don’t know exactly when that will happen. Type is what kind of thing a user wants to do, like have coffee. Priority if it can only happen once a semester, like a school event like a homecoming, on the day it happens the priority goes up. Event allows it to be connected to specific events. Recurring is how often you want to do a goal. Status shows how often a goal has been accomplished. |

# 

# 2. Functional Requirements V2

Priority Levels:

* 1 – *must have*
* 2 – *desired*
* 3 – *opportunistic*

*R.1. Scheduler: Priority Level – 1*

R.1.1. - Users can input their class, work, and other events to be organized and displayed.

R.1.2. - Events can be singular or recurring and can be viewed and organized in a convenient fashion.

R.1.3. - Users will be able to filter out the criteria they would want to implement in their schedule.

R.1.4. - Users will be able to add or change events.

R.1.5. - Users will be able to categorize their events (ex. Class Schedule, Work Schedule, etc.)

*R.2. Accounts: Priority Level – 1*

R.2.1. - Users can make their own account and register using their SFSU ID, Name,

Major, and Minor.

R.2.2 - Users are able to log in with their email id and password

R.2.3. - Users are able to log out.

*R.3. Dark Mode: Priority Level – 3*

R.3.1. - Users set the interface to dark mode to make MyDay’s use in dim light/darkness more convenient.

*R.4. Map: Priority Level – 1*

R.4.1. - Users can view a map of SFSU that shows where their classes are.

R.4.2. - Users are able to view locations to visit in between classes to rest/eat/study/socialize.

R.4.3. - Users will be able to see the distance between their classes based on their schedules. For example, if Bella Jones has a class in the Business Building, then MyDay will calculate the distance from the Business Building assuming she is in that location.

*R.5. Suggested Events: Priority Level – 1*

R.5.1. - MyDay can suggest events or goals at a certain time if there is availability in the

user’s schedule.

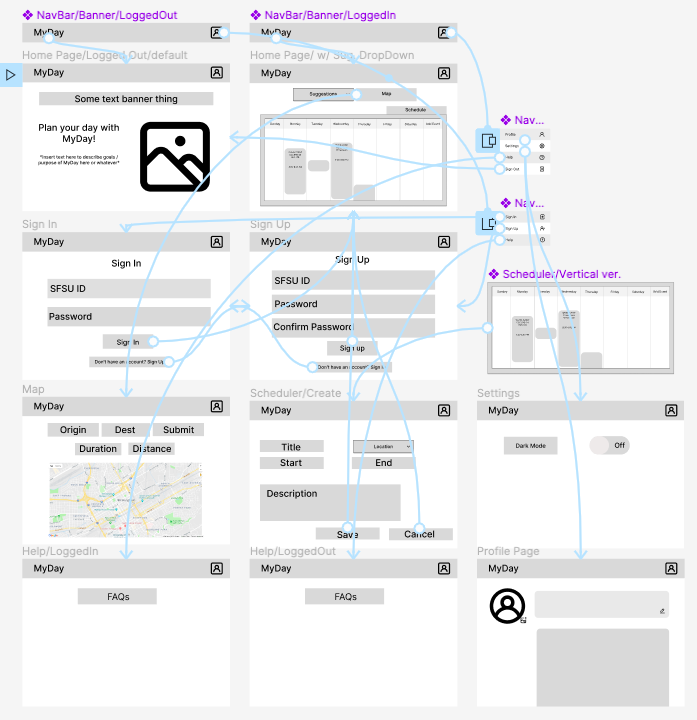
*R.6. Notifications: Priority Level – 3*

R.6.1. - MyDay can send notifications to the user before a certain event, or whenever the user sets a notification to be sent

*R.7. Display: Priority Level – 2*

R.7.1. - Users will be able to view their schedules in daily, weekly, and monthly formats.

# 3. UI Mockups and Storyboards (high level only)

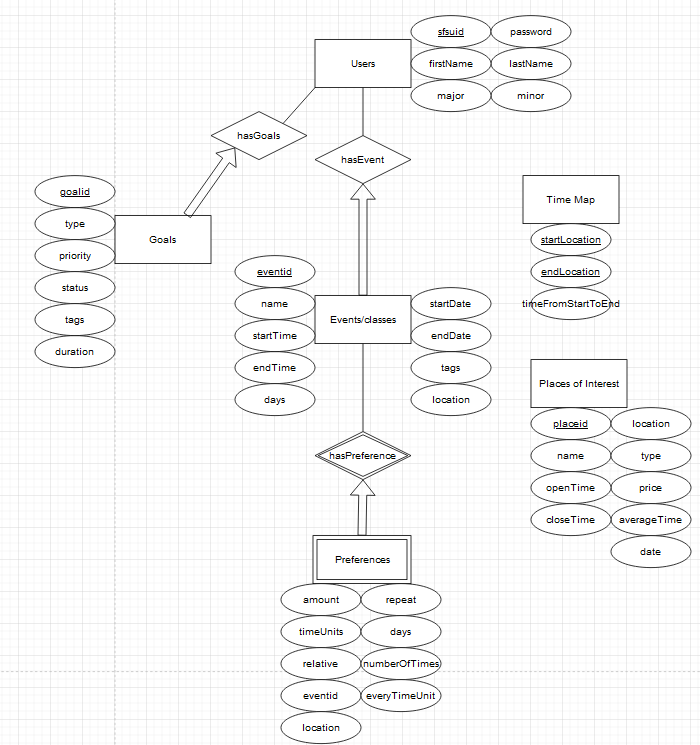


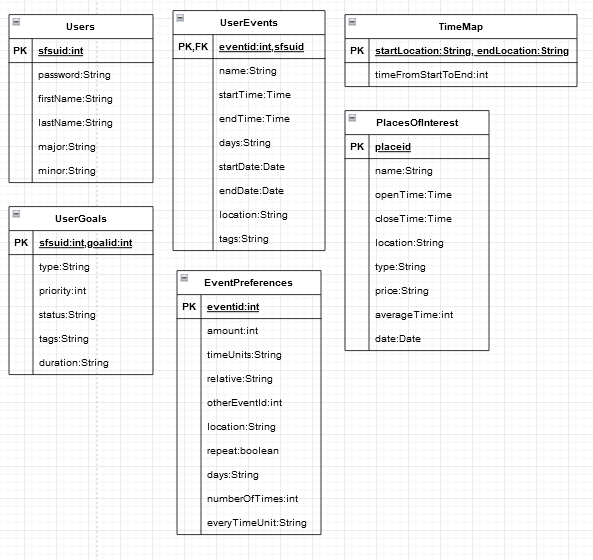
[*UI Mockups and Storyboard link*](https://www.figma.com/file/ZglbhZgXmeMTH8CCtL3QYa/M2-Storyboarding?node-id=0%3A1)

# 

# 4. High-Level Architecture, Database Organization

[*Database Organization Diagram and Tables*](https://drive.google.com/file/d/15K6iFpKArMBuyt34wNnftUj6I3hYt-1M/view?usp=sharing)





| Add/Delete/Search Architecture | Functional Requirement |
| --- | --- |
| *Add/Delete/Search for Users* | User registration, login, and sign out |
| *Add/Delete to Events* | When the user adds or deletes events in the Schedule |
| *Display to Events* | When the user wants to change the view of the Schedule |
| *Search to Time Map* | Search locations to Time Map |
| *Add, Delete to Goals* | When the user wants to set a goal and when the goal is accomplished |
| *Search to Places of Interest* | Search for recommendations to fit user’s schedule |

* Team’s APIs: describe and define at high-level major APIs that you will create among your modules.
* Sign In
* Sign Up
* Sign Out
* Add, Delete, Update, Read Event
* Add, Read User
* Add, Delete, Update, Read Time Map
* Add, Delete, Update, Read Time Goals

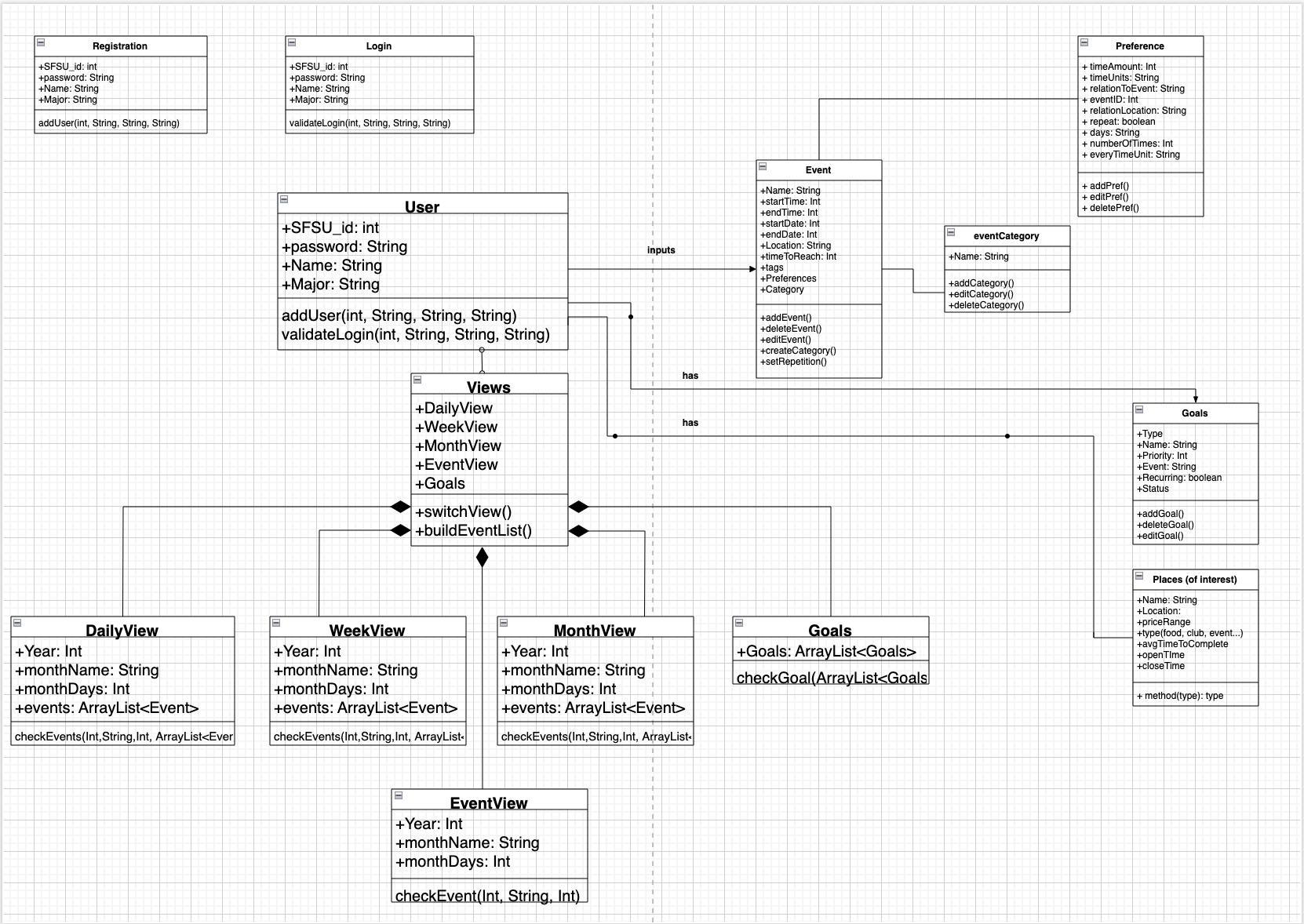
* HTTP Request and Responses defined across backend & frontend:
  + GET, POST
* Route functions are needed to be implemented:
  + GET, POST events
  + GET, POST goals
  + GET, REQ locations
  + GET, POST places of interests
  + GET, POST preferences

* 3rd party API’s, please describe them in your architecture:
* Google Map API: The display of the location for the time map, used by the front end for visual display of the locations that are a part of the user’s events
* Postman: Used to create, share, test and document APIs, both by the front and back end, to ensure the site is functional
* Open-Source Components, please describe them in your architecture:
* Npm scheduler module: Used by the front end to organize events and easily display them for the user

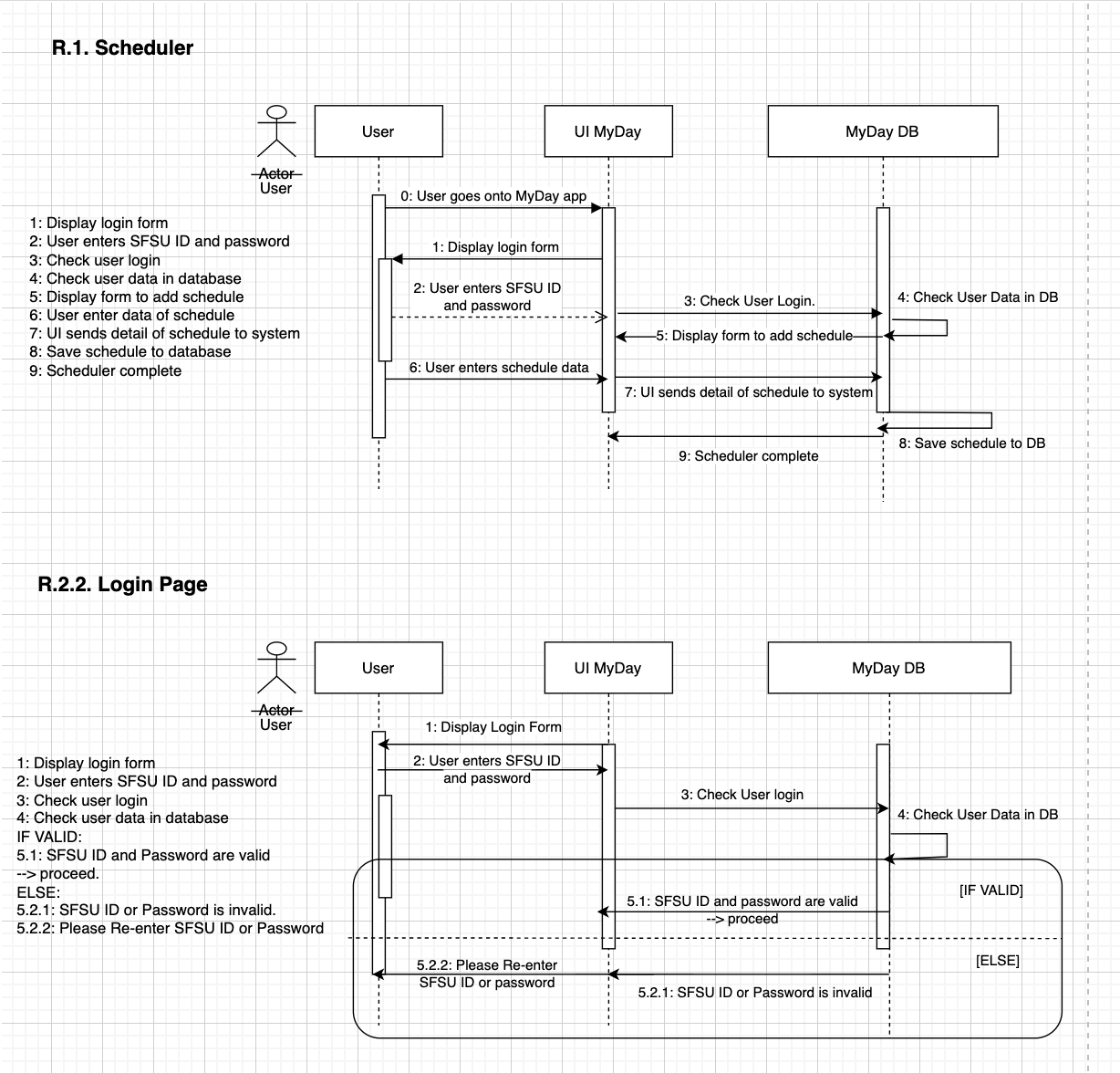
# 

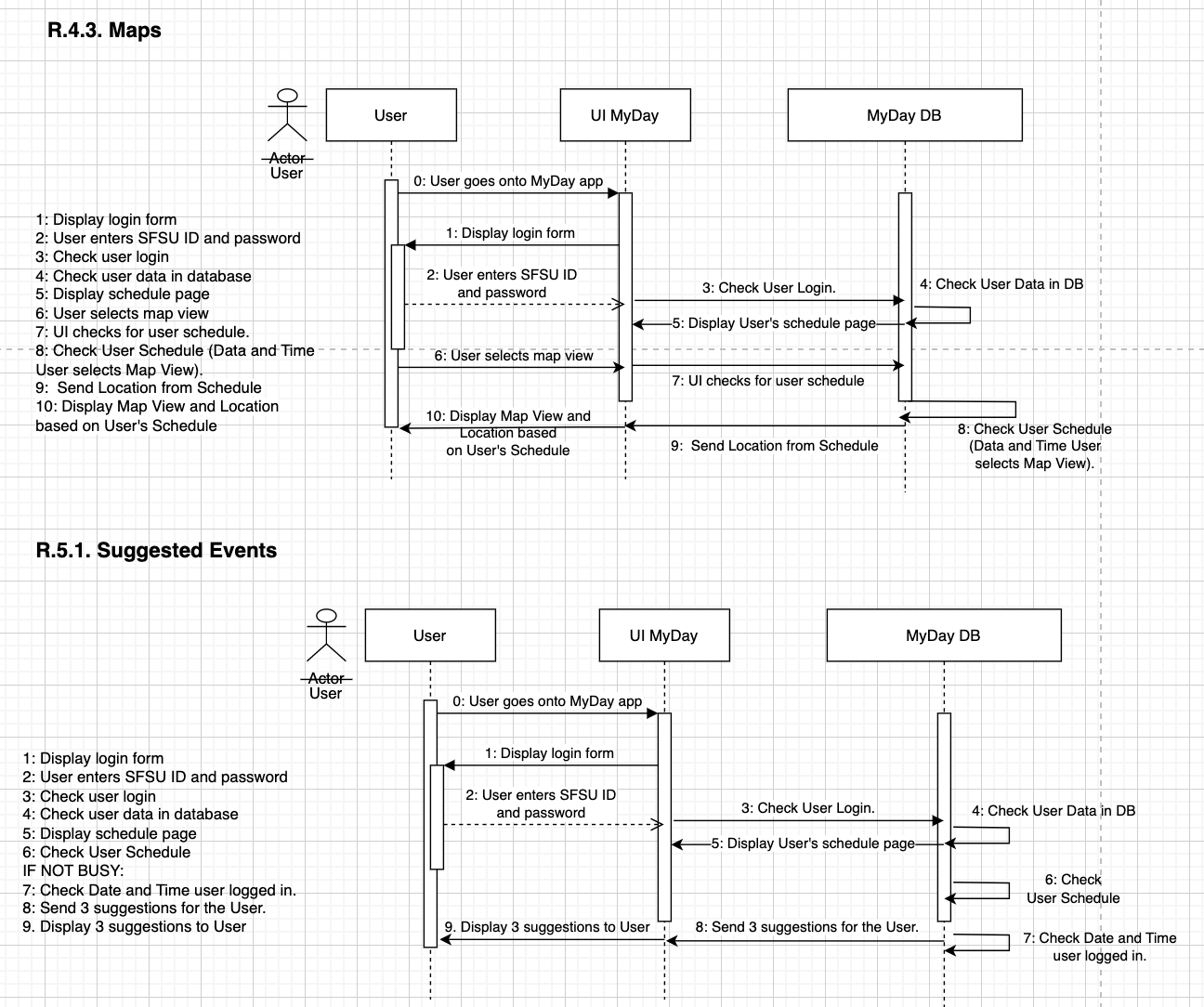
# 5. High-Level UML Diagrams

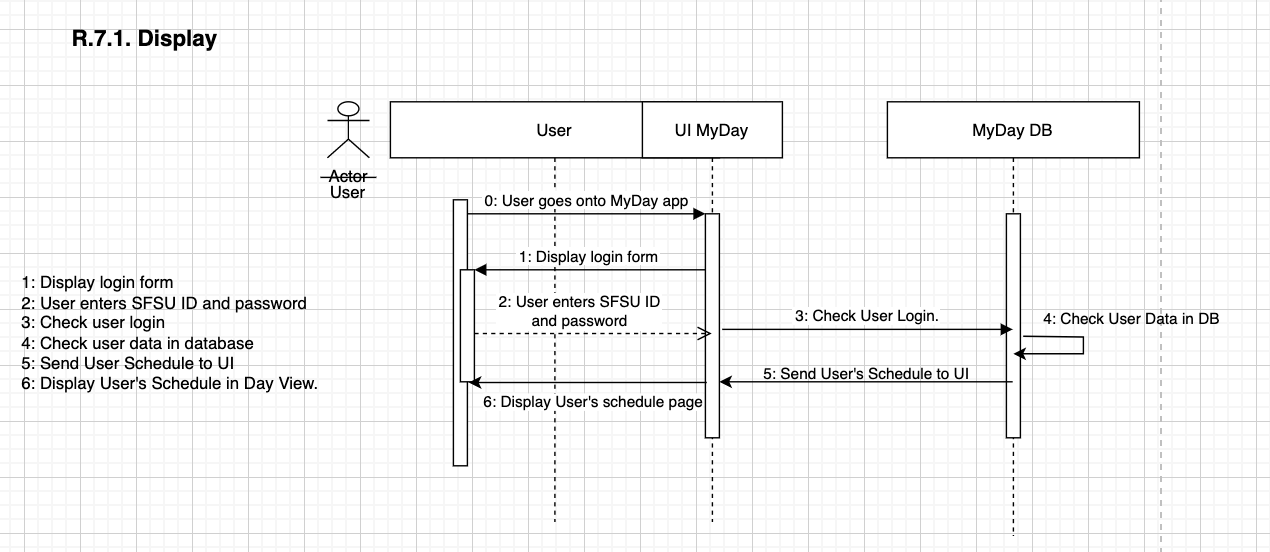
[*High-level UML class diagrams*](https://drive.google.com/file/d/1XV1Dse8rQJbZy-DikRkj01jJqSgsaauX/view?usp=sharing)



[*High-level sequence diagrams*](https://drive.google.com/file/d/1Wo7IiLLXr74ljF-zX7Kdg9Xn4iSZ3qBB/view?usp=sharing)







# 6. Key Risks

One risk that could impede our project at this time would be balancing the workload between teammates with our schedules outside of this class. In order to address this issue, we need to have constant communication throughout the week to let the other team members know if the other team members could pick up an additional task. This would benefit the team in order for us to not be behind in our project.

Another risk would be inputting the school’s entire database for a class schedule. To resolve this conflict, we would have our application be dependent on the input of the user and we would add what is inserted into our database and use the user’s data.

Another risk that could impede our project is not knowing how to accomplish certain functional requirements due to skill sets, etc. To mitigate this problem, we will create study plans that team members can work on in addition to working on our project tasks. We will ensure that every team member is on the same page to be knowledgeable of how each function in our project works.

# 7. Project Management

Outside the scrum meeting, do you have a tool to manage each member’s task?

In order to manage each Milestone 2 task, we split our team into front-end teams and back-end teams. Since there are five team members, we split our teams as shown in the table below. To make the teams even, the team leader is overseeing each task and picks up any task that each team cannot accomplish throughout the week. We share our updates through our Discord server and keep our [Trello](https://trello.com/invite/b/mkSegpic/311a854c1ca7a59ecb1145637a94f4dc/csc-648-team-7) updated consistently. During our scrum meetings, we show our progress in our tasks, whether updated or not, and we discuss what we could do in order to sufficiently stay on task for our project. Outside of the scrum meeting, we all use Trello and Discord. We mostly use Discord because it is easier for us to communicate through that platform to share the specifics in our updates.

| Front End Team | Paul, Henry |
| --- | --- |
| Back End Team | Christine, Francis |
| Team Leader | Nya |