



Wentworth Institute of Technology
COMP4960 – Software Engineering

Software Design Document
for
SyncedUp
Version 1.1
Finalized on 04-07-2024

Prepared By
Team: JTNE, Project Group 2
Jillian Desmond, desmondj2@wit.edu
Thomas Michael, michaelt@wit.edu
Nilay Patel, pateln19@wit.edu
Fabio Tran, tranf@wit.edu
Instructor: Dr. Gyllinsky

GitHub Link
<https://github.com/teachingworkshops/comp4960-spring-2024-project-group-2-syncup.git>

Submitted on 04-07-2024

Contents

Revision History	3
1. Introduction	4
1.1. Document purpose	4
1.2. Product overview	4
1.3. Product functionality	4
1.4. Definitions	4
1.5. Acronyms and abbreviations	5
2. System requirements	5
2.1. Functional requirements	5
2.2. Non-functional requirements	5
2.3. Other requirements	6
3. System architecture	6
3.1. Overall architecture	6
3.2. Components mapping	7
3.3. Technology stack selection	9
4. System Design	9
4.1. UI	9
4.2. Use Case diagrams or Tables	15
4.3. Class diagram	17
4.4. Sequence/activity diagram	18
5. Others	19
6. Test plan	19
6.1 .Function #1: Adding Event to User Calendar	20
6.2 Function #4: Sending a Friend Request	20
7. References	20
8. Appendix	21
8.1 Comparison to Existing Products	21

Revision History

Date	Version	Description	Author(s)
03-14-2024	1.0	Initial draft	Jillian Desmond, Thomas Michael, Nilay Patel, Fabio Tran
03-21-2024	1.1	Revisions to multiple sections based upon review from Professor Gyllinsky	Jillian Desmond, Thomas Michael, Nilay Patel, Fabio Tran
04-07-2024	1.2	Final revisions	Jillian Desmond, Fabio Tran

1. Introduction

1.1. Document purpose

- This document outlines the design choices of our application, SyncUp, and describes technical details of our project. The goal of this document is to provide implementation details for the development team and other associated parties to create the program described in our software requirements specification.

1.2. Product overview

1.2.1. Problem statements

- 1.2.1.1. Absence of Centralized Planning: Due to busy and dynamic schedules, friends and family members have trouble planning social gatherings with one another and often struggle to accommodate everyone's availability with a lack of a centralized source of information.
- 1.2.1.2. Inefficient Communication: Without centralized planning, reaching out to members individually leads to excess time and effort.
- 1.2.1.3. Overlapping of Events: Struggles in coordination could lead to mistakes and the creation of events that overlap with one another or situations where invitees are unable to attend.

1.2.2. Proposed solution

- 1.2.2.1. An application that people can upload data about their schedule and availability. Other people in one's social network can join and provide their data as well which will be analyzed by the application to suggest recommended dates and times to establish event plans.

1.2.3. Novelty

- 1.2.3.1. The application will have a continuous calendar where users can enter and update their availability as needed.
- 1.2.3.2. The application will allow people to be able to join social networks and see when others are available for potential plans.
- 1.2.3.3. The application will make recommendations to users with regards to date and time for events.
- 1.2.3.4. The application will create and save events into a digital calendar at the request of users.

1.3. Product functionality

- This application should allow people to view a personal virtual calendar and be able to create, read, upload, and delete events as needed. There should be a friendship system for users to establish a social network. These connections will be used to allow affiliated users to view each other's calendars and availability. The application should also analyze data and suggest recommended dates and times to establish plans.

1.4. Definitions

- Native: A native app is software designed for a specific operating system or device.
- Offline: In the context of a work environment, offline means a person or system is not available to be contacted and accessed.

- Solid: is a specification that lets individuals and groups store their data securely in decentralized data stores called Pods [5].
- Pods: are like secure web servers for data where owners' control which people and applications can access it [5].

1.5. Acronyms and abbreviations

- API: Application Programming Interface
- CRUD: Create, Read, User, Delete
- IESS: Inrupt Enterprise Solid Server
- GUI: Graphical User Interface
- MVC: Model-View-Controller
- OAuth: Open Authorization
- SRS: Software Requirements Specifications
- UI: User Interface

2. System requirements

2.1. Functional requirements

- 2.1.1. Upon starting, the system shall be able to login to an existing user account.
- 2.1.2. The system shall be able to connect users to multiple groups.
- 2.1.3. The system shall have a tab that displays all the groups users are connected to.
- 2.1.4. Once logged in, the system shall allow users to create an event on that date, this is unavailable time.
- 2.1.5. Within a group, the system shall have a button for planning a time to meet.
- 2.1.6. Upon selecting the button 2.1.5, the system shall suggest times that every group member is available within a specified time frame. If there is no time window when everyone is available, then the system will suggest alternatives with as many members as possible.
- 2.1.7. When producing meeting times 2.1.6, the system shall order these time windows earliest to latest.
- 2.1.8. The system shall allow users to send friend requests to other users.
- 2.1.9. Upon the recipient user accepting the friend request 2.1.8, the system will make a connection between both users.
- 2.1.10. When two users have a connection 2.1.9, the system shall give both users a button to view the other's availability.
- 2.1.11 The GUI (Graphical User Interface) shall display a friends list upon user request.
- 2.1.12 The GUI shall display an error message when a user attempts to send a friend request to a user that does not exist. It should also give the user the option to retry.

2.2. Non-functional requirements

- 2.2.1. The system shall take Solid login data as input and authenticate via API calls to Inrupt.

2.2.2. Upon logout, the system shall remove login data from memory and return the UI to the login page.

2.2.3. The system should provide real-time updates of the user's calendar or a friends' calendar.

2.2.4. The method for suggesting meeting times within a group will use a greedy activity selection algorithm.

2.2.5. The system shall have an updated view (from user input to controller to database) within 1 second of the user input. Updated view included a loading indicator for processes that may take longer than allotted time.

2.2.6. API integration for backend and front-end interactions. Consecutive requests are needed to ensure any changes to data stored in the Solid Pods are shown on UI.

2.2.7. The system shall be available to use and interact with 24/7, with maintenance exceptions.

2.3. Other requirements

2.3.1. The system will use Java as the primary programming language.

2.3.2. The system will use Solid as its data management platform.

2.3.3. The system will use Inrupt PodSpaces and Inrupt's Enterprise Solid Server as its third party provider for Solid Pods

2.3.4. UI should have intuitive navigation, where a user can utilize all functionalities without a tutorial and within a short time frame.

3. System architecture

3.1. Overall architecture

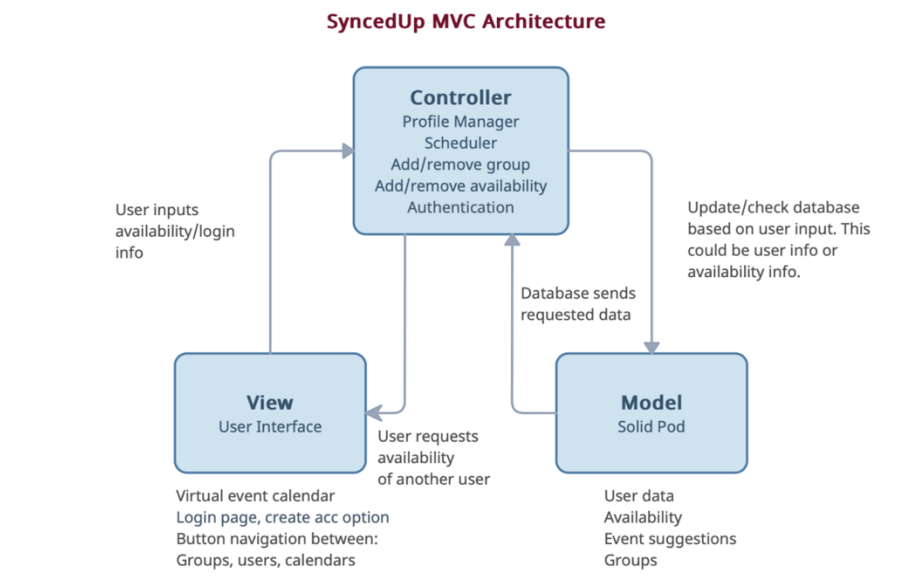


FIGURE 3.1 - General SyncUp Application in the MVC Architecture

MVC Architecture will be the pattern used to create SyncUp, as users will be performing all four CRUD operations on their own data. This design allows the user to do seamlessly on the UI, which will trigger various methods in the controller, resulting in a query, or update to the database.

3.2. Components mapping

3.2.1. Functional requirements

- 3.2.1.1.** Upon starting, the system shall be able to generate a new user account.

Create New Account Mapping

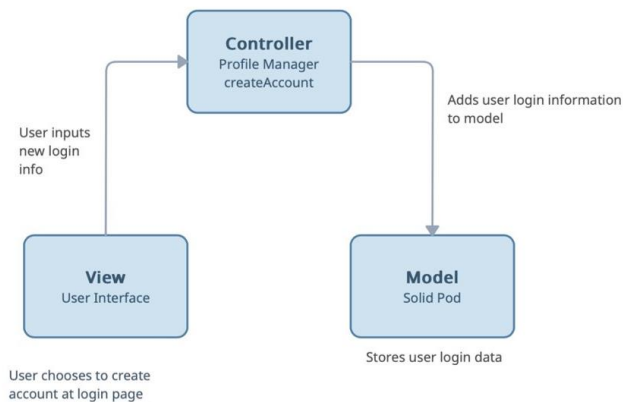


Figure 3.2.1.1 - User Creation in SyncUp using the MVC Architecture

- 3.2.1.2.** Upon selecting the button in 2.1.9, the system shall suggest times that every group member is available within a specified time frame. If there is no time window when everyone is available, then the system will suggest alternatives with as many members as possible.

Generate Meeting Time Mapping

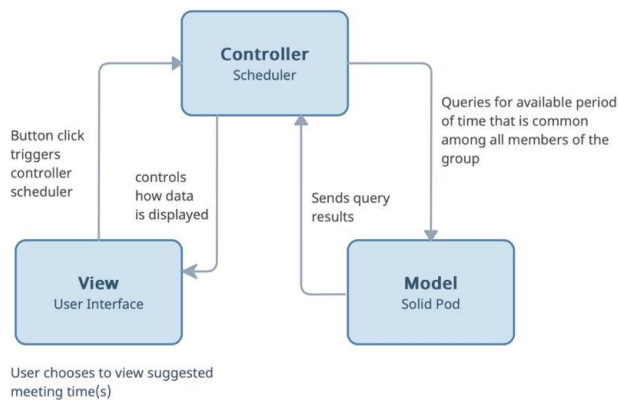


FIGURE 3.2.1.2 – Component mapping diagram of requirement 2.1.10

- 3.2.1.3. Upon account deletion, the system shall delete availability data and reflect that on the UI.

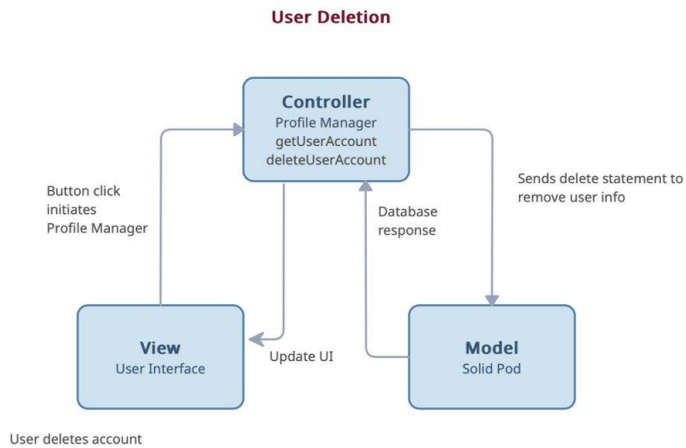


Figure 3.2.1.3 – Component mapping diagram of requirement 2.2.3

3.2.2. Non-functional requirements

- 3.2.2.1. The system shall store login data via database to be queried for comparison with user input to authenticate user.

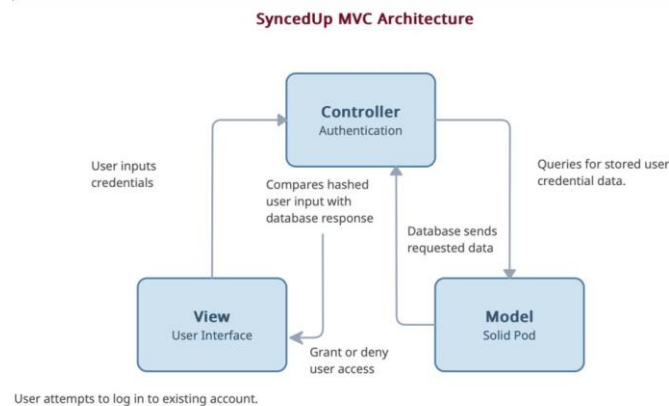


Figure 3.2.2.1 – Component Mapping of requirement 2.2.1

Requirement Number	Mapping (MVC)
2.1.1	MVC
2.1.2	MC

2.1.3	MVC
2.1.4	MVC
2.1.5	MVC
2.1.6	MC
2.1.7	C
2.1.8	MVC
2.1.9	MVC
2.1.10	MVC
2.1.11	VC
2.1.12	MVC
2.2.1	MC
2.2.2	VC
2.2.3	MVC
2.2.4	MC
2.2.5	MVC
2.2.6	MVC
2.2.7	MVC
2.3.1	C
2.3.2	M
2.3.3	M
2.3.4	MV

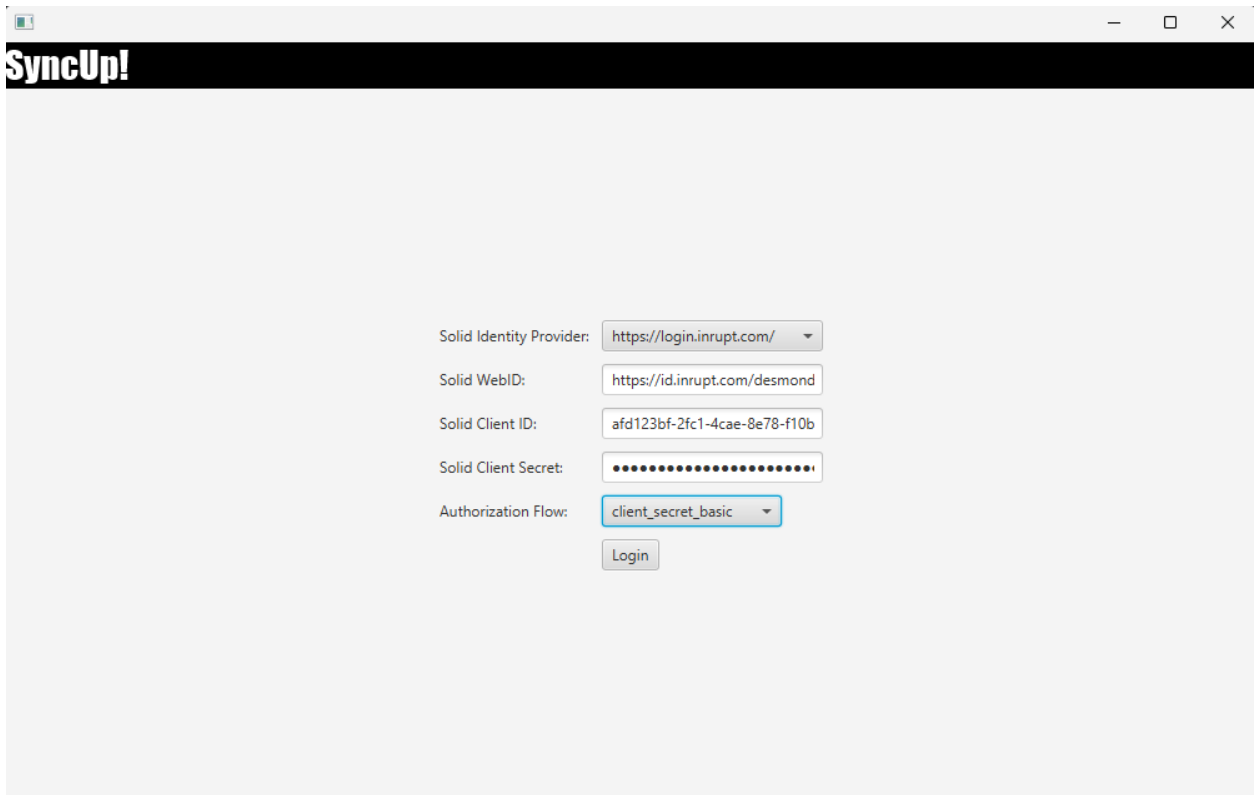
3.3. Technology stack selection

- Solid: A framework/principles designed for a decentralized web in mind that prioritizes users having control over personal data. Different applications can use and retrieve the same data as it is separated from a single entity (organization). Personal online data stores (Pods) store data and share it with applications as needed.
- WebID: A Uniform Resource Identifier (URI) used in Solid to identify a unique user. This is used in authentication and authorization when manipulating data from the Pods.
- API: Application Programming Interface (API) is used for two components to communicate with each other which Solid principles uses HTTP Requests such as GET, POST, PUT, and DELETE. This will be used to access data from Solid Pods and perform CRUD operations.
- Inrupt Java Client Libraries: A library of methods and tools assembled to aid in the development of Solid based applications with the Java programming language.
- JavaFX: A framework for building UI for Java applications. It features many useful tools for creating functional and appealing interfaces.

4. System Design

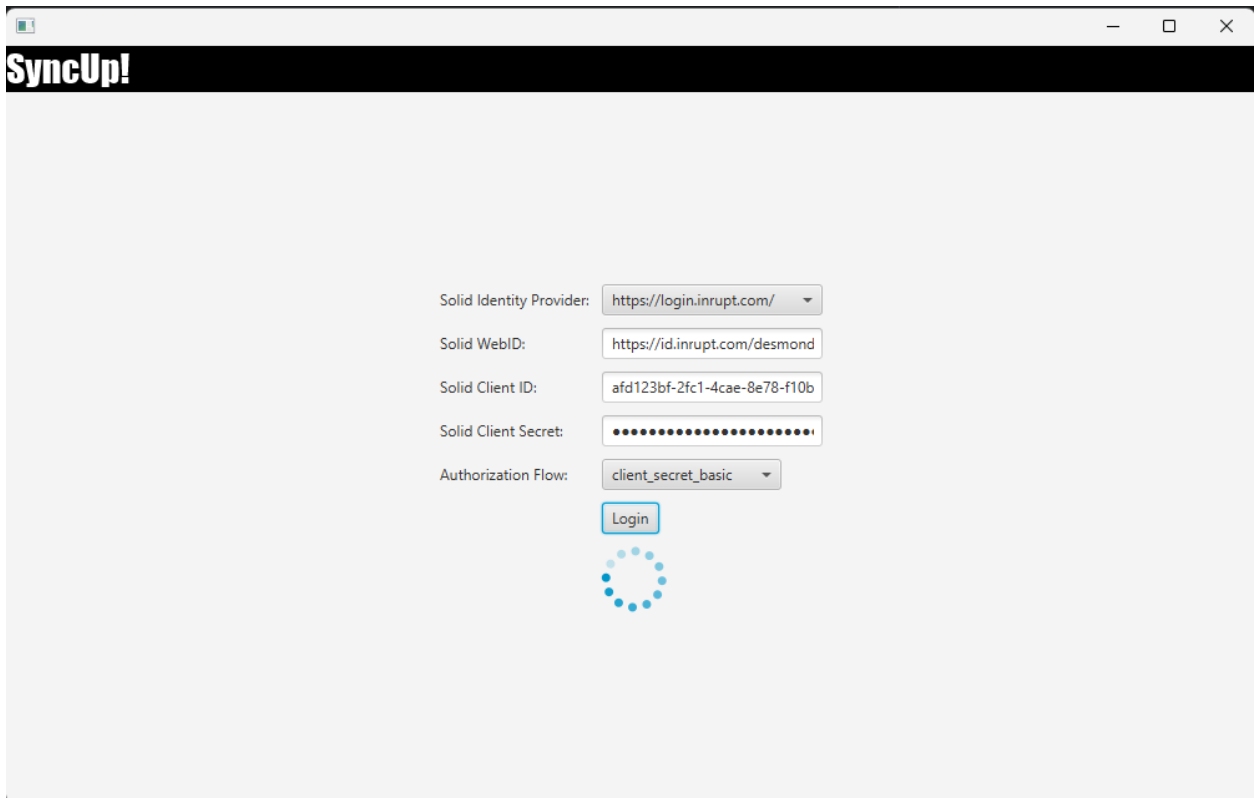
4.1. UI

4.1.1. Function #1: Login Page



The screenshot shows a web browser window with the title "SyncUp!". The page has a black header bar with the "SyncUp!" logo in white. Below the header, the login form is centered. It contains five input fields and a "Login" button. The inputs are: "Solid Identity Provider" (a dropdown menu showing "https://login.inrupt.com/"), "Solid WebID" (a text box with "https://id.inrupt.com/desmond"), "Solid Client ID" (a text box with "afd123bf-2fc1-4cae-8e78-f10b"), "Solid Client Secret" (a text box with masked characters "....."), and "Authorization Flow" (a dropdown menu showing "client_secret_basic"). The "Login" button is a simple rectangular button.

Figure 4.1.1.1: Login Page with example inputs.



This screenshot shows the same login page as Figure 4.1.1.1, but with a loading spinner below the "Login" button. The spinner is a circular arrangement of small blue dots. The "Login" button is now highlighted with a blue border, indicating it has been clicked. The other input fields and the "SyncUp!" header remain the same.

Figure 4.1.1.2: Login Page with loading spinner, the authentication process takes a few seconds.

4.1.2. Function #2: Calendar View

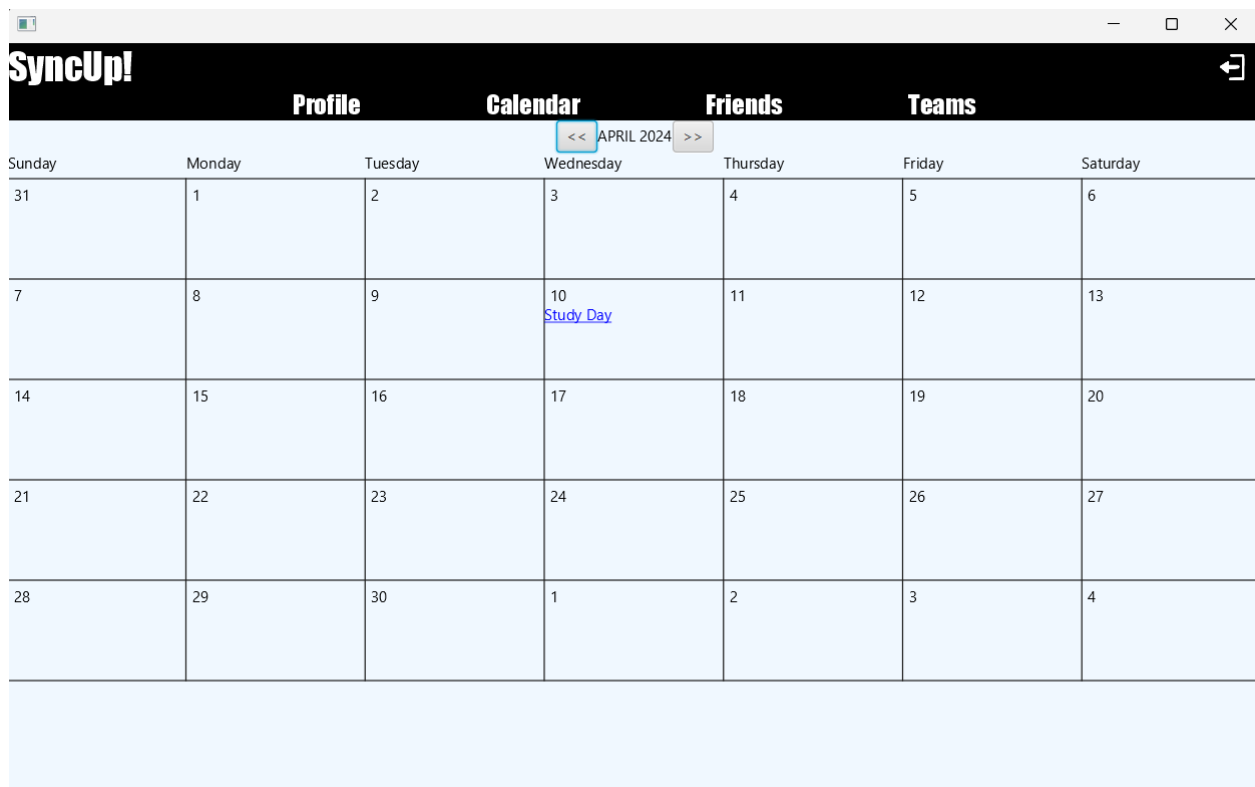


Figure 4.1.2: Calendar View.

4.1.2.1. Subfunction 2.1 View Event Details

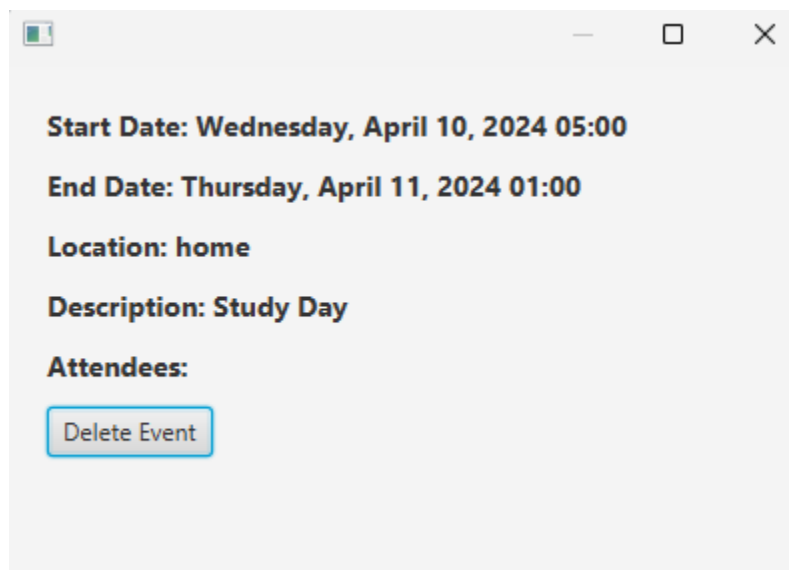


Figure 4.1.2: Event Details View.

4.1.3. Function #3: Profile View

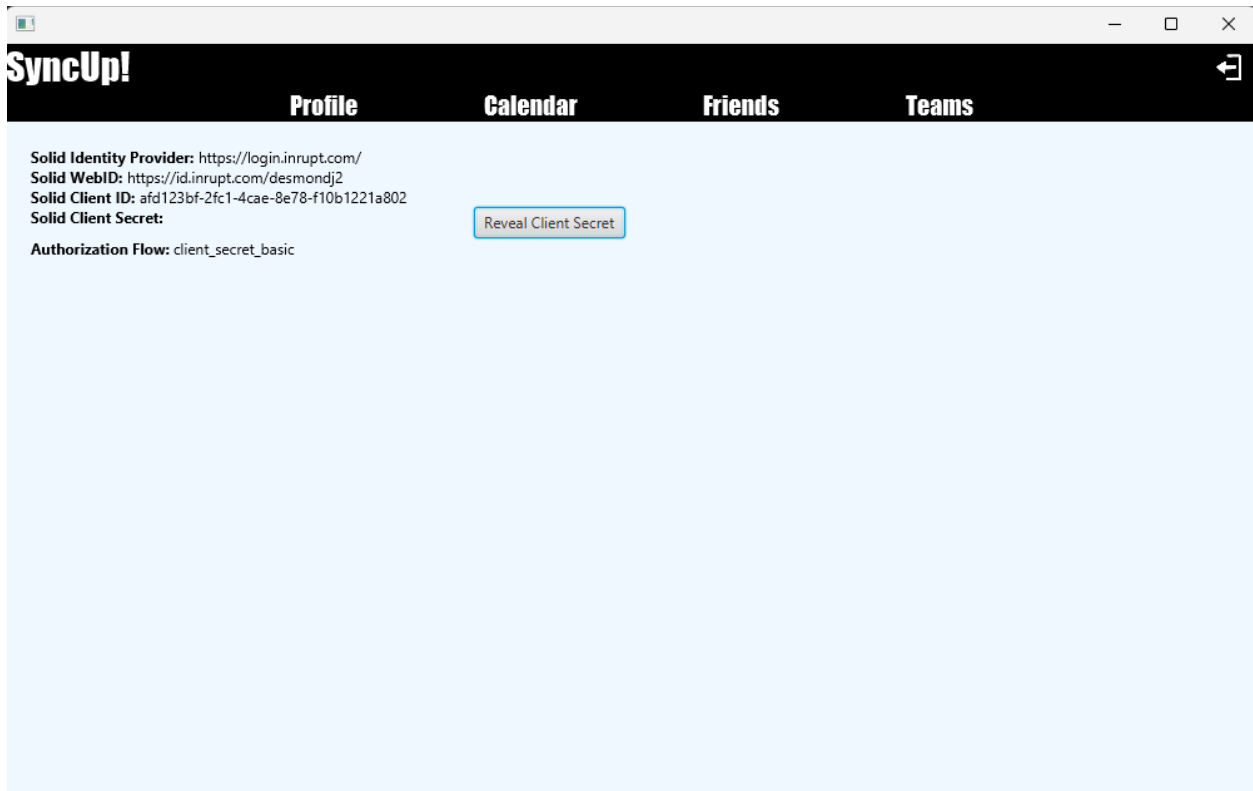


Figure 4.1.3: Profile View.

4.1.4. Function #4: Friends View

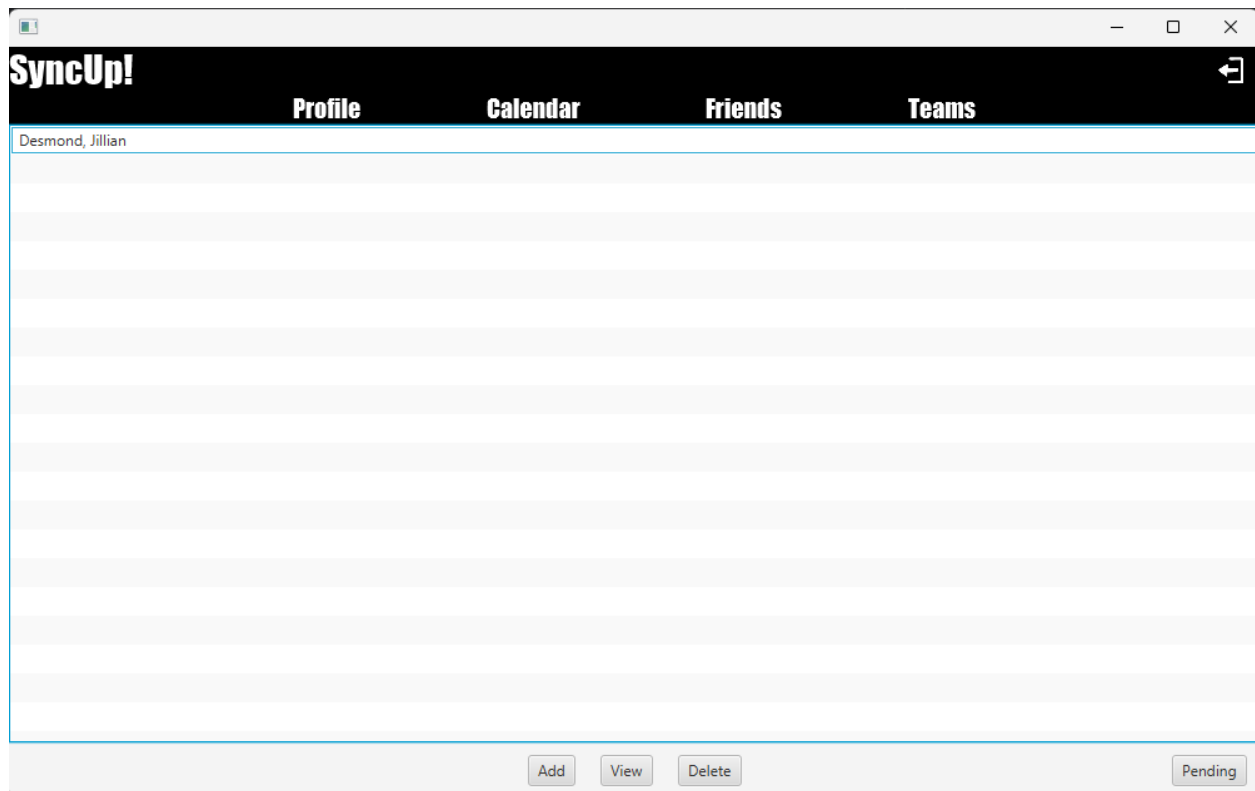


Figure 4.1.4: Friends List View.

4.1.4.1. Subfunction #4.1: Pending Friend Request View

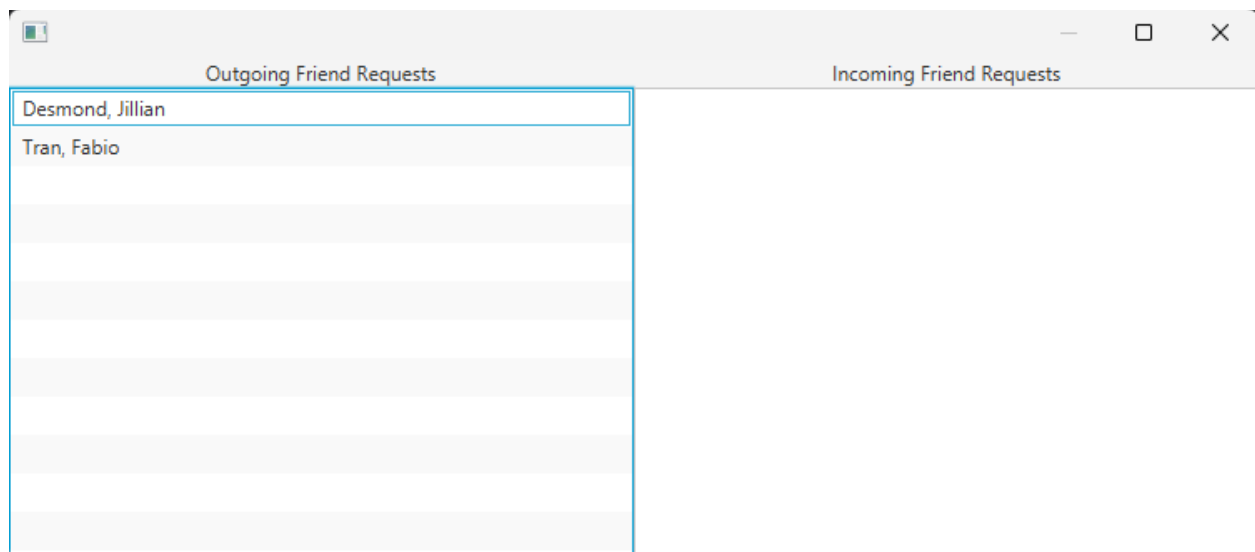
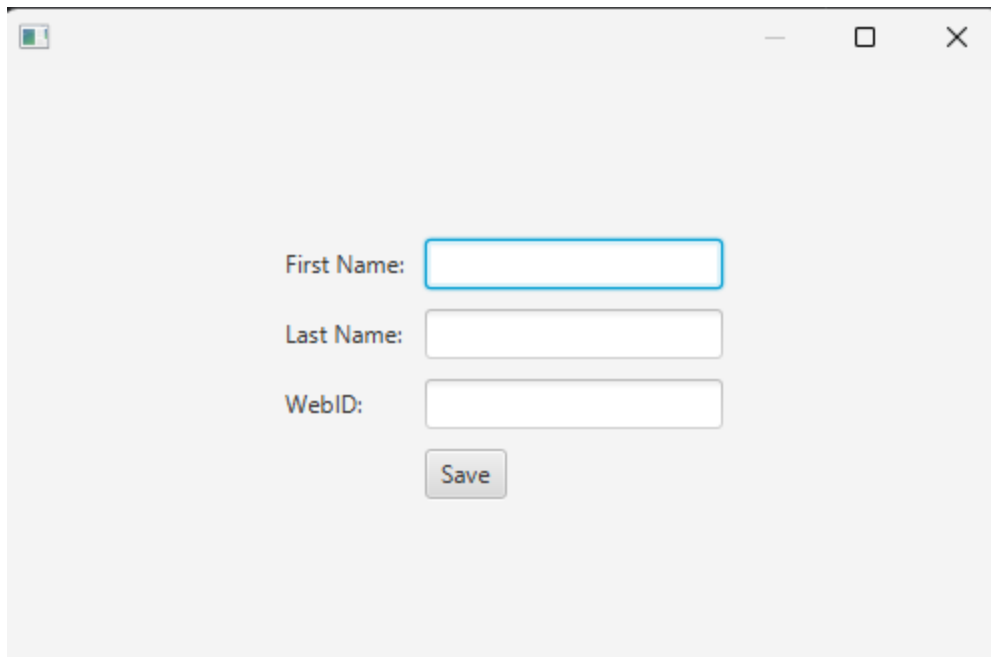


Figure 4.1.4.1: Pending Friend Requests View.

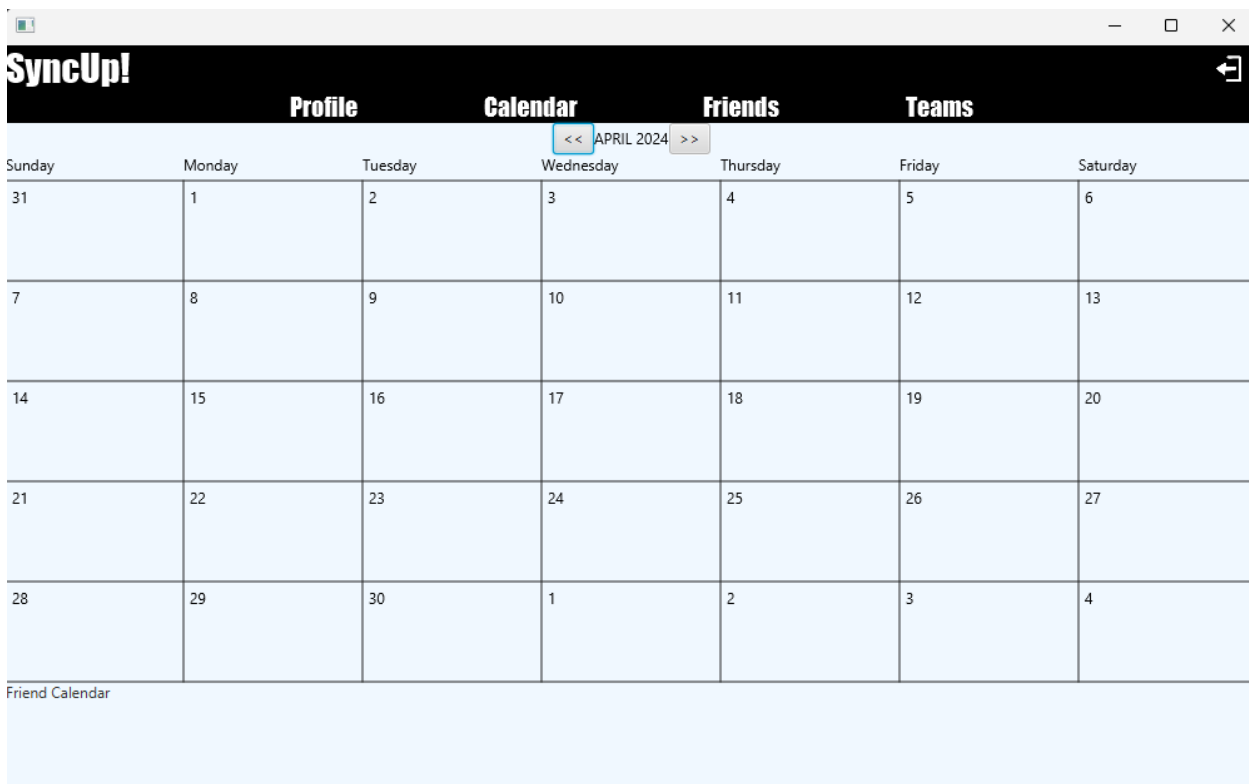
4.1.4.2. Subfunction #4.2: Add a Friend



A screenshot of a web application window titled "Add Friend View". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. The form contains three input fields: "First Name:", "Last Name:", and "WebID:". Each field is a white rectangle with a thin gray border. Below the "WebID:" field is a "Save" button, which is a gray rectangle with rounded corners and the word "Save" in black text.

Figure 4.1.4.3: Add Friend View.

4.1.4.3. Subfunction #4.3: View a Friends Calendar



A screenshot of a web application window titled "Friends Calendar View". The window has a black header bar with the "SyncUp!" logo on the left and a home icon on the right. Below the header is a navigation bar with four tabs: "Profile", "Calendar", "Friends", and "Teams". The "Calendar" tab is selected and highlighted. Below the navigation bar is a calendar for April 2024. The calendar has a light blue background and a grid of days. The days are labeled with their respective numbers (1-31) and the day of the week (Sunday-Saturday). The calendar is currently showing the month of April 2024. Below the calendar is a section labeled "Friend Calendar" with a light blue background.

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

Friend Calendar

Figure 4.1.4.3: Friends Calendar View.

4.2. Use Case diagrams or Tables

4.2.1. Sub function #1

Number	1
System	SyncUp
Name	Subfunction #1: Set Availability
Primary Actors	User
Descriptions	A user can set availability in the system for any day of the year. The user can make that availability repeat daily, weekly, monthly, or yearly.
Preconditions	The user has created an account.
Post-conditions	One or more days have availability assigned to them.
Trigger	User initiates the set availability process
Basic Flow	<ol style="list-style-type: none"> 1. User chooses to set availability 2. Week Calendar pops up 3. User selects a day 4. Input window pops up. 5. User inputs the timeframe they are available for that day. 6. User saves input
Alternate Flow	<ol style="list-style-type: none"> A1.1. User chooses to set availability A1.2.1. Week Calendar pops up A1.2.2. User changes the view to a monthly calendar A1.3. User selects a day A1.4. Input window pops up. A1.5. User inputs the timeframe they are available for that day. A1.6. User saves input
Exception Flow	E 1.1 (2, 3, 4, 5) User decides to cancel the process.
Extensions	<ol style="list-style-type: none"> 5a. User opts for the time window to repeat (options are daily, weekly, monthly, yearly) 6a. User selects another day (repeats steps 3-6, alternate flows, and exception flows still valid).

4.2.2. Sub function #2

Number	2
System	SyncUp
Name	Subfunction #2: Send friend request
Primary Actors	User A, User B
Descriptions	User A sends a friend request to User B
Preconditions	The user has created an account.
Post-conditions	User A and User B are connected as "friends."
Trigger	User initiates send friend request process
Basic Flow	<ol style="list-style-type: none"> 1. User A searches User B by username. 2. User A sends a friend request 3. User B receives request and chooses to accept

Alternate Flow	
Exception Flow	E1.1. (1) There is no user associated with that username E1.2. No results populate from that search E2.1. (3) User B denies the friend request E2.2. Friend request disappears from User Bs display
Extensions	

4.2.3. Sub function #3

Number	3
System	SyncUp
Name	Subfunction #3: Join Group
Primary Actors	User, Group
Descriptions	A user can join multiple groups.
Preconditions	1. The user has created an account. 2. The group exists.
Post-conditions	User is a member of the group.
Trigger	User wants to join a group.
Basic Flow	1. User requests to join a group. 2. A notification is sent to each group member. 3. Any member of the group approves the user request.
Alternate Flow	A1.1. A member of the group sends the user an invite. A1.2. A notification is sent to the user. A1.3. The user accepts the invite.
Exception Flow	E1.1 (3) A member of the group rejects the request to join. E2.1 (A1.3) The user declines the invite.
Extensions	

4.2.4. Sub function #4

Number	4
System	SyncUp
Name	Subfunction #4: Create Event
Primary Actors	User
Descriptions	Create an event within a team availability
Preconditions	1. The user has joined a group 2. The members of the group have added their availability
Post-conditions	Each team member has a new event blocked off on their calendars
Trigger	User wants to plan a time to meet
Basic Flow	1. User selects the option to find a time to meet 2. System searches for date(s) and time(s) that all group members are available. 3. System returns meeting time options.

	<p>4. User selects one</p> <p>5. Event is added to the team members' calendars as unavailable.</p>
Alternate Flow	<p>A1.3. System could not find a time that all team members are available</p> <p>A1.4. System asks the user if they would like to find a meeting time with not all team members</p> <p>A1.5. User selects option to continue without all team members</p> <p>A1.6. System returns a meeting time with as many team members as possible.</p> <p>A1.7. User approves meeting time, and the meeting is added to the selected team members' calendars</p>
Exception Flow	E1.1 (A1.5.) User declines to continue.
Extensions	

4.3. Class diagram

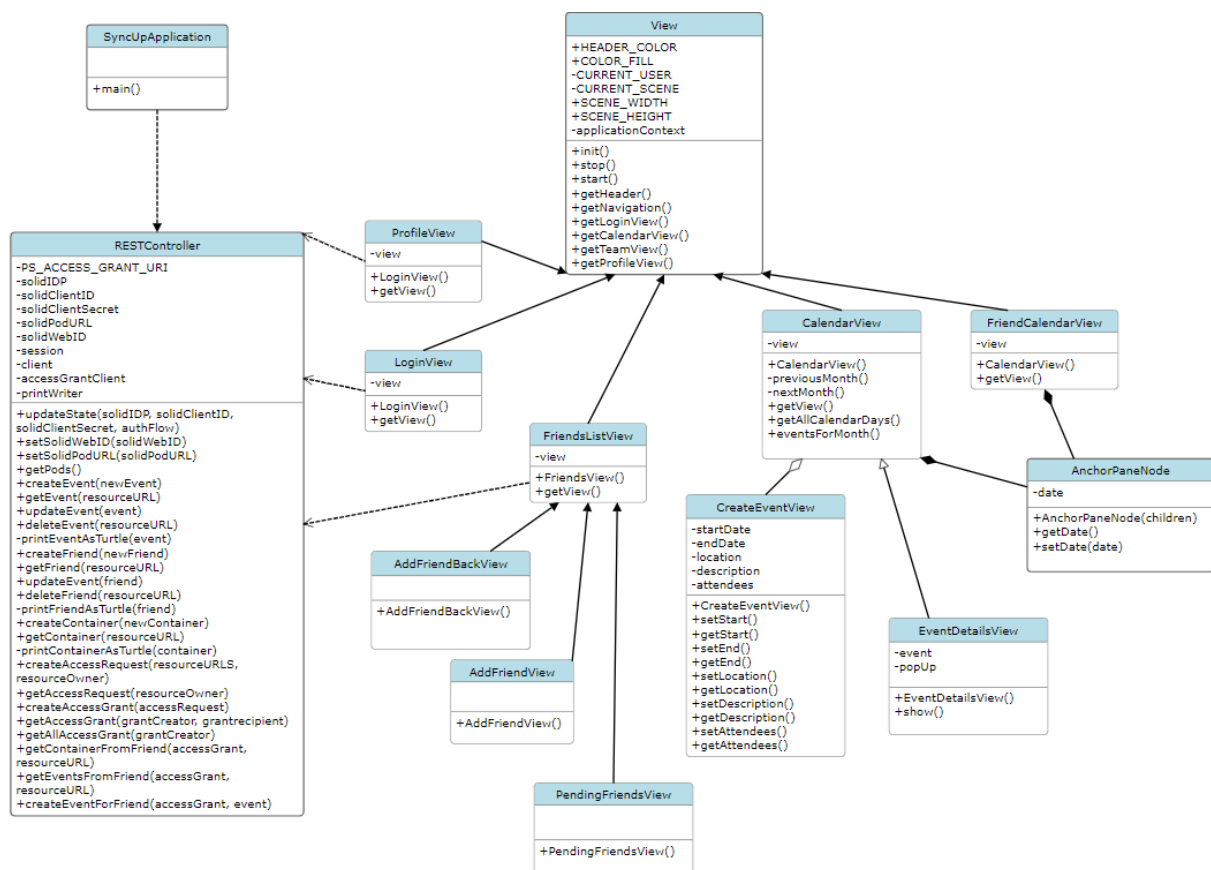


Figure 4.3 – SyncUp Class Diagram

4.4. Sequence/activity diagram

4.4.1. Log in authentication function

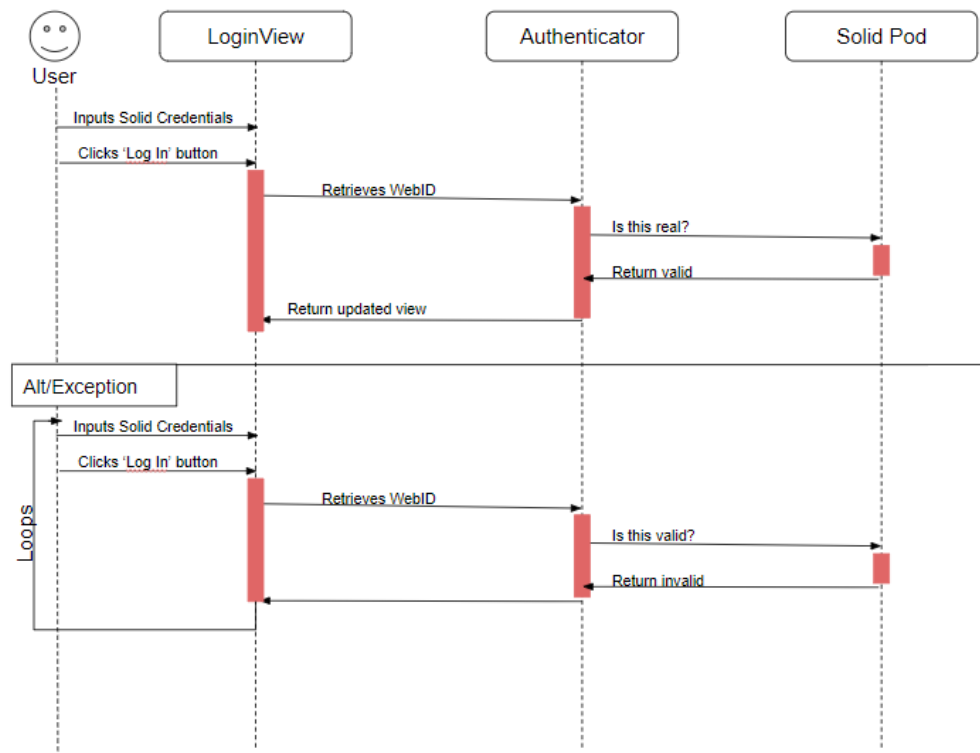


Figure 4.4.1 – Sequence Diagram of the Login function

4.4.2. Input Unavailability Function

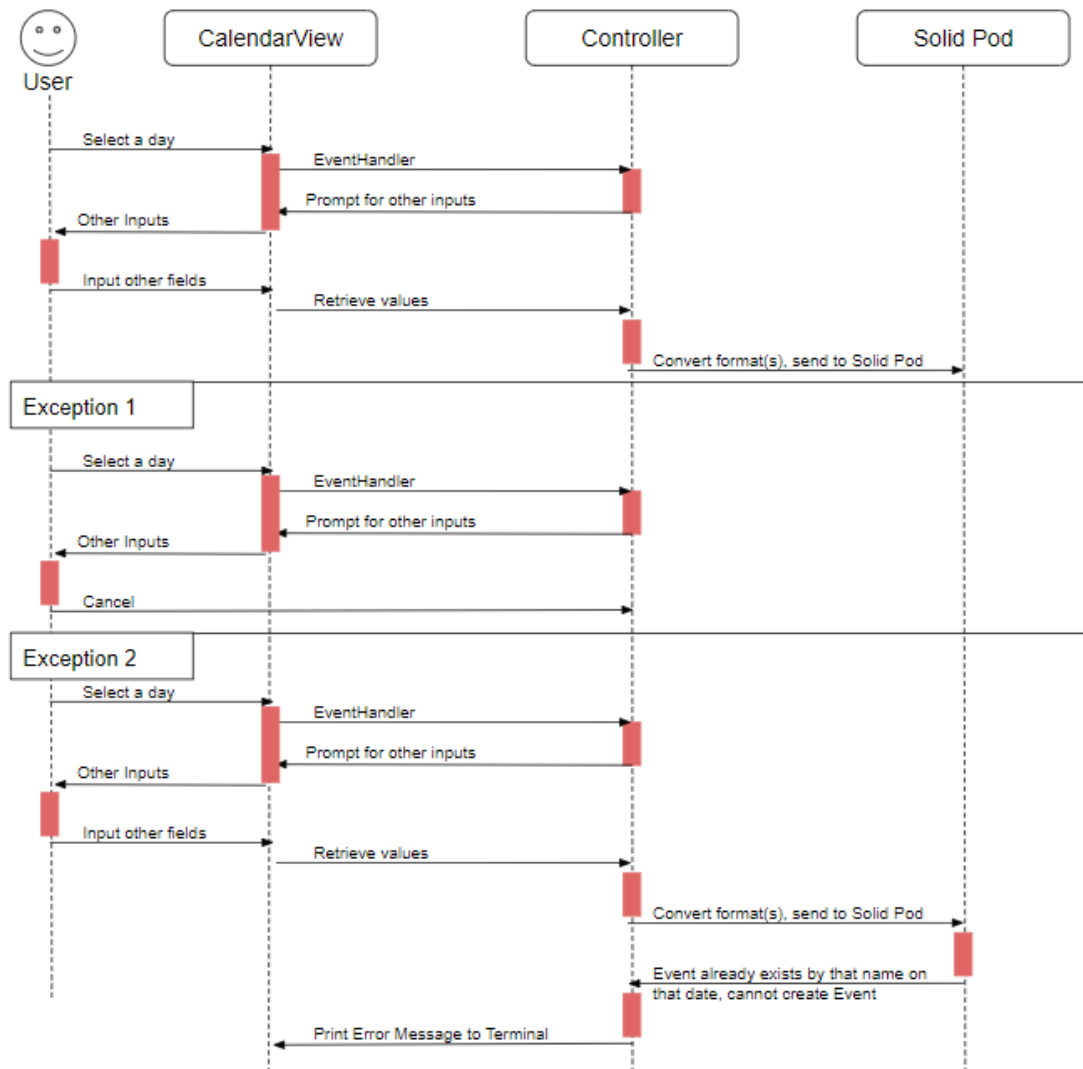


Figure 4.4.2 – Sequence Diagram for the Input Event Function

5. Others

Authentication: Inrupt's ESS' Solid OIDC (OpenID Connect) Broker Service will be used to handle authentication into the SyncUp application. Solid applications typically use a WebID for login, but this service will allow users to login with any OIDC-compliant identity provider as well.

6. Test plan

No.	Test case	User input	Pass criteria
1	Add new event to calendar	Event details: Name, Title, Date, Duration	Event is added to user's calendar, and is visible to user
2	Check someone's availability	Group/user's name, date	Availability for group/other user for given day is displayed to user

3	<i>Send a friend request</i>	<i>Friends name, username, or email</i>	<i>Friend request successfully sends, which is displayed to user. Receiver gets a notification</i>
4	<i>Accept a friend request</i>	<i>Selection of received friend request</i>	<i>Friendship has been established. User's can see each other's public data.</i>
5	<i>Remove a friend</i>	<i>Selection of a current friend to remove</i>	<i>Selected friend is removed from social network, and removes availability sharing</i>
6	<i>Join a group</i>	<i>Accepted invite, or requests to join a group where another user accepts them</i>	<i>User is added to group. Can view group details as well as group availability calendar.</i>
7	<i>View a friend's calendar</i>	<i>Selection of a friend's profile</i>	<i>Friend's public calendar and available slots of time are displayed to user</i>
8	<i>Update existing event</i>	<i>Updated event details: Title, date, time, details.</i>	<i>Same event with new details is displayed, event with old details is no longer visible</i>
9	<i>Delete existing event</i>	<i>Selected event to delete</i>	<i>Event will no longer be visible, and removed from Solid Pod.</i>

6.1 .Function #1: Adding Event to User Calendar

6.1.1.Event is added to pod to store in events.

6.1.2.Event is stored in path of current month. For March 2024, it will be stored in events/202403/event(x).

6.1.3. When the calendar for March 2024 is loaded, the event will be populated along with the calendar from the pod.

6.2 Function #4: Sending a Friend Request

6.2.1 Select the "Add" button on Friends page.

6.2.2 Input the WebID of the friend you want to connect with (as well as whatever name you wish to store them under).

6.2.3 Their information is stored in a container for outgoing friend requests.

6.2.4 An access request is sent to the requestee.

6.2.5 The friend request is displayed on the requestee's Friends page, under the Pending pop up, with the requester's WebID and options to Accept or Deny the request.

7 References

[1] Apple Inc. (n.d.). *iCloud User Guide*. Apple. Retrieved February 25, 2024, from <https://support.apple.com/guide/icloud/what-you-can-do-with-icloud-and-calendar-mm15eb200ab4/icloud>

[2] Calendar (2023, May 2). *Apple Calendar Guide: Everything You Need to Know About iCal*. Retrieved February 25, 2024, from <https://www.calendar.com/apple-calendar/>

[3] Google LLC (n.d.). *Google Calendar Help*. Google. Retrieved February 25, 2024, from <https://support.google.com/calendar/?hl=en#topic=10509740>

[4] Inrupt (n.d.). *Inrupt Java Client Libraries*. Retrieved March 21, 2024, from <https://docs.inrupt.com/developer-tools/java/client-libraries/>

[5] Lau, J. (2023, November 27). *9 Google Calendar features you should start using now*. Zapier. Retrieved February 25, 2024, from <https://zapier.com/blog/google-calendar-schedule/>

[6] The W3C Solid Community Group (n.d.). *Solid: Your data, your choice. Advancing Web standards to empower individuals and groups*. Solid. Retrieved March 21, 2024, from <https://solidproject.org/>

8 Appendix

8.1 Comparison to Existing Products

8.1.1 Apple Calendar

- Apple offers “Calendar” which is a calendar app built into the operating system of nearly all of their electronic products such as iPhone, iPad, and iMacs (Apple).
- This app offers a virtual calendar where users can create events on specific dates and times.
- Apple Calendar supports embedded GPS data for event location purposes, URL support for related material, and notifications feature to remind users about the event.
- Apple has iCloud, a cloud storage solution, that when integrated with their native [1] Calendar app, can allow a user to check their virtual calendar on any of their devices connected to the same iCloud account.
- Users can also share their apple Calendar with friends and family through iCloud.
- While this app is integrated seamlessly with Apple devices, users on Windows or Androids will have to use utilize workarounds to access Apple Calendar (Calendar).
- Shared calendars have permission levels of “View Only,” “Can Edit,” or “Can Edit & Share.”
- Users are not able to directly see friends and families’ calendars unless a specific calendar has been shared between all parties.
- Apple Calendar does not have a suggestion feature to propose potential plans for all invitees to review and accept/reject.
- Apple Calendar does not have a feature that recommends a date and time for friends and family to create new events that do not have conflicting timings.
- Apple Calendar has a public calendar feature but only the original creator of the calendar can make changes to its events and sharing permissions.
- Apple Calendar has many features geared towards personal event creation with added abilities to share for other people to view and edit. However, it lacks features for invitees to provide their input on their own availability.

8.1.2 Google Calendar

- Compared to other calendar applications, Google Calendar also offers a virtual calendar that gives users the ability to create events and add information such as date, time, and location (Google).

- Google Calendar is designed for work organizations rather than friends and family.
- Google Calendar users can subscribe to other user's calendars to view events on their calendars (Lau).
- Users can set days and times for when they are offline [2].
- Google Calendar has a search feature for users to search key words to find created events on their calendar.
- Google Calendar has a focus time feature that users can use to notify external users that a person cannot be bothered at a certain day and time.
- Google Calendar offers some analytics to notify users of what they are spending their time on, which is useful in business to identify how time is spent.
- Google Calendar has support to embed its calendars onto website.
- Like Apple Calendar, there is no permission level to provide suggestions only but there are options to allow people to read or edit only.
- While Apple Calendar is designed for family and friends, Google Calendar is designed to be used in a work or business setting.