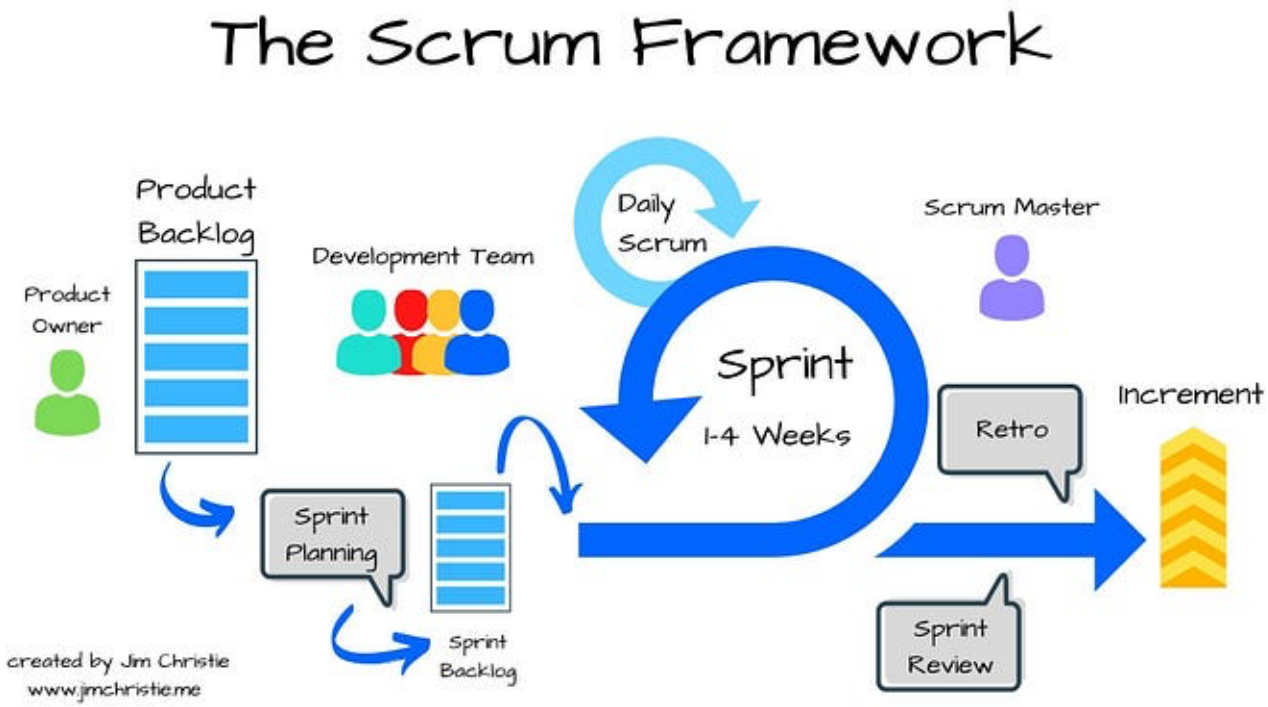
SOLOMA SYSTEM SPECIFICATION



(Christie, 2023)

Andres Gaspar Gonzalez
CSC3150 Systems Design
Andy Cameron
June 13, 2024

Table of Contents

1. Executive Summary

This document provides the system specifications of the Soloma system. The Soloma system is a digital platform based on the Scrum framework, emphasizing communication and collaboration. It aims to help project members focus more on the project rather than on the management of a project. The primary audience for this project is the developers of Soloma. They should be familiar with the SCRUM framework, understand its elements, and have experience developing web applications and data management.

The document provides developers with a high-level system structural model to base their fundamental programming. It also shows the expected relationships between classes and their metadata. This document also includes a high-level architectural design so the development team may understand what features they need to include to meet system requirements. As a minor part of this section, the document consists of a security plan to emphasize security in most, if few, system parts. Finally, this document provides developers with a high-fidelity wireframe and navigation diagram so they may see and understand the intended functionality of Soloma.

The systems customers are other development teams looking for a way to reduce project complexity and communicate better. How Soloma will guide these end-users through the software development life process will determine the system's success. Finally, the next steps for the system are to be cross-platform and SCRUM certified so that users can be confident in delivering efficient and complex developed products.

2. Introduction

The Soloma system is a project management web application based on the SCRUM framework. It provides users with SCRUM-like features that serve as guidance tools throughout a project. It also updates a project's progress and keeps track of project elements, such as the product backlog, daily scrums, sprints, and reviews.

Soloma aims to be a collaborative platform where project teams may communicate with each other and contribute to the project. Thus, by adding a teammate to a project, project participants receive access to a project and can change or add details. Solomas' task is to keep track of such changes and report them to a dashboard screen where the most critical changes, progress, and data are displayed.

2.1. Problem Statement / Project Vision

Many software development teams follow a particular framework to guide them during development. Yet, many often encounter communication and dependency problems. Such problems cause teams to focus less on a project and more on these other matters. Focusing on different issues causes a decrease in efficiency and a reduction in focus on a project. Here, the need for a service that handles such matters appears so that teams can focus on the project rather than the surrounding elements of developing software.

Soloma aims to deliver such services by providing teams with a digitalized platform based on SCRUM. Using Soloma, teams expect to be more efficient and less prone to miscommunication problems. The potential stakeholders for this system are the development team, the UX team, security, Finance, project owners, business owners, and Scrum masters. Each of these stakeholders has some influence and interest in this system. They would all benefit from this system because of better communication, project layout, and contribution.

2.2. System Capabilities

Below is the section that provides Soloma functional requirements by name and ID number. For more information, see sections 4.2 and 5.3 of the system proposal.

1. Process Log-in (UC-2): Users should be able to log in using their credentials.
2. Create vision (UC-4): Users should be able to create a vision (project) folder.
3. Build sprint (UC-5): Users should be able to create a sprint folder.
4. Maintain Project (UC-7): Users should be able to maintain a project following CRUD.
5. Process SCRUM review (UC - 8): Users should be able to perform a SCRUM review at the end of every sprint.
6. Deploy Project (UC–9): Users should be able to decide whether a system is deployable.

2.3.Non-functional Requirements and Design Constraints

Soloma aims to help teams communicate and contribute better to a project. This section briefly overviews the systems' non-functional requirements and scope.

Non-functional requirements:

1. Operational requirements
    1.1. Developers shall begin by building Soloma as a web application. Other versions capable of running on different platforms will be released once 1st version is successful.
    1.2. The user interface shall be user-friendly and be expected to receive excellent reviews.
2. Performance requirements
    2.1. The system graphics must show information that meets current market standards. Runtime.
    2.2. The system shall aim to have a minimal downtime of 90%.
3. Security requirements
    3.1. The system shall have the user sign-in and 2-step verification.
    3.2. Passwords shall renew every 90 days.
4. Cultural requirements 4.1. None.

Constraints:

1. Developers must build Soloma using SCRUM.
2. It will not be an IDE.

2.4.System Evolution

In version one (MVP), Soloma can deliver the essential SCRUM elements to users, such as creating a vision, sprints, scrums, and reviews. These elements are more detailed in section 5.3 of the systems proposal or section 5 of this specification.

2.4.1. Version 2 Changes

Version two will aim to make collaboration and communication better by introducing video calls and now tracking. Below are more in-depth descriptions.

1. Video calls: Soloma will allow users to make video calls in teams or to a single person.
2. Now tracking: When a user is inside a project, the system shows other contributors where a teammate is and what they are doing by hovering an icon over them like a mouse.

2.4.2. Version 3 and beyond Changes

2.5.Document Outline

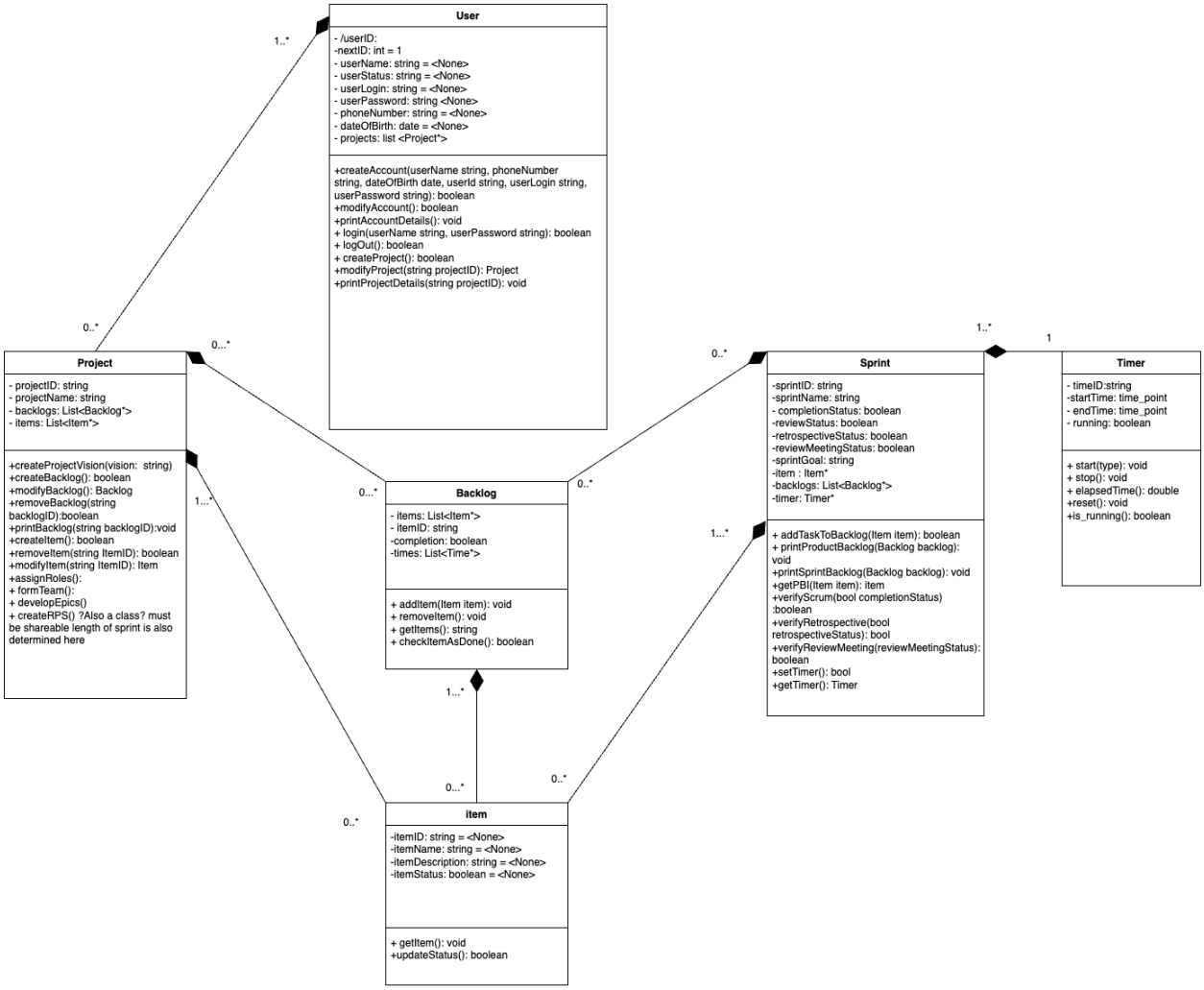
This system specification document provides a layout for the Soloma system. Section 3 provides a structural model for the system. This model contains class diagrams and descriptions. Section 4 includes the architecture design for the system, which provides a high-level overview of the architecture of the Soloma system. Finally, section 5 contains the user-interface elements of Project Soloma. This section analyzes Solomas's functionality and navigation.

3. Structural Model

3.1. Model Introduction

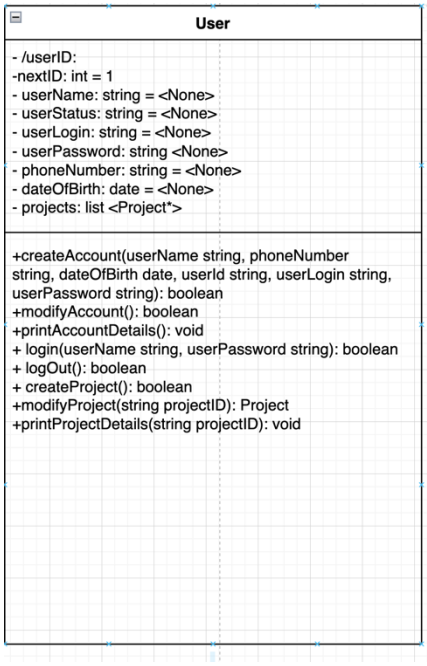
This section reviews a structural model containing a class diagram and the metadata of each class's elements. The class diagram is based on the Unified Modeling Language (UML) and aims to describe (on a high level) the system's structure. Descriptions of class attributes and operations, which aim to provide further detail on each class, comprise the metadata.

3.2. Class Diagrams



3.3.Metadata

User



Description: Represents a user profile.
Visibility: Public
Is Abstract: No

Additional Information:

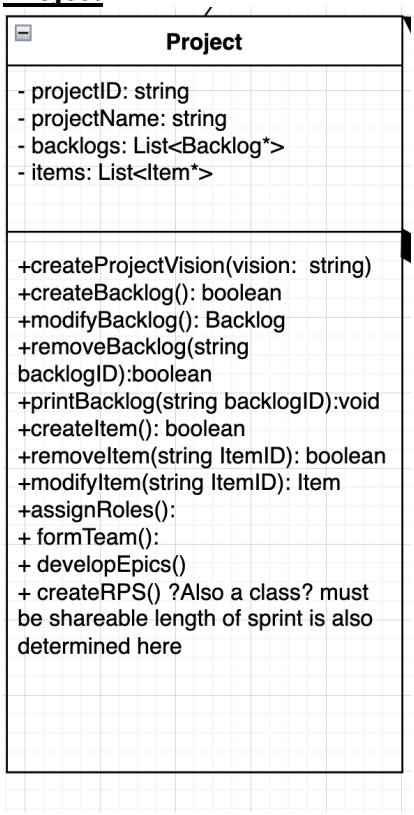
Attributes

Name	Description	Read Only?	Multiplicity
firstName	The first name of the user.	No.	1
lastName	The last name of the user	No.	1
dateOfBirth	Date of birth of user	No.	1
phoneNumber	The Phone number of the user.	No.	0..1
emailAddress	The email address of the user.	No.	1
userID	The unique identifier for each user.	No.	1
userStatus	Represents the company status of the user.	No.	0..1
userLogin	The unique username to user.	No.	1
userPassword	The password of user account.	No.	1
projects	Projects that the user has.	No.	0..*

Operations

Name	Description	Is Query?	Is Polymorphic?
createAccount	Creates a user account with a given first and last name, date of birth, email address, phone number (optional), username, and password. Then, it assigns the user a unique identifier.		No.
modifyAccount	Allows the user to make changes to any attributes passed through as parameters.		No.
printAccountDetails	Prints the account details		No.
login	Logs the user in by accepting a username and a password. Then, it verifies that such elements exist and are part of a user account.		No.
logout	Logs the user out.		No.
createProject	Creates a project object for the user from the Project class.		No.
modifyProject	Lets the user modify aspects of project object.		No.
printProjectDetails	Prints project details.		No.

Project



Description: Represents a project.
Visibility: Public.
Is Abstract: No.

Additional Information:

Attributes

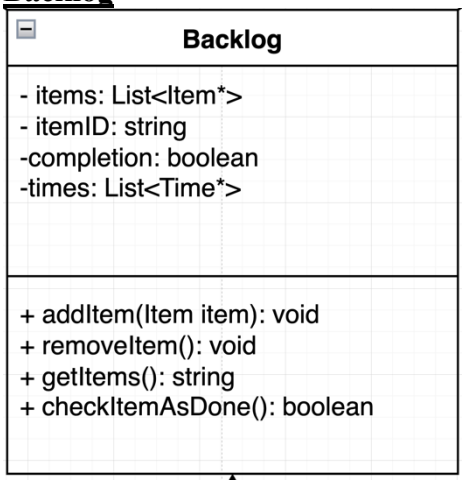
Name	Description	Read Only	Multiplicity
projectID	The unique identifier to a project.	Yes.	1
projectName	The unique name of a project. Also serves as a description of what the project is.	No.	1
backlogs	List of backlogs that a user has created.	No	0..*

Operations

Name	Description	Is Query?	Is Polymorphic?
createProjectVision	The project description	No	No.
createBacklog	Creates a new backlog where project items can be stored.	No.	No.
modifyBacklog	Modifies a certain backlog.	No.	No.
removeBacklog	Removes a certain backlog.	No.	No.

printBacklog	Prints a certain backlogs' details.	Yes.	No.

Backlog



Description: Represents a backlog.
Visibility: Public.
Is Abstract: No

Additional Information:

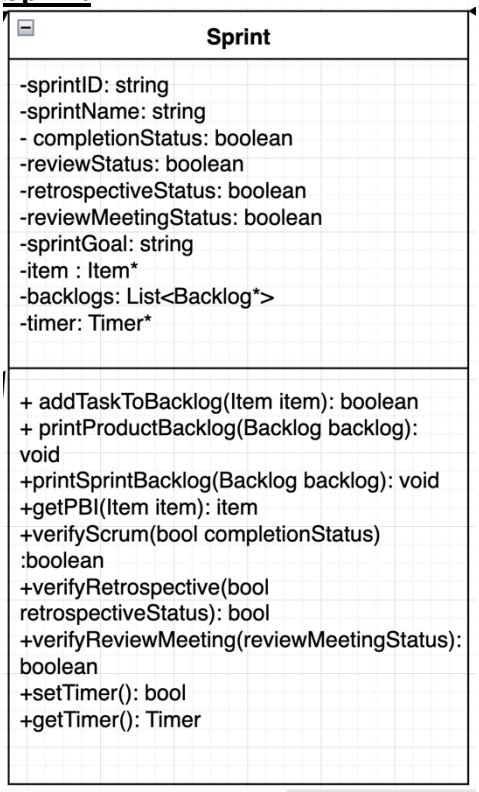
Attributes

Name	Description	Read Only	Multiplicity
Items	An instance of an item.	No.	0..*
itemID	The unique identifier to an item.	Yes.	1
completion	Status of an item. Completed or not completed.	No.	0..1
times	An instance of a timer.	No.	0..1

Operations

Name	Description	Is Query?	Is Polymorphic?
addItem	Adds an item to the backlog.	No.	No.
removeItem	Removes an item from the backlog	No.	No.
getItems	Gets an item from the backlog.	No.	No.
checkItemAsDone	Checks an item as complete.	No.	No.

Sprint



Description: Represents a Sprint.
Visibility: Public
Abstract: No

Additional Information:

Attributes

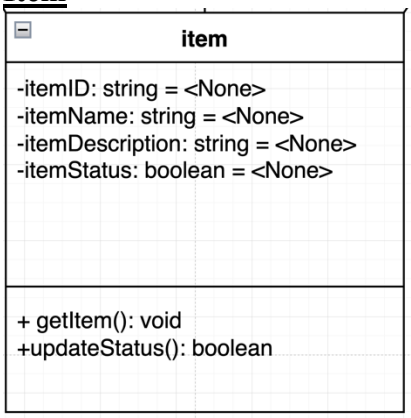
Name	Description	Read Only	Multiplicity
sprintID	Unique sprint identifier.	Yes.	1
sprintName	Unique sprint name.	No.	1
completionStatus	Sprint completion status.	No.	0..1
reviewStatus	The status of a sprint review. Whether one has been performed or not.	No.	0..1
retrospectiveStatus	The status of a sprint retrospective. Whether one has been performed or not.	No.	0..1
sprintGoal	The goal of the sprint.	No.	1..*
item	An instance of an item.	No.	0..*
backlog	An instance of a backlog.	No.	0..*

timer	An instance of a timer.	No.	0..1

Operations

Name	Description	Is Query?	Is Polymorphic?
addTaskToBacklog	Adds an item to a backlog.		
printProductBacklog	Prints the data in the requested backlog.		
printSprintBacklog	Prints the data in the requested backlog.		
getPBI	Prints an item from the requested backlog.		
verifyScrum	Verifies the status of the daily scrum.		
verifyRetrospective	Verifies the status of a retrospective.		
verifyReviewMeeting	Verifies the status of a review meeting.		
setTimer	Sets a timer on the sprint.		
getTimer	Gets the information on a timer.		

Item



Description: Represents an item.
Visibility: Public
Abstract: No

Additional Information:

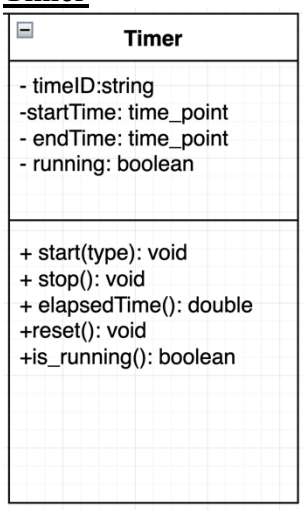
Attributes

Name	Description	Read Only	Multiplicity
itemID	The unique identifier of an item.	Yes.	1
itemName	The name of an item	No.	0..1
itemDescribtion	The description of an item.	No.	0..1
itemStatus	Completion status of an item.	No.	0..1

Operations

Name	Description	Is Queue?	Is Polymorphic?
getSummary	Retrieves a summary of the item ID, name, description, and status.		
updateStatus	Updates status of item. Either complete or incomplete.		

Timer



Description: Represents a timer.
Visibility: Public
Abstract: No

Additional Information

Attributes

Name	Description	Read Only	Multiplicity
timeID	The unique identifier of a time.	Yes.	1
timeName	The name of a timer.	No.	0..1
startTime	The start of a timer.	No.	0..1
endTime	The end of a timer.	No.	0..1
running	Verifies status of timer.	Yes	0..1

Operations

Name	Description	Is Queue?	Is Polymorphic?
start	Starts a timer.		
stop	Stops a timer		
elapsedTime	Retrieves how long has it been since timer has been started.		
reset	Resets a timer.		
is_running	Retrieves status whether timer is running or not.		

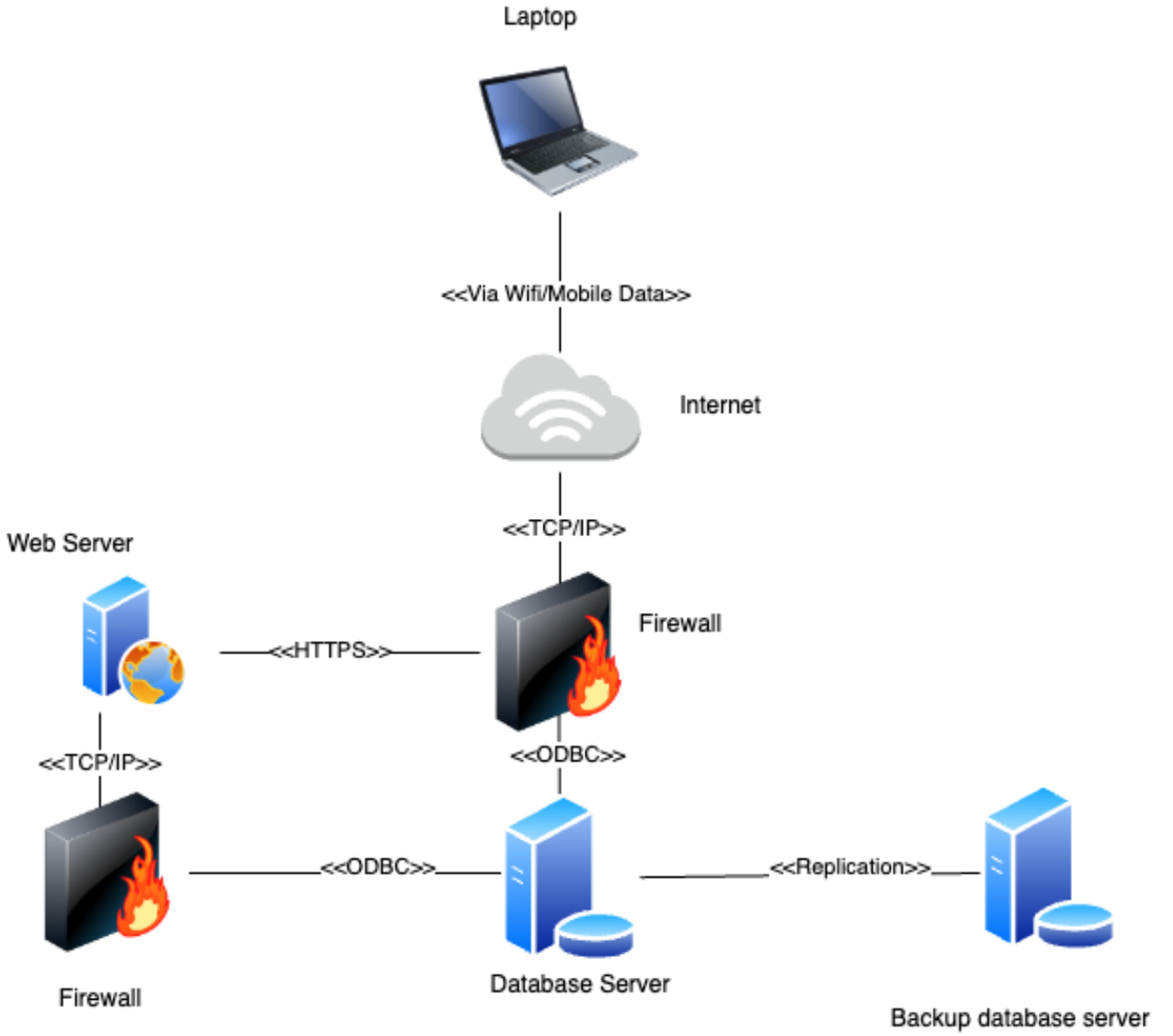
4. Architecture Design

4.1. Architecture Overview

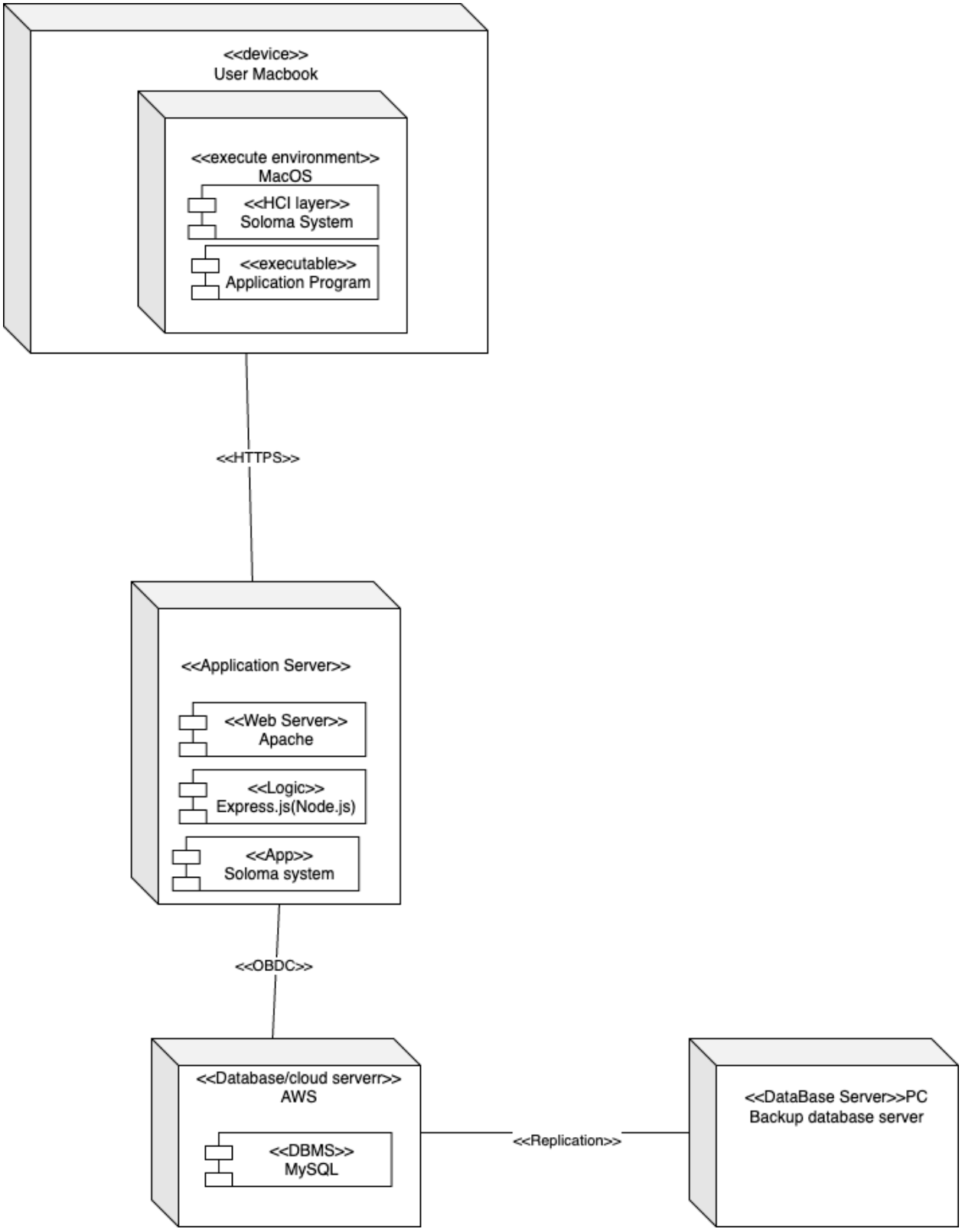
This section analyses Solomas' client-server architecture. Beginning with two deployment diagrams, the developers will see how to build Soloma. The first diagram shows a high-fidelity overview of the architecture Soloma expects to use. The second goes in-depth by showing what software the architecture will use. No purchase of physical servers is necessary for Soloma, as data management will be in the cloud. The subscription to AWS EC2 will handle the data management instead. The hardware and software requirements section gives the elements Soloma needs to succeed. The security plan section, even though small, will provide developers with the proper path in developing Soloma so that project data is secure for future users.

4.2.Infrastructure Model

4.2.1. Deployment Diagram 1 – Architecture Overview



4.2.2. Deployment Diagram 2 – Nodes and Artifacts



4.3. Hardware and Software Requirements

4.3.1. Hardware Components

1. Web Server:

Soloma will depend on the Apache HTTP server to receive HTTP requests from the client. This server was chosen because it allows for static content without additional processing.

2. Database/cloud server:

Soloma will use MySQL as the database management system. The simplicity of tables and schema-based structure is ideal for the Soloma system.

3. Backup Database server:

Soloma will use Amazon Aurora, which is MySQL-compatible, for the backup database server. It allows for continuous backup to Amazon EC2.

4.3.2. Required Software Components

1. Visual Studio Code (VS Code) is a source code editor that will be needed to implement this system. It is easy to install and allows for many extensions of modules such as express.js and node.js.
2. Git will also be required as the Soloma system must have a version control. It allows for distributed version control, such as offline work, local branching, and parallel development. It is also an extension of VS Code, which makes things easier.
3. Draw.io will be used as a design tool for class, deployment, and use case diagrams.
4. Balsamiq will be used as a design tool for wireframes as it has good high-fidelity templates and icons. It also has linkable pages so that some navigation can be shown.

4.4.Security Plan

The Soloma system is a project management system that aims to be a collaboration platform. Additionally, one of the main aspects of this system is to store project data. Therefore, data must be secured, and the privacy of a project must be kept private to authorized users only. Below are the potential risks that need to be considered.

4.4.1. Security Overview

- Data Breaches: If the system data is breached, this could cause legal, monetary, and regulatory damage to the application and the users' project.
- Unauthorized Access: This can happen through stolen or guessed credentials. The potential risks of this are project deletion or project data manipulation that aims to damage a user's or project's reputation.
- Date Leaks: Projects stored and developed in this app are meant to be private to the company or user since they may bring monetary value. Thus, data leaks could cause identity theft or project idea theft.

4.4.2. Security Plan

Threats / Components	Data Breaches	Unauthorized Access	Data Leaks	
User Laptop	2,3	1	1,2,3	
Internet connection				
Application Server				
Database server	2,3			
Backup Database Server				

Controls

1. Robust 2-step authorization password software.
2. Strong firewall software
3. Virus checking software

5. User-Interface

5.1. User-Interface Requirements and Constraints

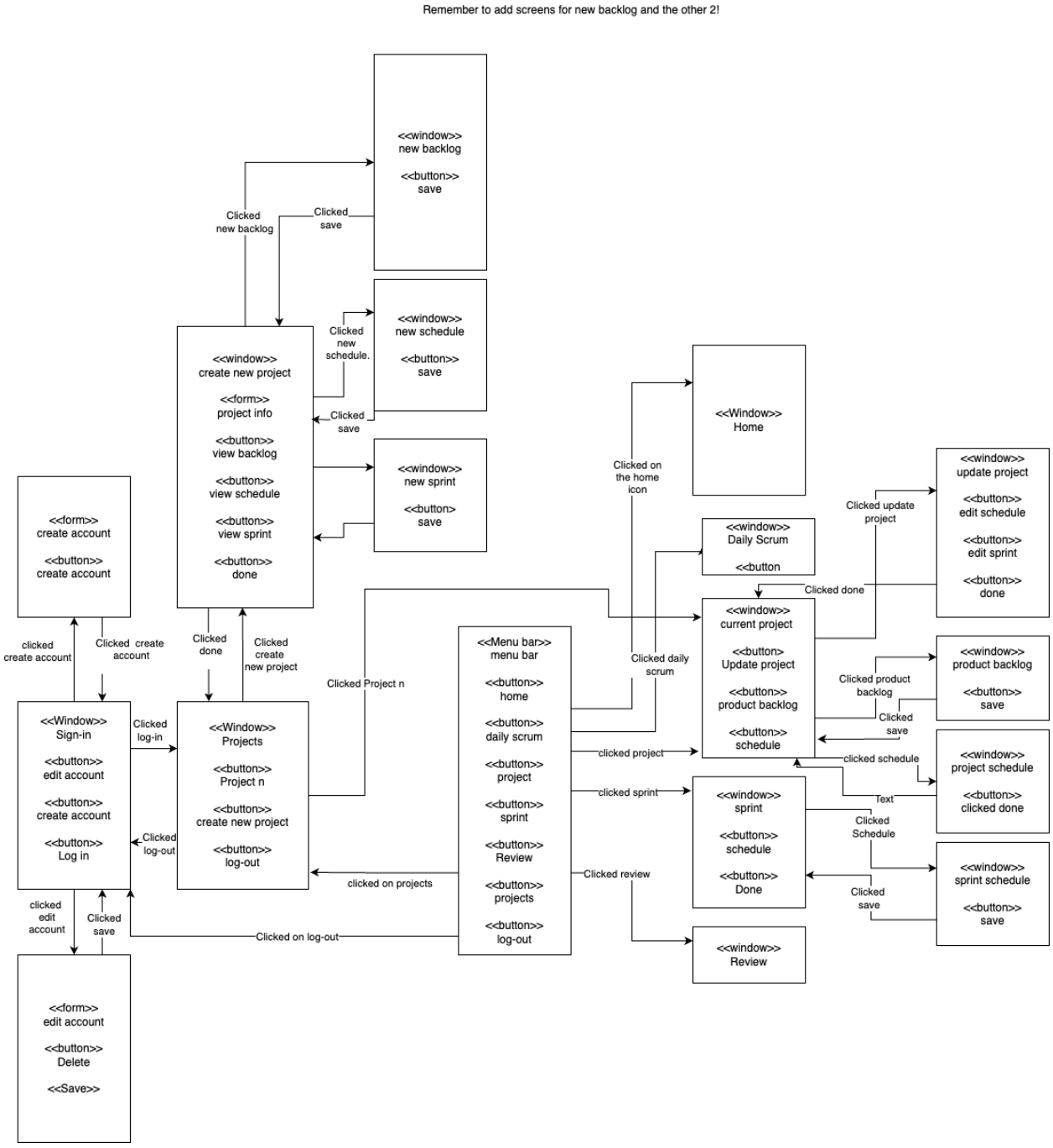
Soloma aims to be a platform that digitalizes the SCRUM framework. Therefore, the user interface must contain the elements of SCRUM and be able to output the progress of a project. Soloma also must be easy to navigate and secure. In other words, it must be easy to create projects, sprints, and schedules but it also must be secure when deleting such elements. This is because we do not want a user to lose all their work over a typo or accidental deletion.

Some constraints of the Soloma system are:

- If a user selects to delete an item, they must enter their password to do so.
- The only screens that are allowed to show the menu bar are the screens after a user has successfully logged in.
- The only screens that are GIVEN access to the menu bar are the screens after a user has selected an existing project. This is because of the way Soloma is set up. A user must first select a project or create one before being able to see the home screen. It is like choosing a directory or folder. If no folder exists then there is nothing there, if one does then they may only see the items of that folder after opening it. Furthermore, for the screens that are logged in but not yet inside a project they are only allowed the log-out or projects buttons. Once a user is inside a project there are no restrictions for navigation.
- Many of these screens are similar in layout and name but they are different in terms of the constraints of the above. That is, for screens that are about creating a new element of a project they must be different from those of editing since the editing screens are inside of a project and thus give the user more access.
- The Soloma system should also prioritize getting out the information in the best way possible. Some requirements that will support this is that all screens have the ability of zooming in or out, have read-aloud, and color contrast.
- The system should also be responsive and not lag as well as be well laid out in terms of most preferred icon placement. It would not seem efficient to place a button at the bottom left corner where users tend to look less rather the bottom middle of the screen.

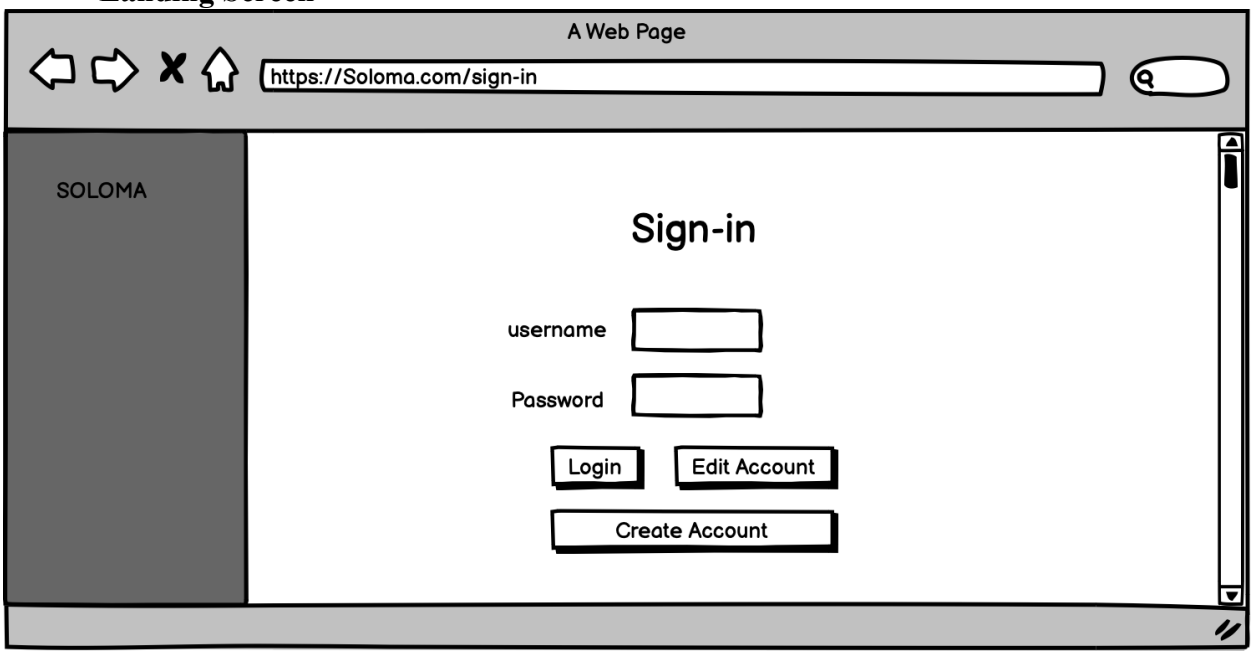
5.2. Window/Screen Navigation Diagram

The following diagram is a navigation diagram that aims to show how the Soloma system should navigate when a user selects an option to go to another screen.



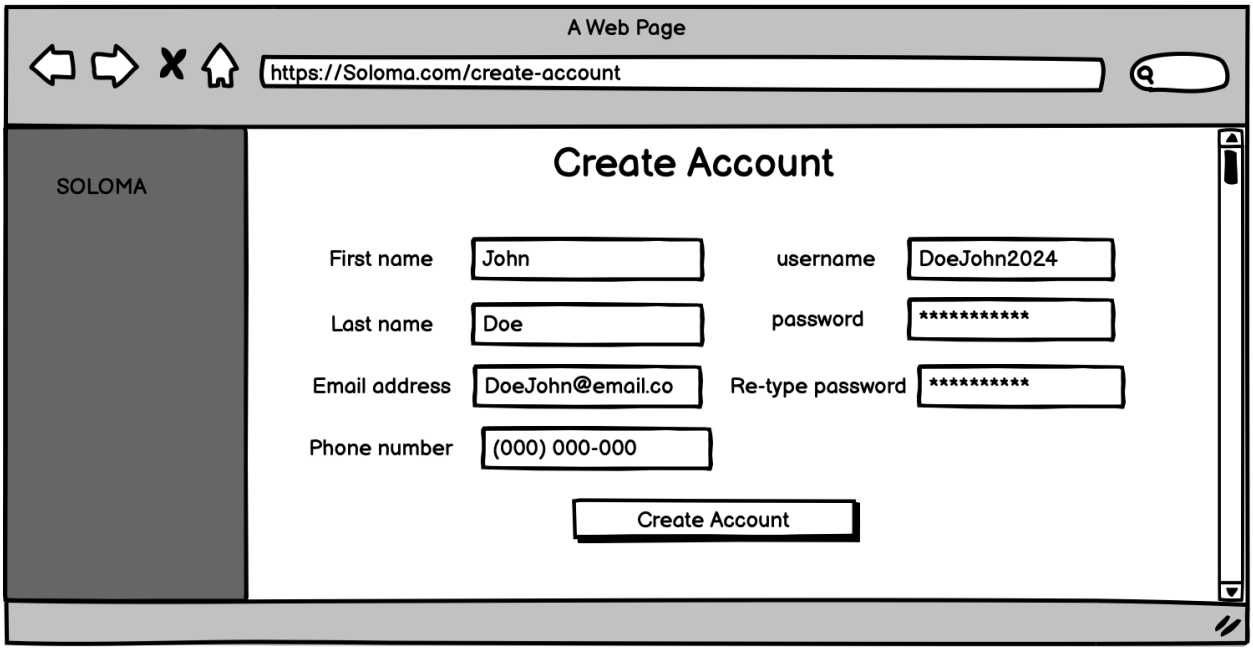
5.3. UI Wireframes

Landing Screen



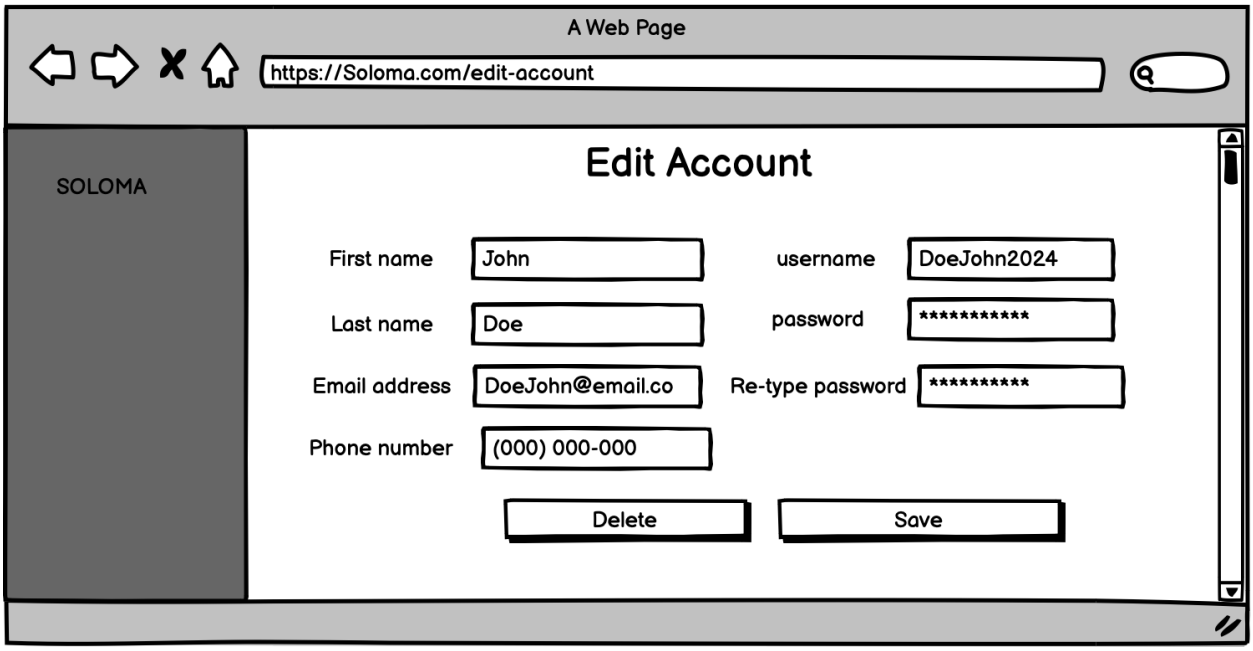
This screen is intended to be a sign-in screen that gives users three options. If they already have an account, they can enter their credentials in username and password. If they do not have an account, then they can create one by pressing “create account” which would navigate the user to another page. If the user would like to edit their account, then the user could select “Edit Account” which would navigate the user to another page.

Create account screen



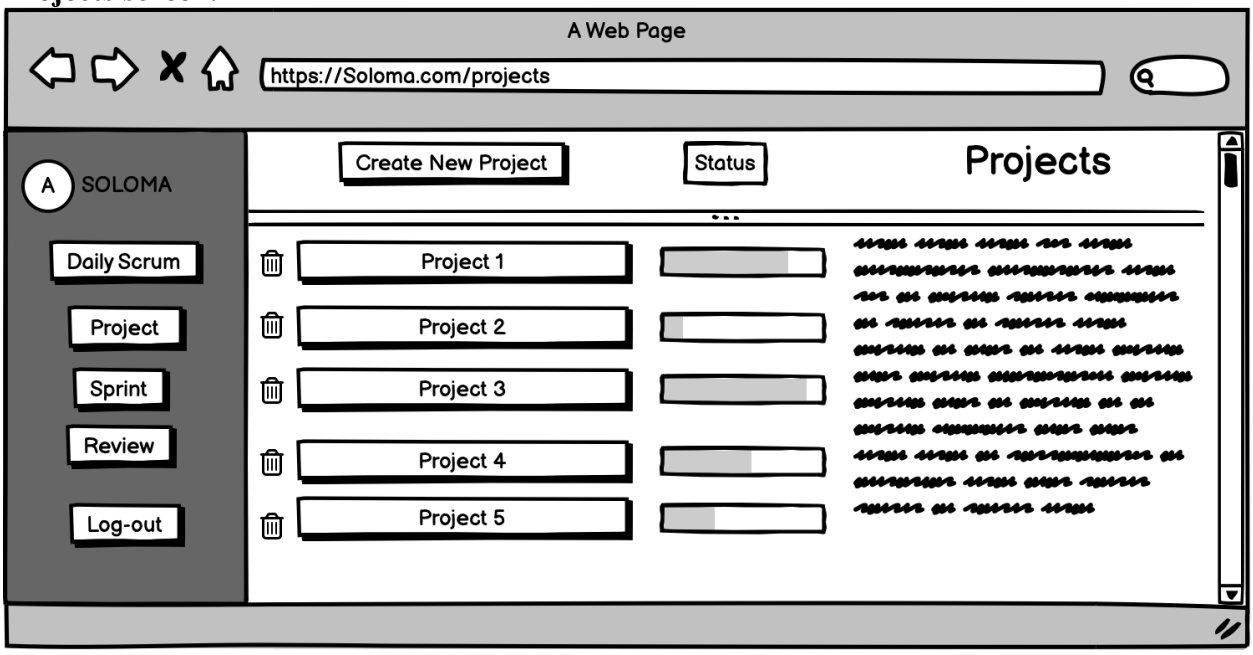
Create Account form asks the user for personal information as well as a password to go along with their account. Once the user is done filling in all the required information then the user can select “create account”. The user will then be sent back to the sign in page where they can use their newly created credentials and log in.

Edit account screen



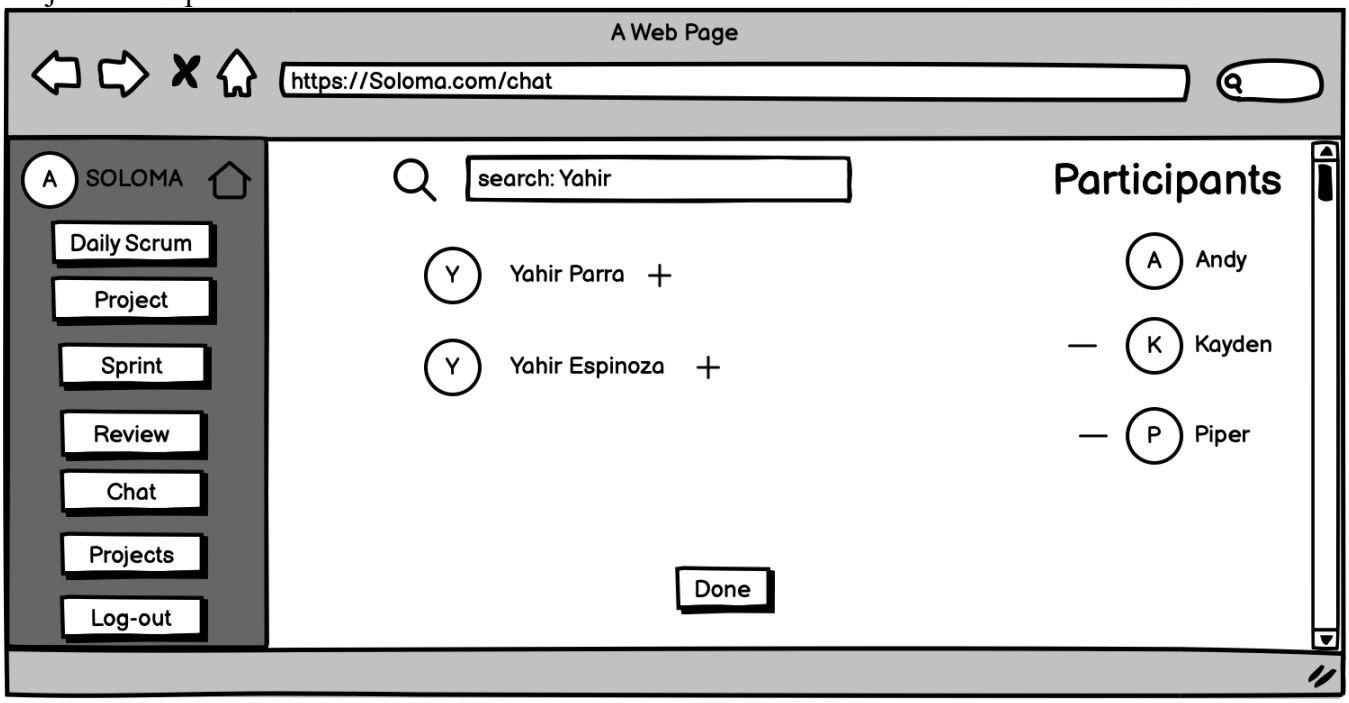
This form comes from one of the options in the sign in screen. Edit account allows a user to edit their account and confirms the account by checking if the entered password is correct. The user also has the option to delete the account if they wish to do so. The “save” button saves the users updated information and then guides back to the sign in page where they can log in.//have a button to change password or username or a login before that

Projects screen.



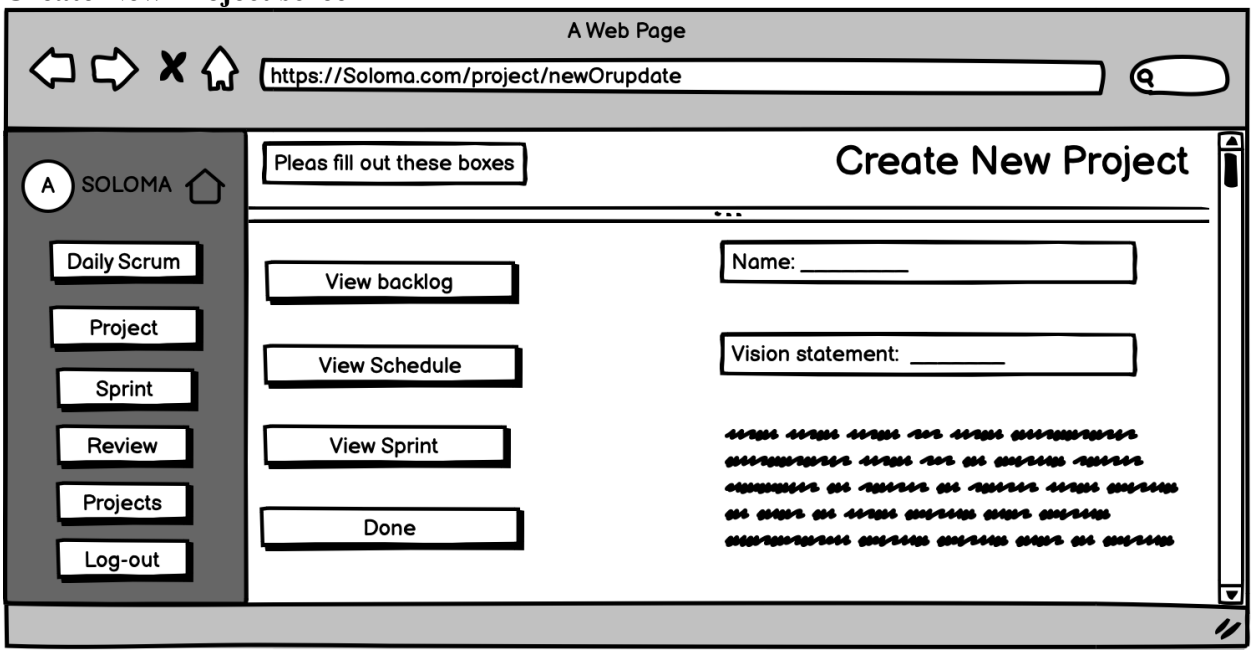
The projects screen is the first screen that a user will see after successfully logging in. All menu buttons except log-out are excluded and do not work. This is because a user must select a project from their previous work or create a new one. “Create New Project” navigates the user to another page where they can create a new project. If the user already has a project, they are working on they can simply select that project and they will be guided to the project’s dashboard. To the right of the projects are status bars that show the level of completion on the project. To the left there are mini trash can icons to symbolize the option to delete a project. If the icon is selected there must be a warning pop-up with the option to cancel or “Yes I am sure” to delete the project which would then require the users account password. To the right of the bars is a description of notes that a user has left for a certain project. If there are no projects the notes section is empty.

Project Participants



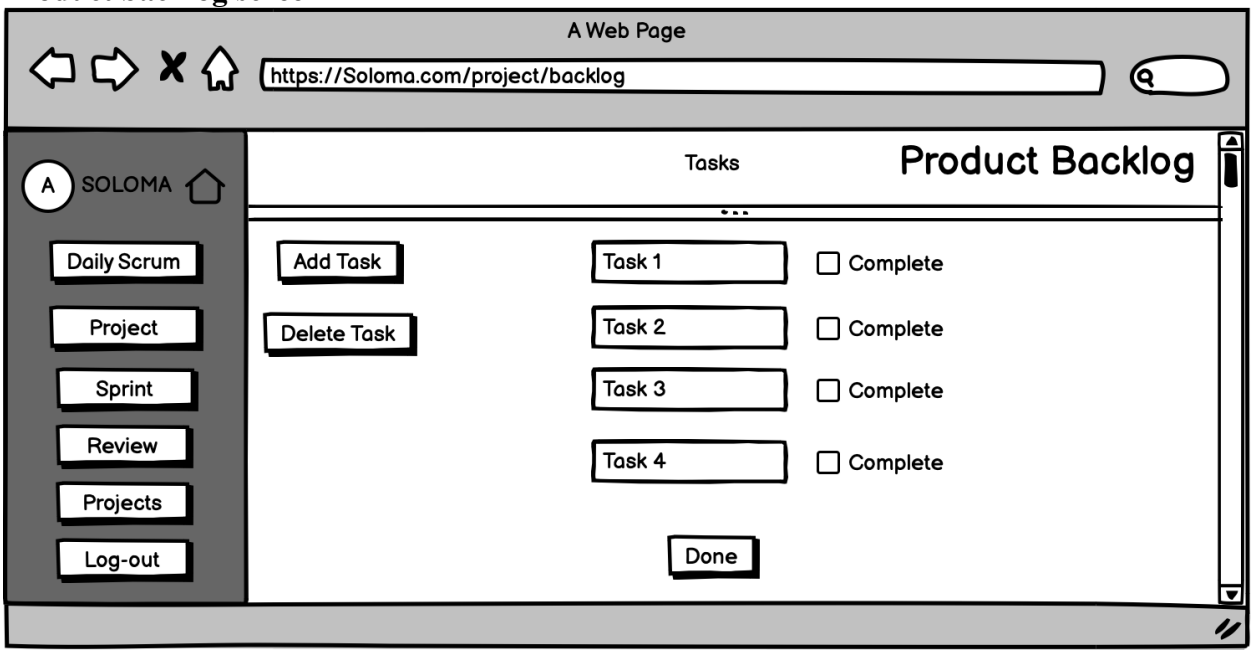
This screen comes from the project screen. Once inside a project the user may choose to add particpants/team members to a project. The user has the option to add or remove a participant by selecrting the minus or addition symbol. By adding a participant the user grants permission to a team member to access and change any part of the project. Owner status always remains with who created the project. Soloma tracks every particpants contribution.

Create New Project screen



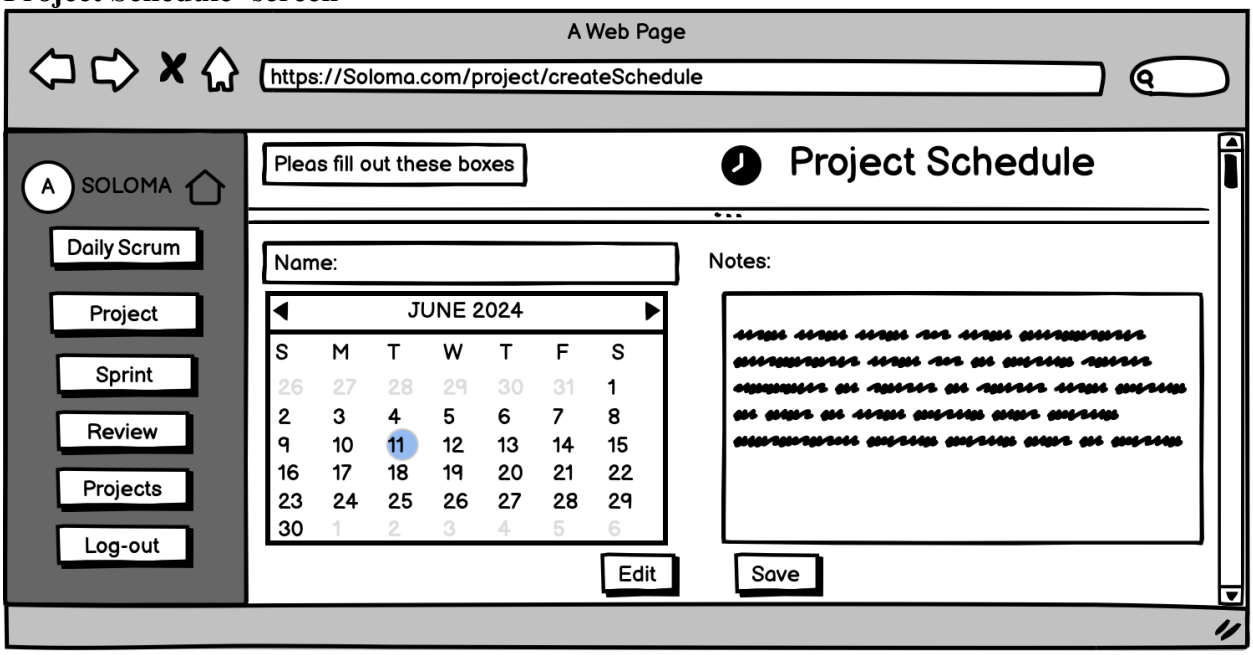
This screen is the screen following the users' selection of the "Create New Project" button in the projects screen. Here the user is given four buttons to choose from. "Create backlog" allows the user to create a backlog for the project. "Create Schedule" allows the user to set a schedule for the project. "Create Sprint" allows the user to create a sprint for the project" Once the user has inputted a name and vision statement for the project the user can click done to be taken to the projects section where they can select their new project. The name and vision statement are the only requirements for creating a new project. All the other elements may be added later. Since this screen is part of creating a new project, the menu bar is also limited to only the log-out button.

Product backlog screen



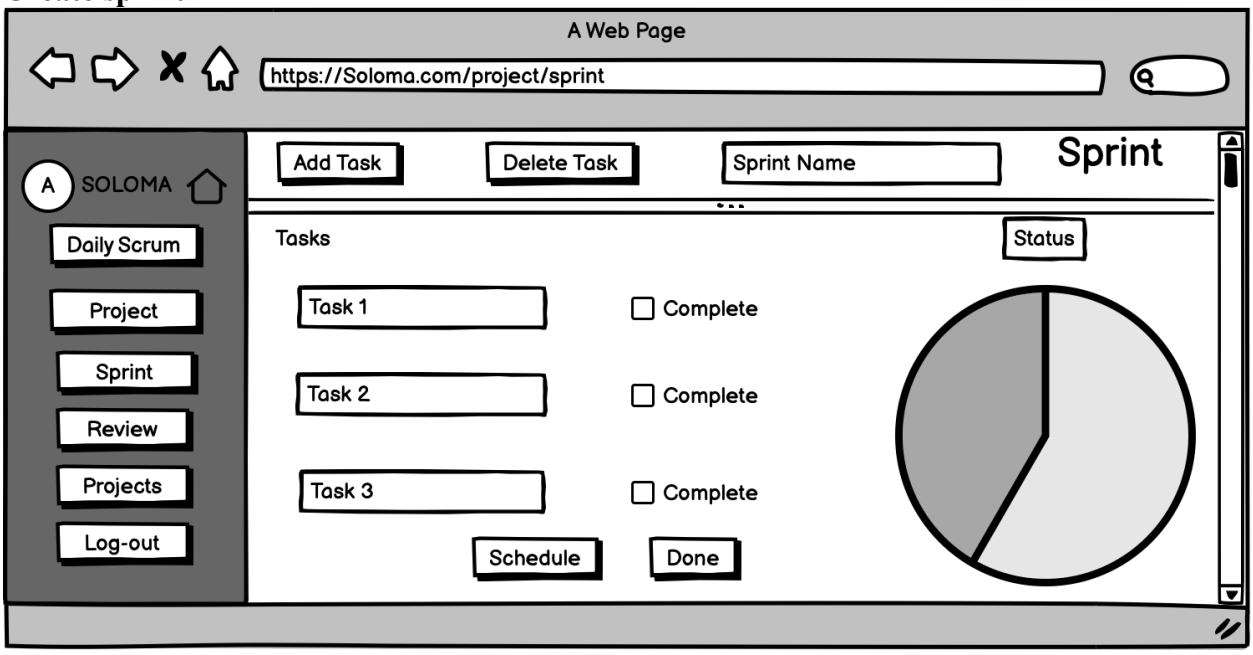
This screen comes after selecting “view backlog” from the options of the previous screen “create new project”. This screen has buttons that add, delete, or mark a task as complete. Once the user is satisfied, they may press “done” to go back to the previous screen. If the user decides to add a task to the backlog a new bar will appear to the right under tasks and a task name can be inputted. If the user decides to delete a task, they may press the bin icon and the same process as deleting a project occurs. In contrast, marking a task as complete does not delete the task. The user must implicitly tell the system to delete the task by following the delete task process. Since this screen is part of creating a new project, the menu bar is also limited to only the log-out button.

Project Schedule- screen



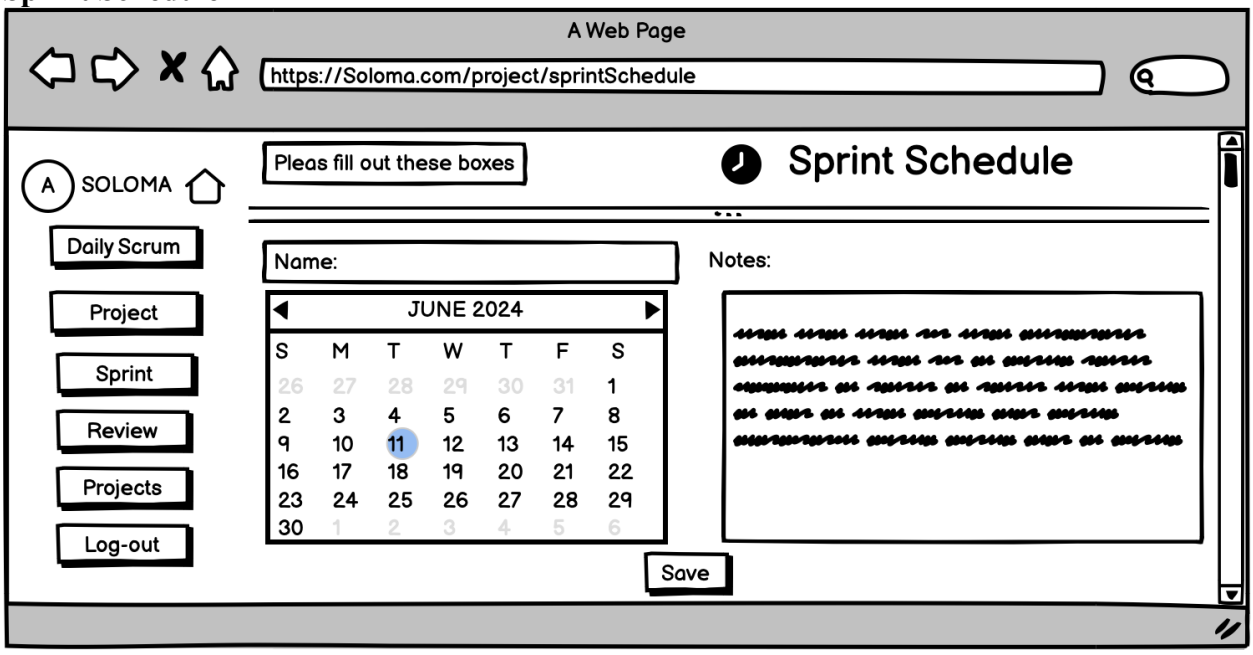
This screen comes after selecting the “view schedule” option from the “create project” screen. It allows the user to select a time and date for when the project is due. Name and notes sections are provided to give the schedule more of a description. Once the user is satisfied with their selection they may press save and then they will be navigated to the previous page. Since this screen is part of creating a new project, the menu bar is also limited to only the log-out button.

Create sprint



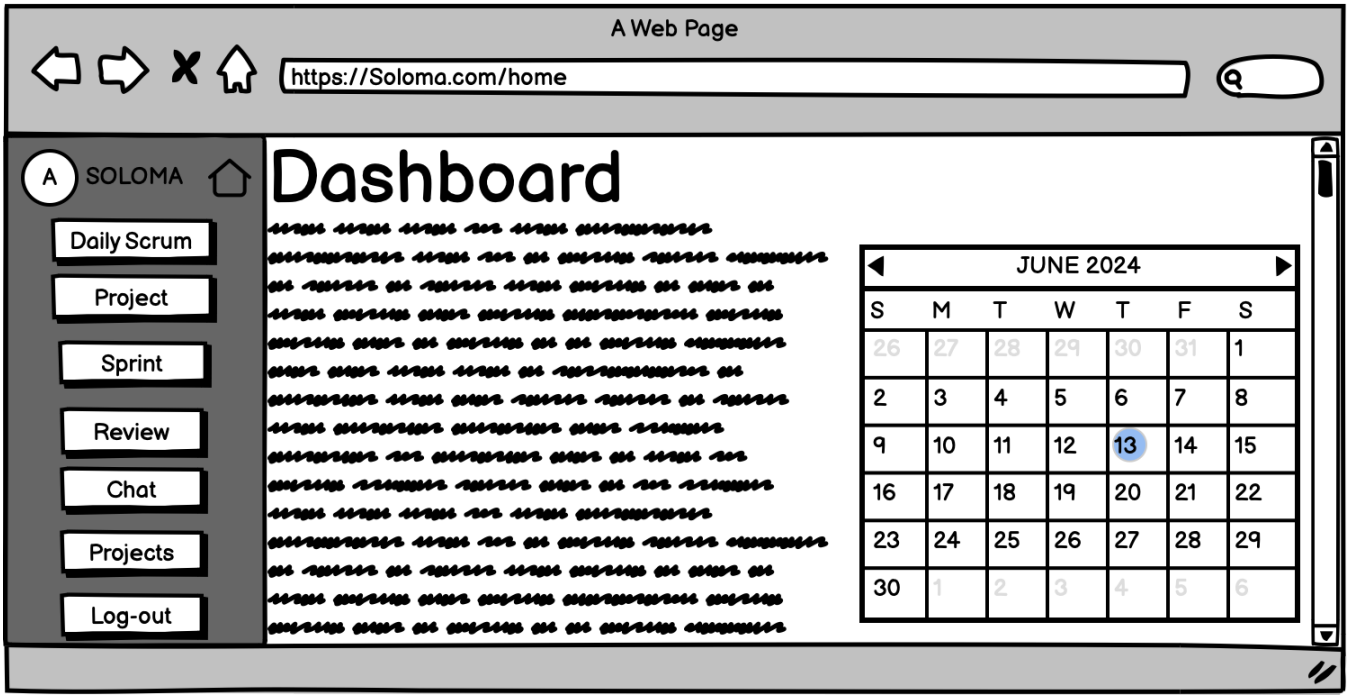
Create Sprint comes from selecting “create sprint” in the “create project” screen. There are six options in this screen. Add task allows the user to add a task to the sprint. Selecting the schedule button takes the user to another screen where they may set a schedule for the sprint. At the top right is a section for the sprint name. To the right of the tasks the option to mark a task a complete is given via checkboxes. The bin icons are like the previous screens. If a user selects the bin, then they are choosing to delete a task and are taken through the deletion process. There is also a sprint pie chart that marks the progress/percentage of work done based on the number of tasks and schedule. In this case, the sprint is new so there would be no status yet to be shown. Since this screen is part of creating a new project, the menu bar is also limited to only the log-out button.

Sprint Schedule



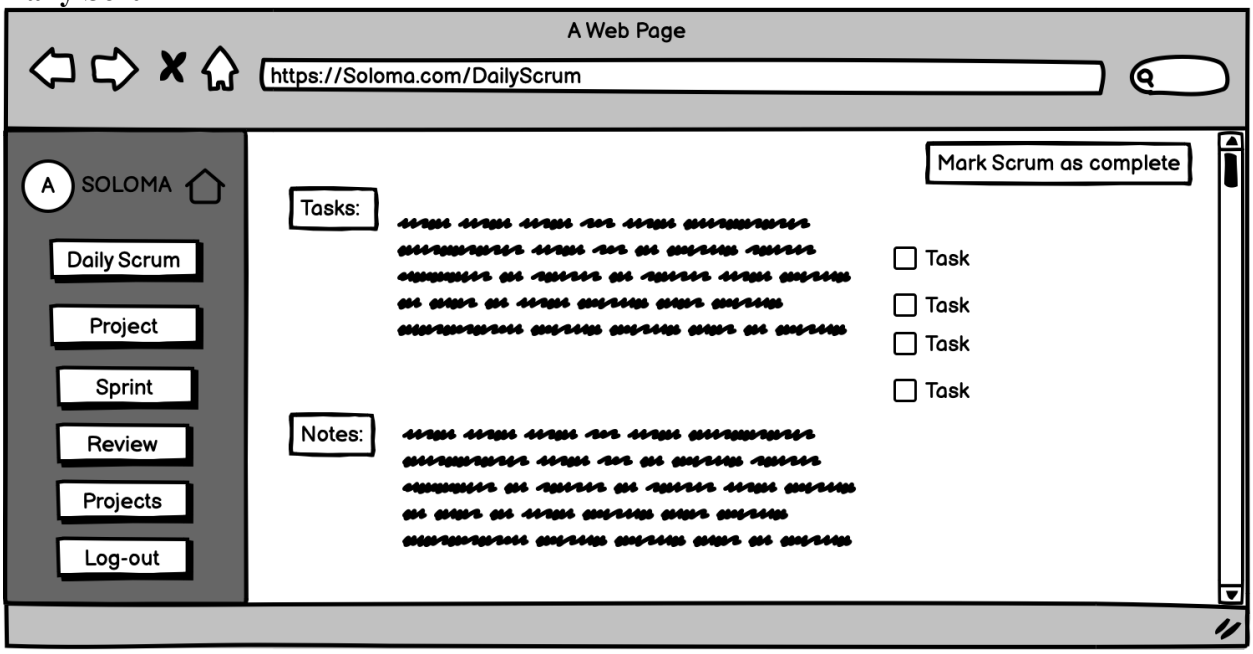
The Sprint schedule screen asks the user to input a name for the schedule as well as some notes and a select time and date that the sprint must take. Once the user is satisfied, they may select the save button to return to the previous screen (Create sprint). Since this screen is part of creating a new project, the menu bar is also limited to only the log-out button.

Dashboard



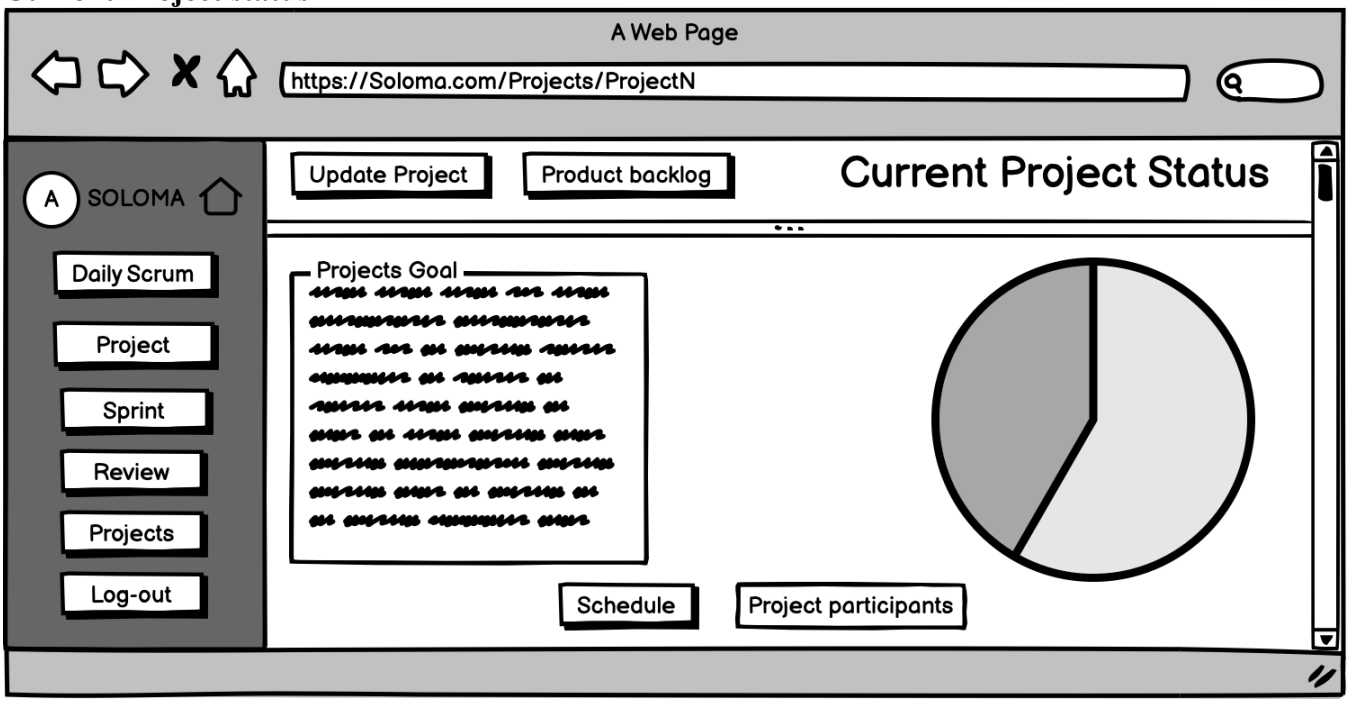
After finally creating a new project or selecting an existing project the user is directed to the home screen. Here all the menu options are available and guide the user to the elements of a project. This screen has the title of Dashboard because it is here where all the data and progress of the project is reported. There is also a calendar that helps the user see where they are in terms of scheduling. Overall, this screen will give the user most of the information they may need about how the project is going. Soloma aims to combine all the information from the project elements and display it in the best manner possible for the user. The home icon next to the title of the application will guide the user to this screen no matter where they may be only if the user is inside a project directory. The last two menu options allow the user to either go back to the projects directory or to log-out.

Daily Scrum



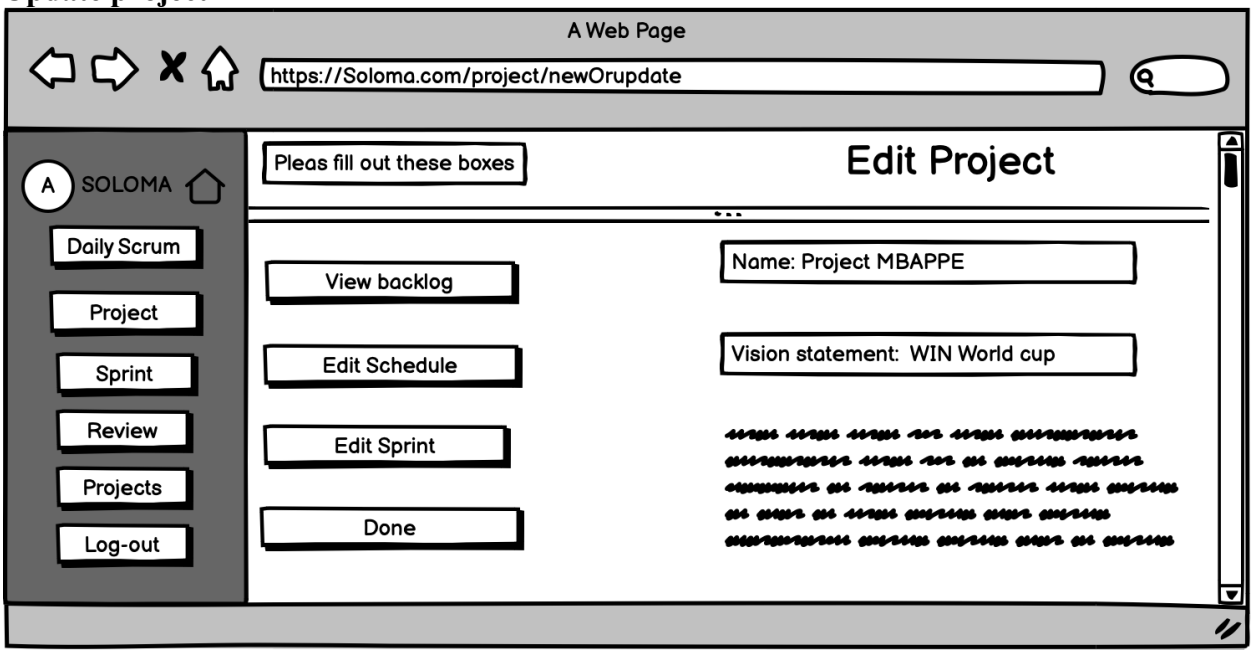
This screen comes from the menu bar to the left. Part of the SCRUM framework is to perform a daily standup (scrum). This screen helps the user perform a standup by review tasks and notes on the project.

Current Project status



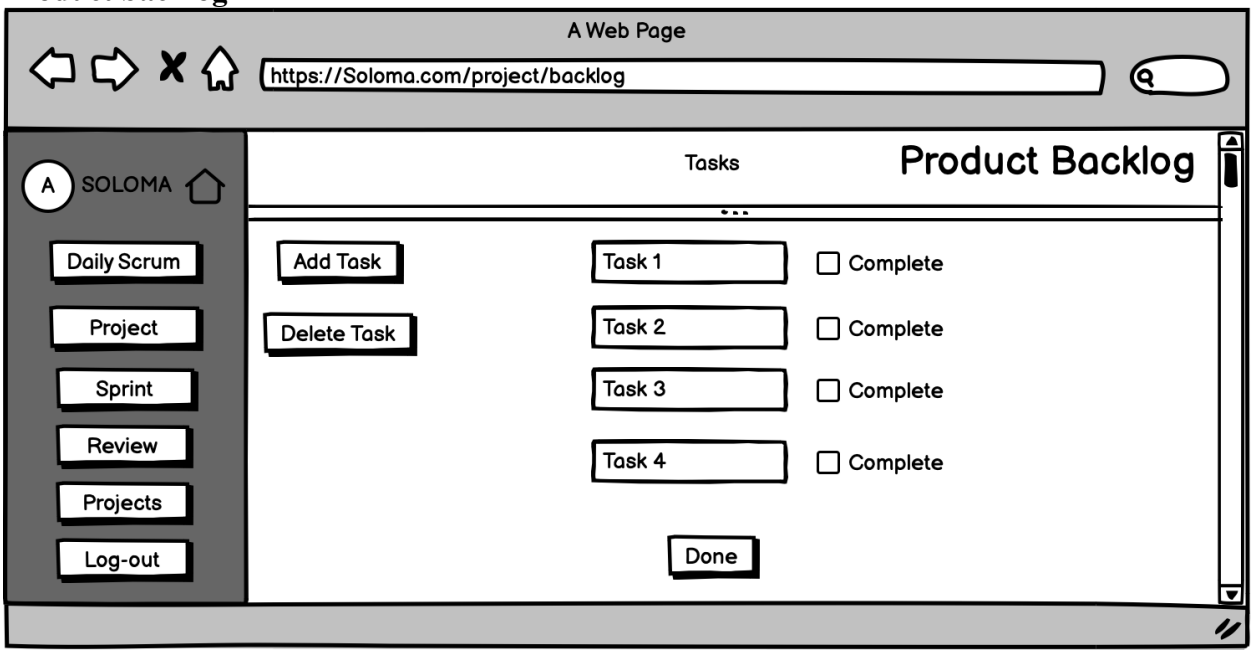
This screen is the current project screen. Different from the dashboard this screen allows the use to make changes to the current project. There are three buttons other than the menu on this screen. Update project takes the user to another screen so that they may update the project. Product backlog takes the user to the backlog so they may update it. Finally, schedule takes the user to the schedule of the project. To right of this screen is a pie chart that shows the project progress in terms of tasks completed.

Update project



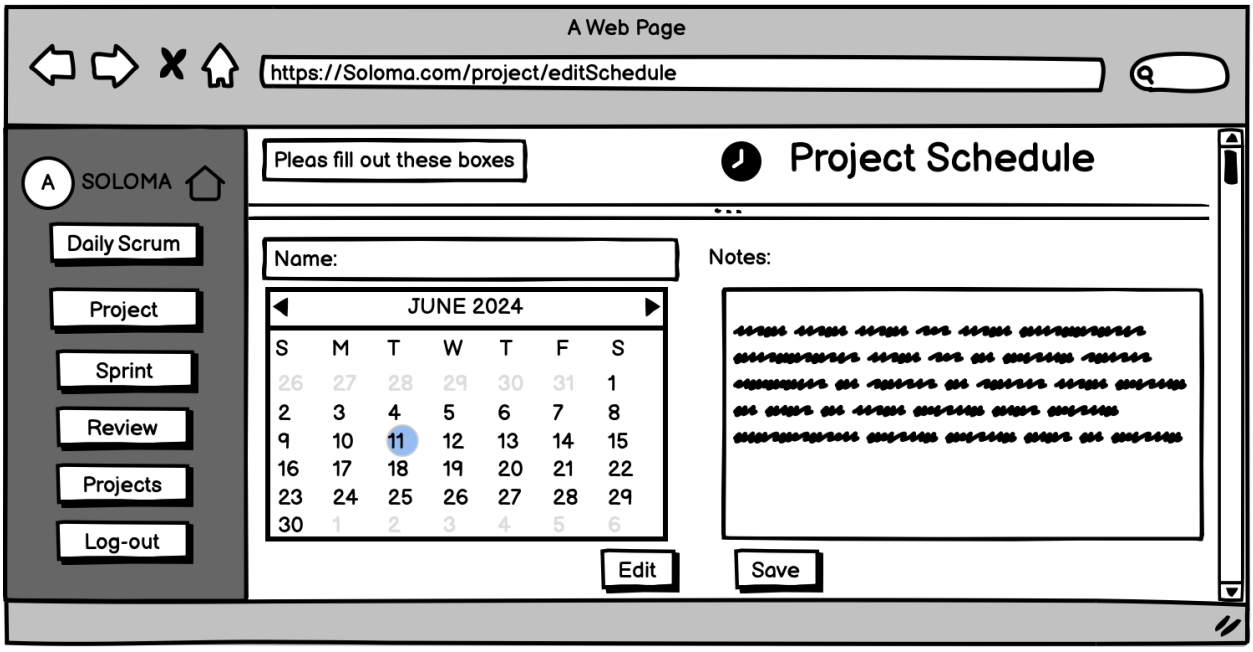
Update project is very similar to create project as it gives the user the same options of visiting the separte elements of a project. View backlog takes the user to the backlog, edit schedule take the user to the schedule. Edit sprint takes the user to the sprint of the project. Finally, done takes the user to the previous page.

Product backlog



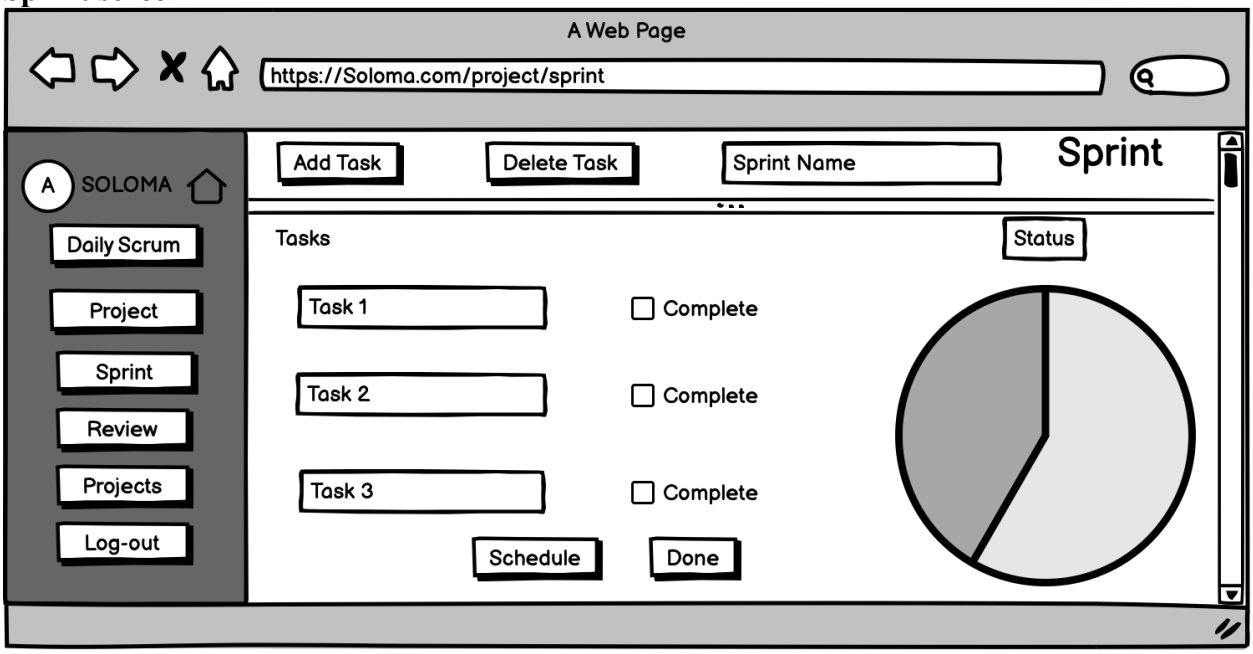
The product backlog screen allows the user to create, read, update and delete tasks from the project. When the button “Add Task” is pressed a column under tasks is added and it will have the option for a name. To edit a task, name the user can simply double click the column. The deletion process is the same for any item in this system. So, if the user selects the bin, they must enter their password to delete the task. To the right of each task there is also a checkbox which a user may check or uncheck to mark as complete. Since this instance of viewing the product backlog is already within a project directory the user also has access to all the menu options to the right. “Done” takes the user to the previous screen .

Edit schedule



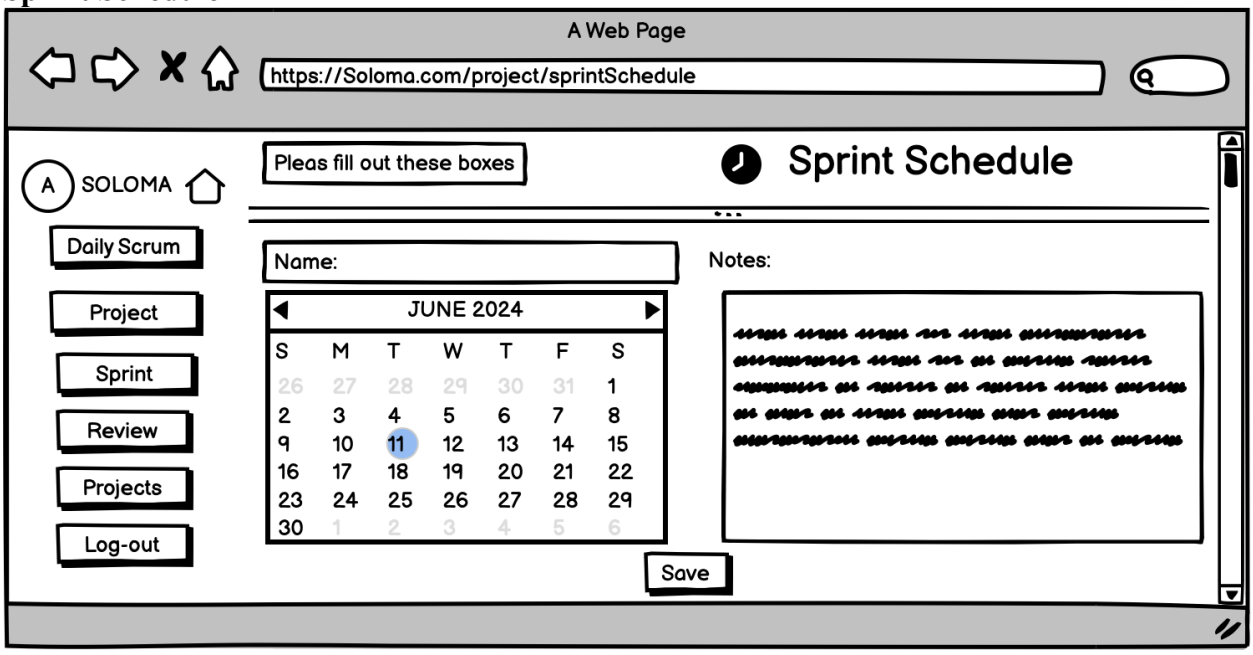
This screen is very similar to the “Create Schedule” screen when creating a new project. If the project already has a schedule or if it does not the user would only need to press the “Edit” button to edit the elements of this screen. Normally this would not be a common page to visit unless the user wanted to extend the deadline to a project. It is also noteworthy to point out that the calendar on this screen is not just intended for readability but that its main intention is to track the projects progress. This schedule helps determine the information that is outputted to the dashboard that a user sees daily.

Sprint screen



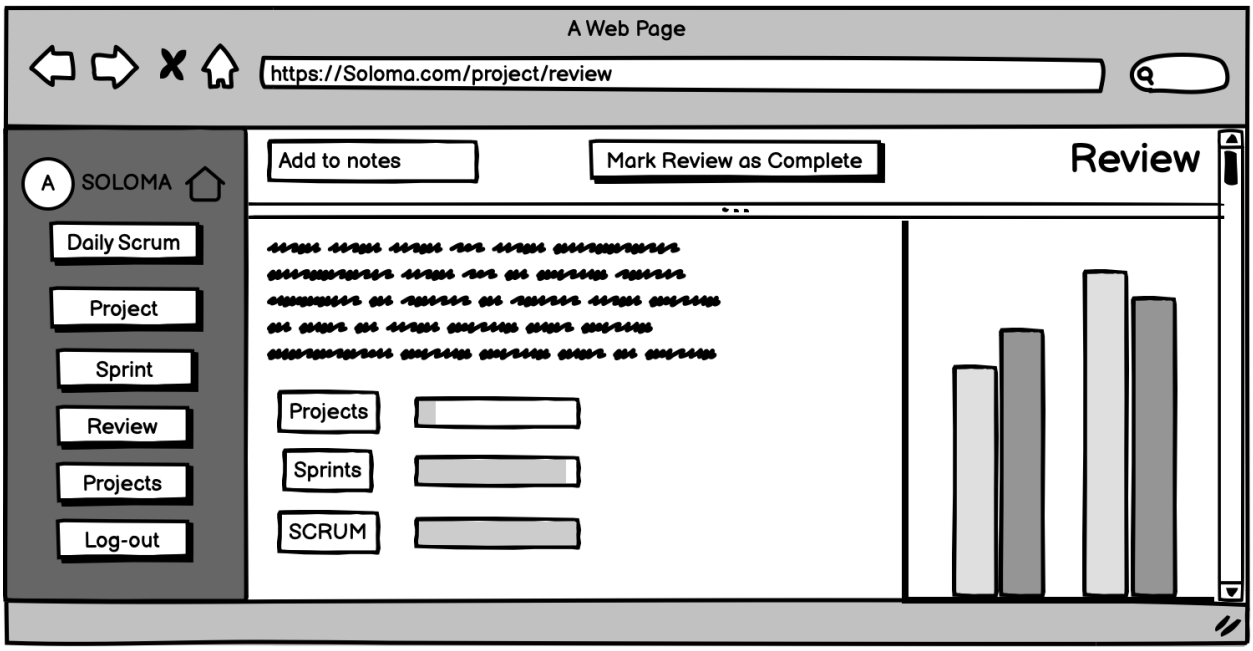
The sprint screen is the same as the one for creating a new sprint. The only difference is since the user is already inside a project directory, they are given access to all the items of the menu bar. This screen may also be accessed using the menu bar.

Sprint Schedule



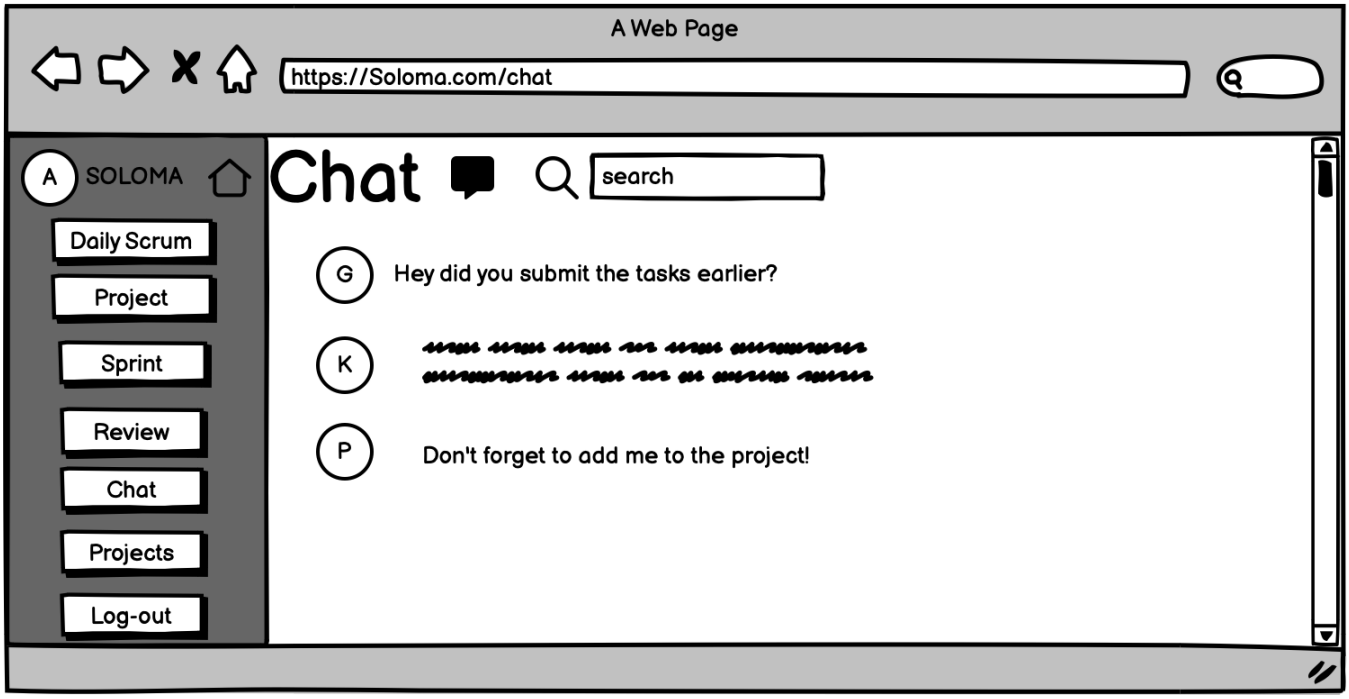
The Sprint schedule screen asks the user to input a name for the schedule as well as some notes and a select time and date that the sprint must take. Once the user is satisfied, they may select the save button to return to the previous screen (Create sprint). Unlike the other sprint schedule screen this screen is given access to all the menu options since it is inside a project. To edit any of the schedule elements the user simply selects “edit”.

Review



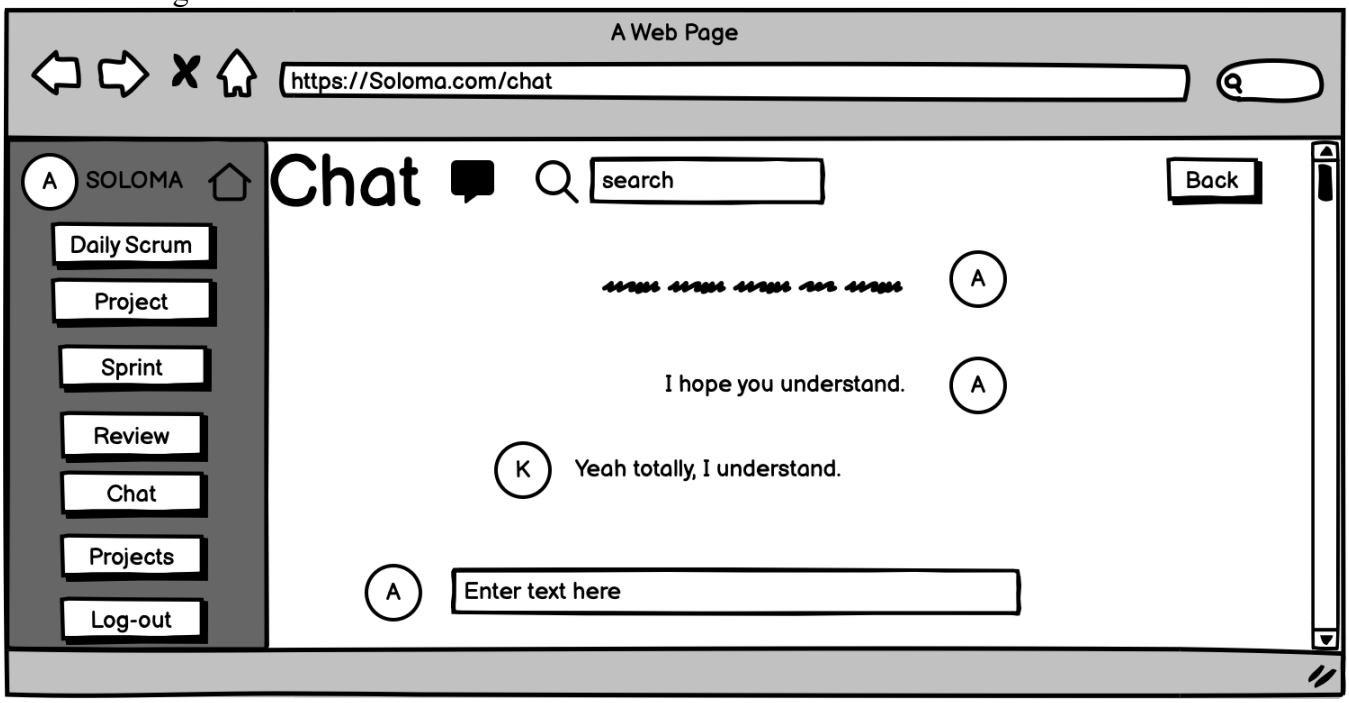
This screen gives the user more information on the project. It provides a graph and completion bars to show the user how elements of the project are going. A notes section at the top left is also set so that the user may take notes on the project. This review page is mostly set to be done once a sprint is complete and is a must before another sprint may be created. To mark a review as complete, both the sprint bar and the SCRUM bar must be complete.

Chat



This screen comes after selecting the chat button on the menu bar. It lets the user communicate with other team members in a company. When a user clicks on a chat it takes them to a different screen.

Direct Messages



This screen is the direct messages where the user can communicate the another team member.

5.4. Reports: "Formal Output" Design

Soloma will not generate any reports such as receipts, reports or invoices other than what is displayed in the usual screen interaction.

6. Appendices

6.1. Glossary

- SCRUM: An Agile methodology framework.

6.2. References / Bibliography

Cameron, A. (2024). *CSC 3150*. Lecture, Seattle ; Washington.
Larman, C. (2005). *Applying UML and patterns: An introduction to object-oriented analysis and design and the unified proces*. Prentice-Hall.

Jones, J. (2022). *sprint burn down chart*. What is a Sprint Burndown Chart & What is its Significance? UNICHRONE. Retrieved May 10, 2024, from https://unichrone.com/blog/agile/sprint-burndown- chart/.

Alder, G., & Koreng, I. (2005). Draw.io. *draw.io*. computer software, JGraph Ltd. Retrieved 2024, from https://www.drawio.com/.

Christie, J. (2023). *The Scrum Framework*. Retrieved 2024, from https://fatihyazici7.medium.com/overview-of-scrum-framework-845a84802c78.

6.3. Supporting documentation