

# CareCoord

# Design Document

A mobile application for coordinating elderly care

The Caretechers

Members: Aaron Heo, Brynnli Borrowman, Benjamin Hatch, Seng Horn Rith

Course: CS4000 Fall 2022 - Senior Capstone Design

Date: December 9th 2022

# Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>Executive Summary</b>	<b>4</b>
Background and Technical Requirements	5
<b>System Architecture</b>	<b>8</b>
<b>Personnel</b>	<b>9</b>
<b>System Features</b>	<b>10</b>
<b>Software Engineering Tools and Techniques</b>	<b>12</b>
1. Agile vs. Traditional	12
2. Tools + Main libraries	12
3. Code Reviews	12
4. Documentation	12
5. Team Communication	12
6. Team Meetings	13
<b>Timeline</b>	<b>14</b>
<b>Appendix A: UI Sketches</b>	<b>17</b>
Figures 1a & 1b	17
Figures 2a & 2b	18
Figures 3a & 3b	19
Figures 4a & 4b	20
Figures 5a & 5b	21
Figures 6a & 6b	22
Figures 7a & 7b	23
<b>Appendix B: Use Cases</b>	<b>25</b>
Use Case 1: Edit frequency of visits - Aaron Heo	25
Use Case 2: Edit the tasks list- Aaron Heo	25
Use Case 3: Enter health vitals- Aaron Heo	25
Use Case 4: Recording a Caretaking Visit - Ben Hatch	26
Use Case 5: Communication with other caretakers - Ben Hatch	26
Use Case 6: Viewing the caretaking schedule - Ben Hatch	27
Use Case 7: Signing In/Out of the App - Ben Hatch	27

Use Case 8: Edit profile - Seng Rith	28
Use Case 8a: Change Notification Settings Preferences - Seng Rith	28
Use Case 9a: Edit/Set Availability for care duty - Seng Rith	29
Use Case 9b: Set out of town period - Seng Rith	29
Use Case 9c: Set “on call” caretaker - Seng Rith	30
Use Case 10: Look at past visits info - Seng Rith	30
Use Case 11a: Creating notes - Brynnli Borrowman	31
Use Case 11b: Viewing notes - Brynnli Borrowman	31
Use Case 11c: Searching notes - Brynnli Borrowman	31
Use Case 12: Creating a CareCoord Account - Brynnli Borrowman	32
Use Case 13a: Creating a CareCoord Group - Brynnli Borrowman	32
Use Case 13b: Inviting Others to Your CareCoord Group - Brynnli Borrowman	32
Use Case 13c: Joining a CareCoord Account - Brynnli Borrowman	33
Use Case 14: Viewing Contact Information for Group Members - Brynnli Borrowman	33
<b>Appendix C: Revisions</b>	<b>34</b>
Table of Contents	34
Executive Summary	34
Background and Technical Requirements	34
System Architecture	34
Personnel	34
System Features	34
Software Engineering Tools and Techniques	34
Timeline	34
Appendix A: UI Sketches	34
Appendix B: Use Cases	34

## Executive Summary

Many of us will become a primary caretaker of an elderly relative at some point in our lifetime. However, we will likely have busy lives of our own, and many people simply don't have time to check up on elderly relatives consistently. Leaving elderly relatives alone at home without consistent caretaking visits can be incredibly dangerous physically, mentally, and financially (food can run out, heating/cooling issues can arise, falls can become lethal, bills can go unpaid, etc.).

Most families attempt to ensure elderly loved ones are consistently taken care of by sharing the burden of caretaking between multiple family members. However, coordinating care between family members can be messy. Family members could forget which days they are responsible for caretaking visits, or neglect important caretaking tasks (such as refilling medication). Thus, by attempting to split up caretaking visits, the quality of caretaking for the elderly loved one might actually be *worsened*, which paradoxically exposes the elderly loved one to the same dangerous situations previously described!

Our project, CareCoord, is a mobile application for family caretakers (the users) that will mitigate the problems of shared caretaking by accomplishing four main goals, which are:

1. Ensure that the elderly relative does not go more than "X" days (defined by the user) without a wellness-check. This will be accomplished through an in-house scheduling solution that allows caretakers to coordinate different tasks and schedule time off, as well alert caretakers (SMS, email) that a visit is needed.
2. Ensure that crucial caretaking information is centralized and easily accessible. This will be achieved by an in-app group messaging system for conversations about caretaking, as well as a notes section. This messaging system and notes will have search functionality so that caretakers can search for any information needed.
3. Make sure that caretakers complete all of the essential household chores and perform a proper wellness check. In order to accomplish this goal, we will implement a feature to create a checklist of items that need to be completed around the house or for the elderly person.
4. Facilitate better tracking of the elderly person's health. There will be a feature that allows the caretaker to enter health vitals (heart rate, temperature, etc.) which will then be displayed using auto-generated graphs so that a caretaker can easily recognize positive or negative patterns in the elderly person's health.

After extensive discussion with people that have shared caretaking responsibilities with other family members, we are very confident that our project will ease the burdens of shared caretaking and bolster the safety of elderly loved ones living alone if the goals above are accomplished.

## Background and Technical Requirements

### Why CareCoord?

Elderly loved ones are subjected to a variety of serious dangers as a result of poorly coordinated/low quality shared caretaking. Bills can go unpaid, heating/cooling issues can arise, food can run out, fall risks can become more lethal, and more. Unfortunately, shared caretaking between family members is often not executed well. Family members often forget which days they are responsible for visiting elderly relatives, or neglect important caretaking tasks, such as checking if there is food in the pantry. Additionally, important caretaking information is often “lost in the weeds” of various text messages between caretakers. All of these scenarios expose elderly loved ones in need of caretaking to the dangers previously described.

Our project, CareCoord, will ensure that family-centered shared caretaking is properly coordinated and executed correctly, therefore mitigating the aforementioned dangers. It will accomplish this by providing family caretakers with a mobile application that will facilitate easier coordinated scheduling and reminders for caretaking visits, sharing of relevant caretaking information between family members, caretaking visit checklists, and health tracking mechanisms, all in *one place*.

Currently, family members responsible for shared caretaking tend to use existing tools such as iMessage, Google Calendar, Notes, the Health App, or a combination thereof to achieve the proposed functionality of CareCoord. Having to use many different tools can be overwhelming. There is no current platform that keeps track of the different aspects of caretaking in one place and in an easy to use manner. Additionally, these tools are not tailored for caretaking of a loved one. CareCoord is unique because it provides a centralized platform for caretakers, and it is designed specifically for caretaking.

### Required Technologies

React Native: React Native is a frontend framework that is very similar to React for the web. It has the same great features of the web version such as component based html elements and hooks to keeping track of and manipulating various state variables. React Native was the framework of choice because it allows us to seamlessly create user interfaces for mobile devices for both Android and iOS using one codebase. In addition, our team possesses a fair amount of prior experience with either the web version of React or React Native.

Express.js: Express.js is a JavaScript framework that allows us to easily implement a web server that will contain all of our REST API endpoints.

MySQL: For our database, we will be using a MySQL database. We chose this database because it was what we had the most experience with as a team.

AWS: AWS is a cloud platform that provides many services for building and maintaining software. We will mainly be using AWS to host our servers (frontend, backend, database).

SendGrid: If an elderly person has not been checked on for a certain amount of time, our mobile app will send a notification to notify users that they need to be checked on. We will send these notifications using a push notification on mobile devices as well as email. SendGrid provides an API that allows us to easily send these email notifications.

Google OAuth 2.0: Google OAuth 2.0 will allow our team to easily implement login functionality in a simple and secure manner. In addition, Google OAuth 2.0 allows us to pull important information such as an email when a user logs in using Google.

Docker: Docker is a technology that allows us to containerize our project so that we can start up a clean version of our mobile application every time. If we run into a situation where we need to restart our app because of an error or unexpected behavior, we will be able to consistently restart it to a working state.

Git: Version control is a crucial part of software development. It allows us to keep track of changes that have been made and combine different branches. Git is widely used and our team has previous experience with it which was why we decided to utilize this technology.

### **Assets & Engines**

Although most of the mobile application components will be built from scratch, we can leverage certain resources in order to speed up development time. There is a rich library of open-source React Native mobile UI components available for reuse online (that can be installed through Node JS's npm package manager). This will allow us to develop the frontend (the mobile interface) much faster.

Rather than implementing our own secure login functionality from scratch, we will be using Google OAuth (see "Required Technologies" section above). This will save us plenty of development time, as well as ensure proper security protocol is being adhered to.

Most of the application will be designed and developed from scratch, from the structure of the database to the individual API endpoints on the backend. This will provide us with flexibility to make the app function exactly as desired (see “UI Sketches”).

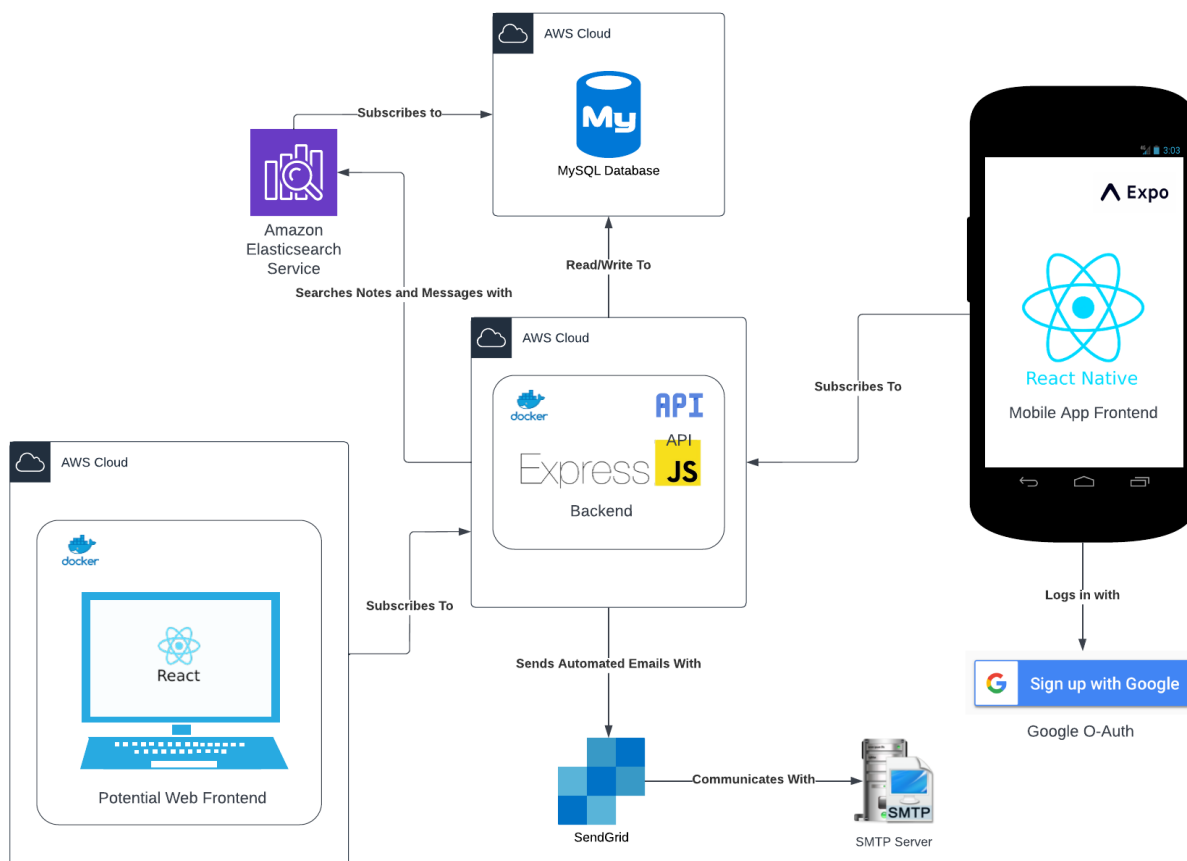
### **Software/Hardware Requirements**

The CareCoord application will be developed and tested as a mobile app for iOS on an Apple iPhone. Our clients will download an iOS application to their iPhone, which should be updated to the most recent operating system. At the moment, we do not foresee any abnormal memory usage from our app. It will be cloud-operated, without the need for excessively large files on the client’s iPhone.

We are planning on using Google OAuth 2.0 as our account management system, and this will require our clients to have a Gmail account.

Since we are using React Native to develop the CareCoord application, it will likely work on Android phones as well. However, we are focusing on iPhone development, and are considering Android compatibility as a stretch goal. Thus, Android compatibility will not be guaranteed.

## System Architecture



Our project has three main components - a database, a backend (API), and a frontend (mobile application). The database will be MySQL, as the data we will be dealing with will be structured (relational), and MySQL is relatively inexpensive compared to other databases. The frontend will fetch data from the backend's HTTP REST API. The backend (built using the Express JS framework) will fetch data from the MySQL database, or through Amazon's ElasticSearch service, which will help improve search results for caretakers' notes and messages. This ElasticSearch service will also connect to the database in order to prepare its searching algorithm. All creation of data in the database will be executed by SQL queries from the backend. The backend will also use the external SendGrid service API to send email notifications. The mobile app frontend will be built in React Native. It will use Google's O-Auth service to handle the security/login functionality for users. Because React Native is similar to React, it is possible that we will also build a web application frontend in React as well, though this is definitely a stretch goal. The backend will be containerized using Docker in order to ensure consistency once it is deployed. The backend and database will both be deployed using Amazon Web Services. The mobile application will need to be approved by the Apple app store in order to be deployed. For now, we will run a local version of our app's frontend using a service called Expo.



## Personnel

Ben Hatch - React Native (Mobile) Frontend (Tasks, Calendar, Settings)

Out of all of the group members, Ben has the most experience with React Native. Thus, he will be responsible for the most customized/demanding interfaces of CareCoord's mobile application frontend, which will be built in React Native (see architecture diagram). Ben also has significant frontend development experience at his current job, which he has worked at for over 2 years. He has worked on a calendar UI in the past.

Seng Horn Rith - React Native (Mobile) Frontend (Notes, Messaging System, Vitals Tracker)

Seng has a solid understanding of how React Native frontend works and has worked with Android application development. Throughout his undergraduate career, Seng did well in adapting and learning new frameworks quickly whenever he is conditioned to.

Brynnli Borrowman - Express.js Backend, Localhost implementation, Sendgrid

Brynnli will be responsible for the backend database and API implementation. Brynnli has experience with databases, AWS, and APIs through her jobs as a Data Engineer at SLCC, a Software Engineer Intern at Pluralsight, and as a TA for the Database Systems class.

Aaron Heo - Express.js Backend, Elasticsearch

Aaron will be responsible for the backend server which will utilize the Express.js framework. This backend server will house all of our API endpoints that the front end will interact with. Aaron has experience with using an Express.js server in a professional environment which is the reason he will be responsible for the backend. While Aaron does not have direct experience with Elasticsearch, he has experience with other areas of AWS and is willing to research the technical details of integrating Elasticsearch into our mobile application.

## System Features

### Rank 1:

- Core database tables
- Localhost implementation
- Basic UI/UX implementation (pages without functionality)
  - Notes: a place to store information in individual pages
  - Messaging: a basic group messaging system
  - Health Vitals: track health vitals such as blood pressure and heart rate
  - Calendar: a basic calendar
  - Tasks: a checklist of scheduled tasks
- Barebones Express server with a healthcheck route

### Rank 2:

- Deploy app on AWS
  - DB
  - Express server
- Google oAuth login
- Database test/example data
- Working API with tests and test data
- Settings Page
  - Notification preferences
  - Group settings
    - Creating a group
    - Inviting others to groups
    - Joining a group
  - My availability
    - Set out of town time
  - Account profile
- Calendar Page
  - See who is scheduled to visit mom on specific days
- Message Page
  - Send messages
  - Pin messages
- Task page
  - New tasks
  - Edit tasks
- Health vitals
  - Display health vitals (without graphs)
  - Enter health vitals

- Notes Page
  - Create/Edit/Delete Note
  - View sorted notes (default last edited note)
  - Search notes by title

**Rank 3:**

- Use Sendgrid to send email notifications
- Display health vitals using graphs
- Integrate with Google Calendar or Device's Calendar
- Get it on App Store / Google Play Store
- Search messages
- Set an "on-call" caretaker who will help in emergencies (this could be based on caretaker availability or volunteer-based)
- Allow easy use of multiple caretaking groups

## Software Engineering Tools and Techniques

### 1. Agile vs. Traditional

We will be using agile for planning our development, as our group has the most experience with this development cycle. This is also probably the best cycle because we aren't entirely sure of *all* the properties/features the codebase will have.

### 2. Tools + Main libraries

- VS code
- React Native (frontend)
- Express (backend server)
- MySQL (database)
- Jest + supertest (api testing framework)
  - Manual frontend tests

### 3. Code Reviews

Our team will follow a “single signoff” protocol when it comes to reviewing code. We have determined that this is best because a complete signoff could cause blockers when different modules of the project depend on one another's development. However, there are some cases where a full team signoff will be warranted/executed, such as for crucial parts of the codebase that have a significant effect on other group members' code.

### 4. Documentation

We will be creating a README file for our frontend and backend code. This README will contain a “Getting Started” section that explains how to get the program up and running as well as common troubleshooting steps.

In our codebase, we plan to comment after creating code. We aim to write clean and easily readable code first and only include inline comments when we are writing complex code. We also plan on creating header comments for each method we implement.

### 5. Team Communication

We will be using Discord and Zoom as our main communication methods. Occasionally, we will meet in person for tasks such as presentation rehearsals.

#### 6. Team Meetings

We plan to meet every Wednesday and Friday. During the meetings, we will discuss what we have done and the roadblocks we face.

## Timeline

	Aaron	Ben	Brynnli	Seng	Major Benchmarks
<b>ALPHA</b>					
<b>Week 1</b>	Create a plan for how we are going to handle instances of tasks that repeat.	Create a non-functional skeleton UI for settings page.	Refactor database primary keys as needed.	Develop functional interfaces for pinning and searching messages.	
<b>Week 2</b>	Create GET /calendar/group/:groupId endpoint.	Make the Settings interface functional (connect it to the backend).	Work on the more complicated Tasks endpoints.	Fetch data using authenticated tokens.	Core functionality of Settings Interface complete
<b>Week 3</b>	Complete endpoints for editing visits(complete tasks).	Develop deleting/editing task functionality on the frontend.	Finish the more complicated Tasks endpoints.	Develop delete, edit, search and sort notes functionality on the frontend.	Rank 3 Notes functionality complete.
<b>Week 4</b>	Complete endpoints for volunteering and un-volunteering for a visit.	Connect the frontend's calendar interface to the backend.	Work on User endpoints.	Develop User Registration interface to allow users to register and join caretaking groups.	Calendar page connected to backend.
<b>BETA</b>					
<b>Week 5</b>	Look into how we can display health vitals with graphs.	Create a fully functional "Record a Visit" interface in the frontend (see UI sketches).	Work on Group endpoints.	Create the health vitals skeleton page interface.	Users can now record caretaking visits.
<b>Week 6</b>	Complete	Create an	Work on health	Connect health	Development

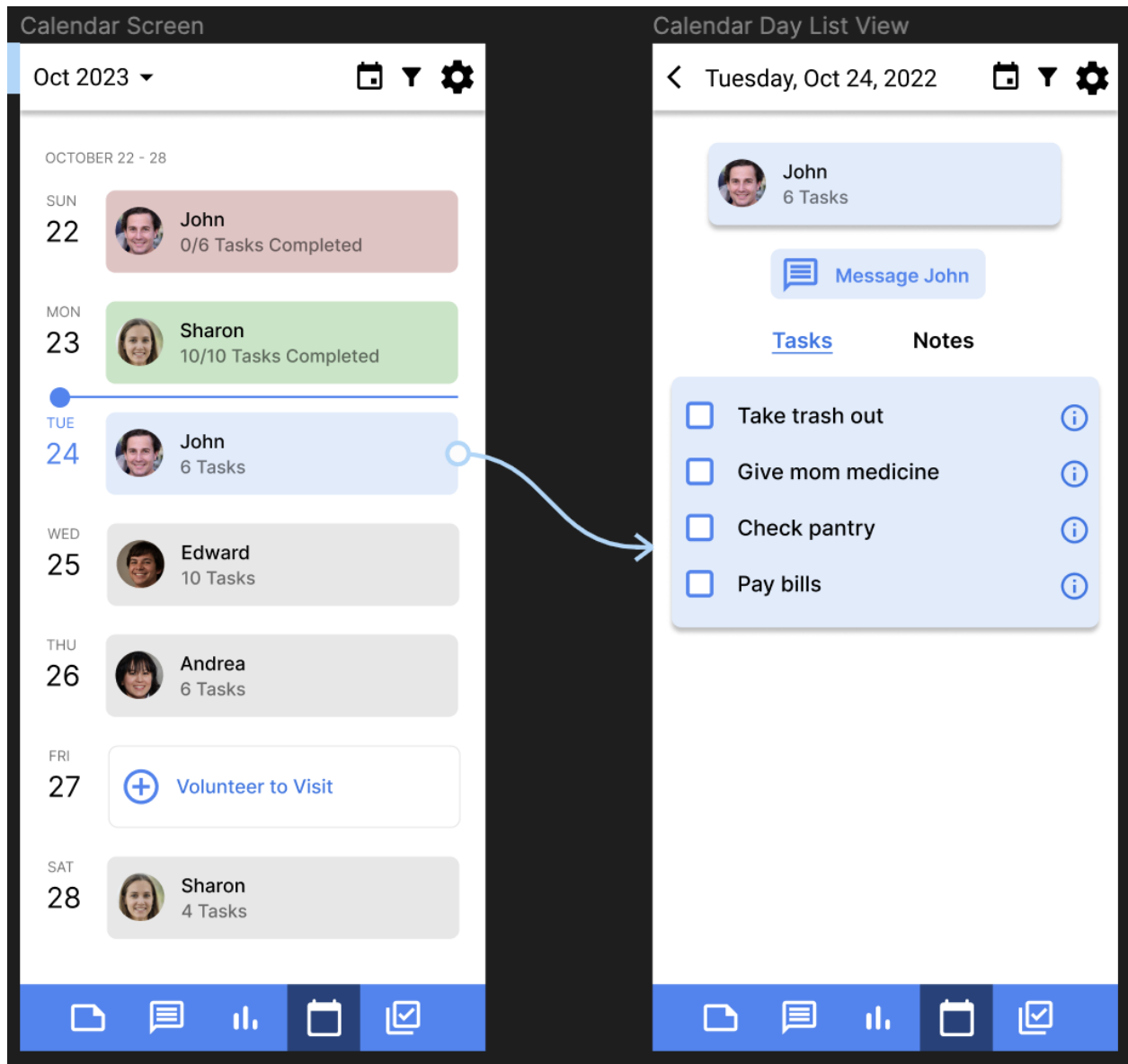
	endpoints for health vitals.	interface that enables users to view specific caretaking visits from the Calendar page (see UI sketches).	vital endpoints	vitals frontend to the backend to fetch data and allow users to add health vitals information.	for the Health Vitals and Calendar interfaces complete.
<b>Week 7</b>	Finish off User endpoints.	Automate the sending of a few push notifications from the application.	Work on message endpoints.	Refurnish messaging page to become fully functional and allow direct messaging.	Message interface development complete.
<b>Week 8</b>	Work on endpoints for notifications using Sendgrid.	Finish developing push notifications for the calendar, settings, and task sections of the app.	Finish endpoints needed for push notifications.	Develop graph displays for showing health vitals information.	Push notifications are fully functional.
<b>FINAL</b>					
<b>Week 9</b>	Update/add new endpoints that are needed for multiple group support.	Allow users to be part of multiple caretaking groups (through further development of the settings interface).	Refactor database as needed to allow users to be part of multiple groups.	Develop a page to view caretaker information in the same group.	Users are allowed to participate in multiple caretaking groups.
<b>Week 10</b>	Start looking into the process of releasing our app on the app store.	Create tests for the Calendar and Tasks interfaces.	Testing for Calendar and Tasks endpoints.	Create tests for messaging user interface.	
<b>Week 11</b>	Look into creating google calendar events	Create tests for the Settings interface.	Testing for settings and user endpoints.	Test the Notes and User Registration	

	based off of visits a user volunteers for.			interfaces.	
<b>Week 12</b>	Fix any last minute bugs and clean up our codebase.	Polish styling/flow of the Calendar, Tasks, and Settings interfaces to prepare for demo day!	Polishing endpoints and backend codebase.	Polish styling/flow of Notes, Messages, & Health vitals interfaces.	Development and testing of the application is complete.

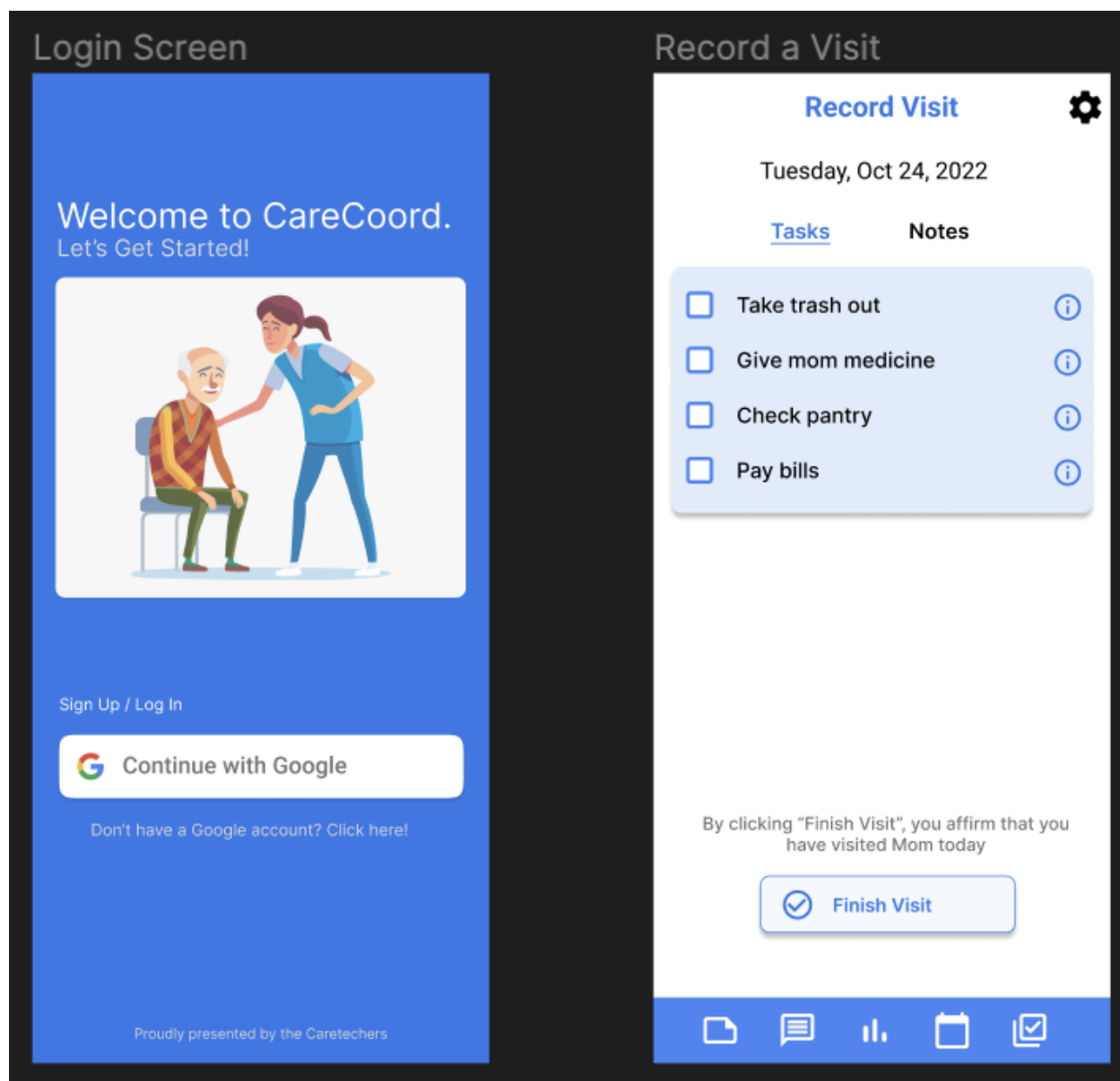


## Appendix A: UI Sketches

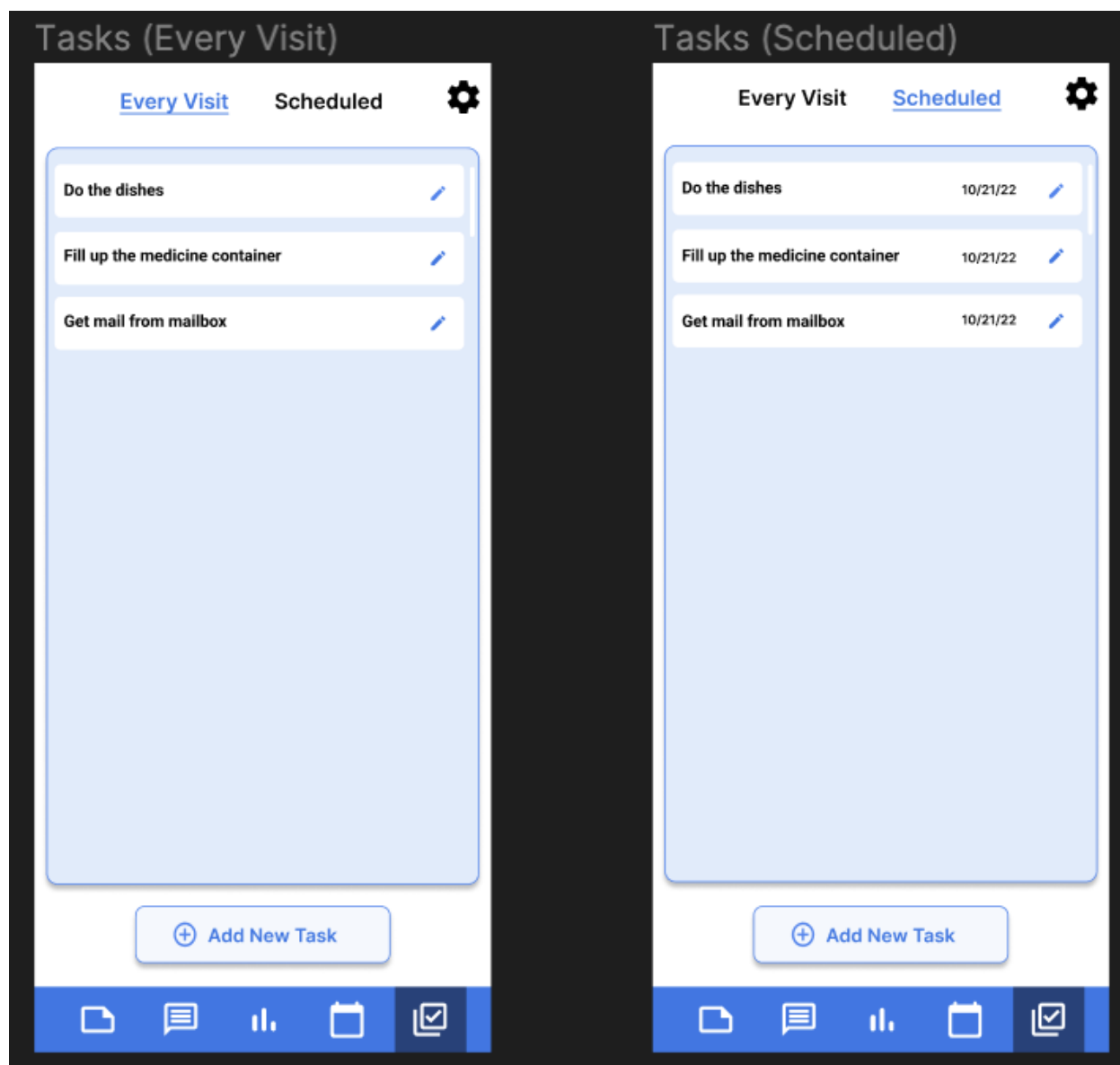
Figures 1a & 1b



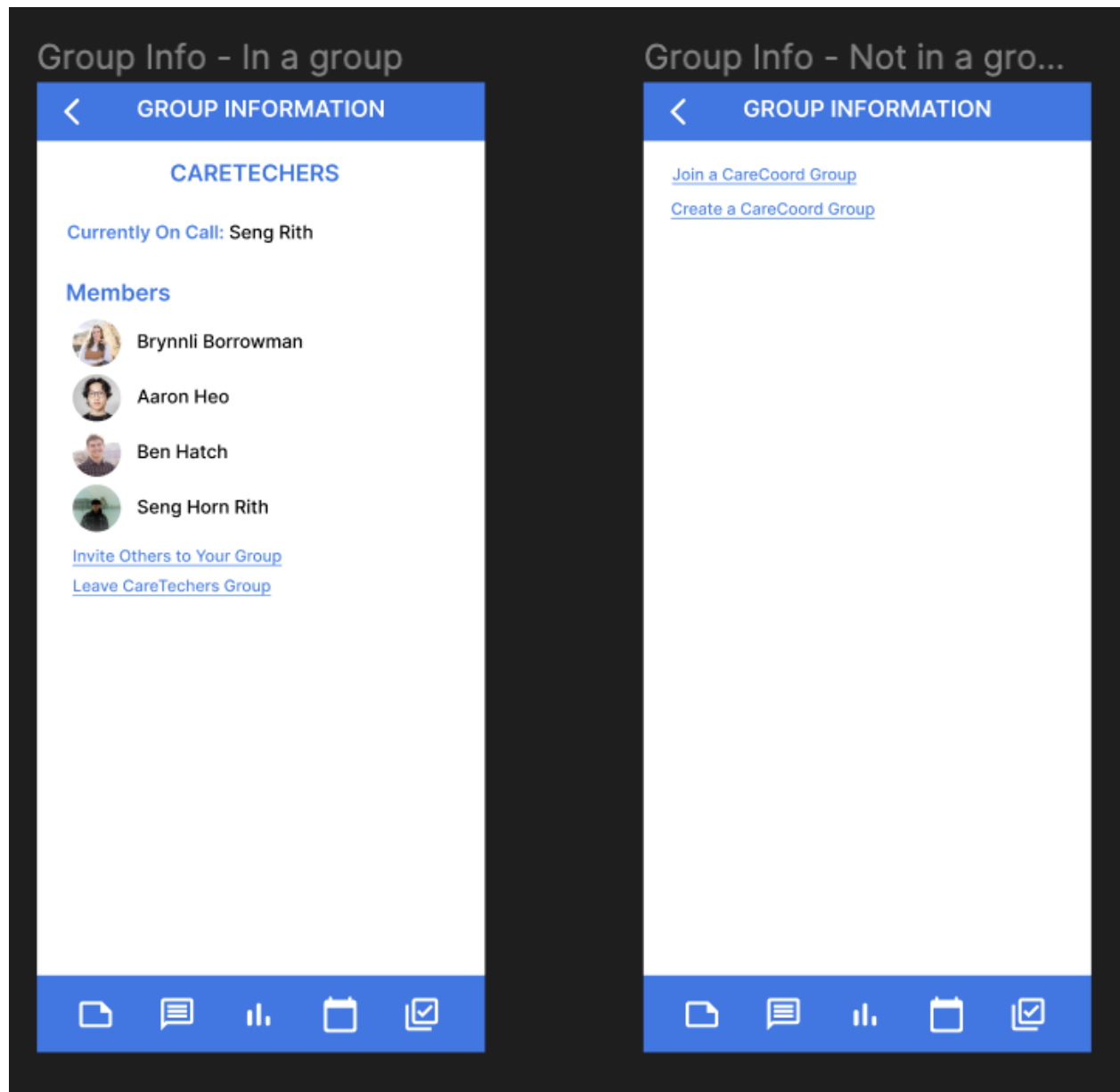
Figures 2a &amp; 2b



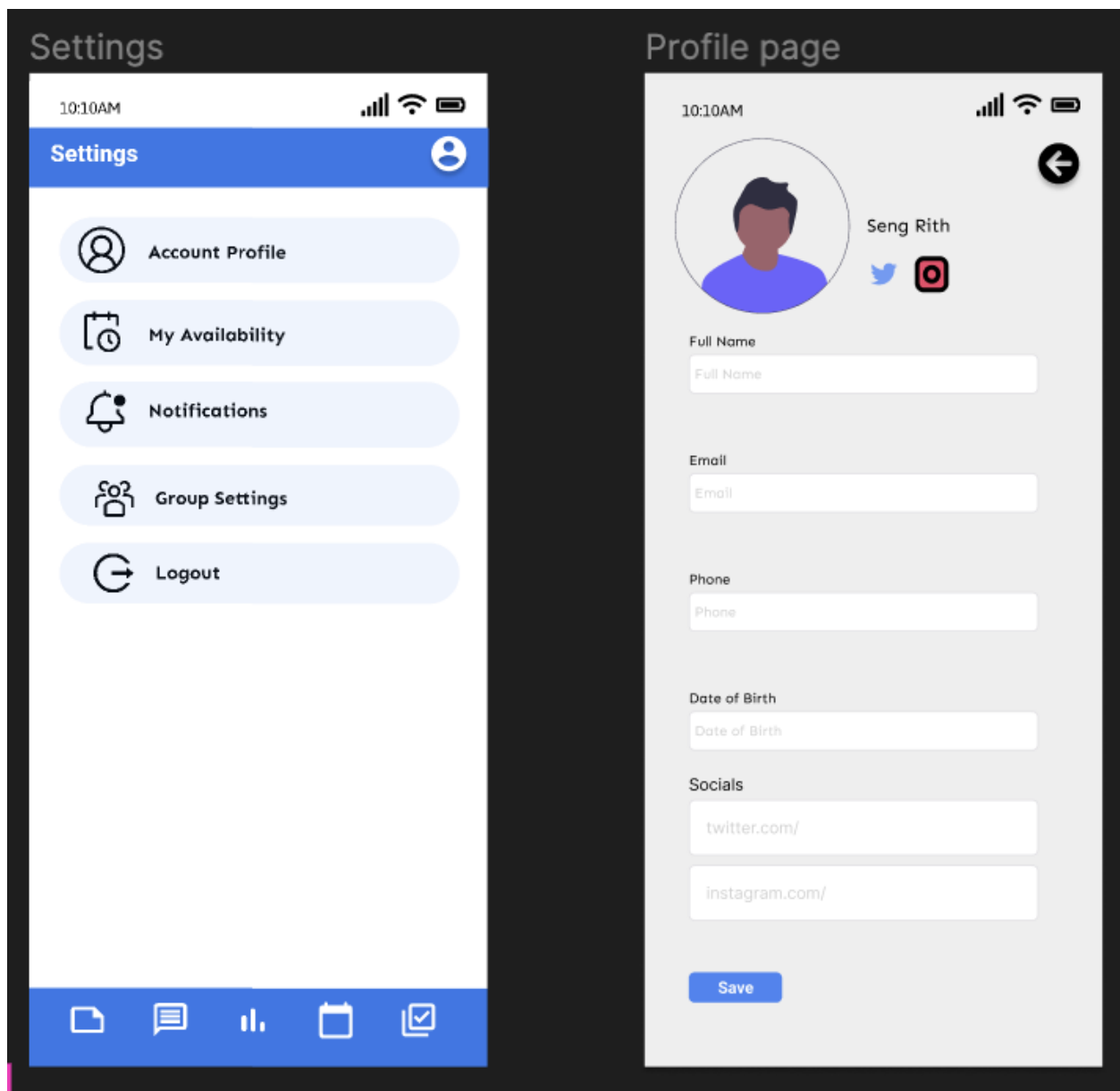
Figures 3a &amp; 3b



Figures 4a &amp; 4b



Figures 5a &amp; 5b



Figures 6a &amp; 6b

**CareTechers Chat** 

Just checked on Mom! She's doing fine.

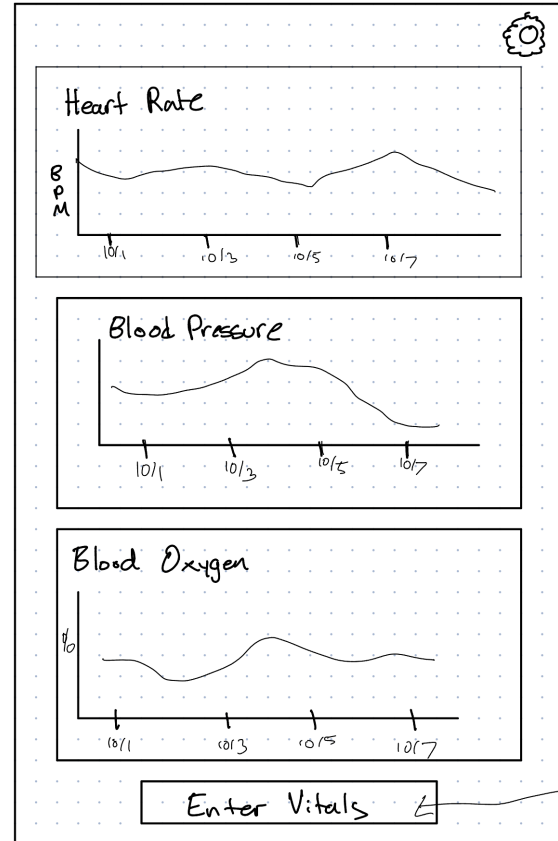
☺ Seng  
Good to hear!

☺ Ben  
Thanks


☺ Aaron  
I'll do the next one!

🔍



Figures 7a &amp; 7b

Notes 

Sort By Date v






---


Groceries  
10:01 am Eggs

---






Garage Code  
4/9/22 1234

---

< Groceries 

Eggs  
Milk

Figures 8a &amp; 8b

⏪

## Availability Schedule

Aaron's Group

Sat		_____
Sun		_____

Ben's Group

Sat		_____
Sun		_____

Brynnli's Group

Sat		_____
Sun		_____

Seng's Group

--	--

⏪

## Notifications

☒ Email Notification

☒ Mobile Push Notification



## Appendix B: Use Cases

### **Use Case 1: Edit frequency of visits - Aaron Heo**

- Actor/User: Caretaker
- User Story: A caretaker needs to set the frequency of visits that need to be made to the elderly person. If the elderly person has not been visited for this certain period of time, they will receive a notification via text and/or email.
- Course of Events:
  1. Login
  2. Go to settings page
  3. Go into the notifications menu
  4. Update notification preferences
- Exceptions/Alternates: none
- Related UI: Figure 5a

### **Use Case 2: Edit the tasks list- Aaron Heo**

- Actor/User: Caretaker
- User Story: Caretakers need to know what tasks need to be done on each visit so they need to add tasks to the todo list. These tasks will be organized into recurring tasks and one-off scheduled tasks.
- Course of Events:
  1. Login
  2. Navigate to the checklist page
  3. Click new task button
  4. Fill out modal for new task
  5. Click save task button
- Exceptions/Alternates:
  - Alternate: A simple reminder app could be used such as the “Reminders” app on the iPhone, but it does not integrate into the caretaking calendar.
- Related UI: Figure 3a, Figure 3b

### **Use Case 3: Enter health vitals- Aaron Heo**

- Actor/User: Caretaker

- User Story: It is important for caretakers to log health vitals such as heart rate and blood pressure of the elderly person. This way, doctors or professional caretakers will be able to see a history of the elderly person's health or find patterns in their health.
- Course of Events:
  1. Login
  2. Main caretaking group page is presented
  3. Navigate to health vitals page
  4. Click desired health vital
  5. Fill out health vital modal
  6. Click save
- Exceptions/Alternates: none
- Related UI: Figure 6b

#### **Use Case 4: Recording a Caretaking Visit - Ben Hatch**

- Actor/User: A family member responsible for taking care of an elderly relative (Mom)
- User Story: As a caretaker, I want to be able to track what days I have visited my elderly relative and record the tasks that I complete each visit.
- Course of Events:
  1. Get a reminder from the CareCoord app to visit Mom
  2. Go visit Mom
  3. Navigate to the "Record a Visit" page in the app
  4. View caretaking tasks for that day on this page and mark them as completed as I finish them
  5. Click "Done" to finish recording the caretaking visit
  6. See that the caretaking calendar is now updated to show that I visited mom and completed all caretaking tasks today
- Exceptions/Alternates:
  - Exception: If I do not visit mom, the app will alert myself and other caretakers that someone needs to check up on her.
- Related UI: Figure 2b

#### **Use Case 5: Communication with other caretakers - Ben Hatch**

- Actor/User: A family member responsible for taking care of an elderly relative (Mom)
- User Story: As a caretaker, I want all relevant caretaking communication to be in one place and easily digestible so that information is not "lost in the weeds" between other caretakers.
- Course of Events:

1. Learn some new important caretaking information (let's say Mom no longer needs to take medication)
  2. Open the CareCoord app and navigate to the "Messages" page in the app
  3. Type "medication" in the search bar to see if anyone else has sent a message about this information already
  4. Construct message about Mom's medication. ?Give message a "importance" rating of %?
  5. Click "send" to notify other caretakers about this information
  6. Click "pin" to make sure this message is not "lost in the weeds" as other messages are sent in the future
  7. Click on the "pinned messages" panel to make sure the message is stored there
- Exceptions/Alternates:
    - Alternative: Users can forward text messages they receive to the CareCoord app for storage (would need to interface with Apple iMessage)
  - Related UI: Figure 6a

#### **Use Case 6: Viewing the caretaking schedule - Ben Hatch**

- Actor/User: A family member responsible for taking care of an elderly relative (Mom)
- User Story: As a family member who is sharing the responsibility of taking care an elderly loved one with my siblings, I want to see who is visiting mom on specific days, as well as the caretaking tasks they need to complete on those days.
- Course of Events:
  1. Open CareCoord application
  2. Navigate to the "Calendar Page"
  3. See a calendar that has my siblings' names next to the corresponding days they are responsible for visiting Mom
  4. Click on a specific day on the calendar
  5. See a summary of who is visiting Mom that day and what tasks need to be completed.
- Exceptions/Alternates:
  - Alternative: Users can forward text messages they receive to the CareCoord app for storage (would need to interface with Apple iMessage)
- Related UI: Figures 1a & 1b

#### **Use Case 7: Signing In/Out of the App - Ben Hatch**

- Actor/User: A family member responsible for taking care of an elderly relative (Mom) that is using the CareCoord app

- User Story: As a user of CareCoord, I want to be able to log in and out of the application so that I can protect my privacy/sensitive caretaking information.
- Course of Events:
  1. Open the CareCoord app on my smartphone
  2. If I'm not logged in already, the "Login" page should be the only viewable page on the app
  3. Login to CareCoord conveniently using Google OAuth (aka "Sign in with Google")
  4. Once I'm done using the app, navigate to the "Account" page and click "Sign Out" button
  5. I'm now signed out of my account and automatically redirected to the "Login" page
  6. See a summary of who is visiting Mom that day and what tasks need to be completed.
- Exceptions/Alternates: none
- Related UI: Figure 2a

#### **Use Case 8: Edit profile - Seng Rith**

- Actor/User: Person who uses CareCoord to take care of their beloved elders
- User Story: As a good caretaker that works cooperatively with other caretakers, I want people to see what my appearance is like. I also want other caretakers who have mutual caretaking groups to be able to see my information like my name, pronoun, DOB(optionally) and especially contact information(email or phone number)
- Course of Events:
  1. Navigate to the user profile page
  2. Upload a profile photo
  3. Add/edit contact information
  4. Edit username if needed
  5. Add or edit pronoun if needed
  6. Save the changes
- Exceptions/Alternates: none
- Related UI: Figure 5b

#### **Use Case 8a: Change Notification Settings Preferences - Seng Rith**

- Actor/User: As the eldest sibling in my family, I am responsible for taking care of my mom's parents AND dad's parents. Since I am in two different care providing groups, I would like to set available time to take care of my grandparents. I'm available on Monday to take care of my dad's parents who are located in downtown Salt Lake City. On Friday, I usually drive to Ogden

to visit my cousins who are in the same town as my other grandparents. So, I want to set Monday available to one group and Friday to another group.

- Course of Events:
  1. Navigate to Settings and then Notifications
  2. Select reminder for assigned duty
  3. Choose how many days or hours before the duty I want to be notified
  4. Save changes
- Exceptions/Alternates:
  - Alternative: The user can turn off the notification from their device settings if they don't want to get any notification. They can also set their own reminder if they want to.
- Related UI: Figure 8b

#### **Use Case 9a: Edit/Set Availability for care duty - Seng Rith**

- Actor/User: Person who uses CareCoord to take care of their beloved elders
- User Story: As a responsible caretaker with a busy schedule, I want the application to notify me about my care duty so I can be ready for it.
- Course of Events:
  1. On the Home Page, Choose the group I want to set the availability for.
  2. In the group, I navigate to schedule and change availability in the schedule settings.
  3. Then, I save changes.
  4. Lastly, proceed to do the same thing with another group.
- Exceptions/Alternates: none
- Related UI: Figure 8a

#### **Use Case 9b: Set out of town period - Seng Rith**

- Actor/User: Person who is responsible of caretaking duty
- User Story: As Fall break comes, I want to go on vacation for a week. I have already asked my boss at work for the time off. But, I still need to let everyone in my caretaking groups know that I will be out of town from Monday to Saturday. So, someone else needs to take care of my duty while I'm out of town.
- Course of Events:
  1. Navigate to the care taking groups on CareCoord and check which groups I'm supposed to be on duty during my vacation plan.
  2. When I see my time conflict on the schedule of a caretaking group, I set an "out of town" time and the application notifies everyone that I'm going to be "out of town".

3. I then also check who else is available on my duty day and reach out to them to see if they can take care of it.
- Exceptions/Alternates: none
  - Related UI: Figure 5a

#### **Use Case 9c: Set “on call” caretaker - Seng Rith**

- Actor/User: Person who is responsible of caretaking duty
- User Story: Having many people in the same care providing group can be a complicated thing. Sometimes, we want to check up what mom is doing with the person who’s taking care of mom. So, it is nice to be able to set who is on duty and have the application automatically set who is the on-call person.
- Course of Events:
  1. Group owner or admin assigns people to which day they will be on duty
  2. CareCoord uses this information to set the current “on call” caretaker and update it every day.
- Exceptions/Alternates:
  - Alternate: There’s no one on duty during the time.
- Related UI: Figure 4a

#### **Use Case 10: Look at past visits info - Seng Rith**

- Actor/User: Person who is responsible of caretaking duty
- User Story: As I arrived at my grandparents house on my care-duty date, I noticed the plants on the front porch were drying out and dying. I’m suspecting that it hasn’t been watered last week. So, I’m going to find out who was on duty last week and didn’t water the plants.
- Course of Events:
  1. Navigating to the history of the schedule, I found out that Som was on care duty last week.
  2. Looking at the check lists, I didn’t see anything marked as done. I had to reach out to Som and ask him what happened.
  3. I texted Som through and asked him if he forgot to do the checklists and water the plants. Som admitted that he forgot to water the plants. He explained to me that grandad asked him to work on the garden when he was at the house. It was a tiring day for him so he forgot about the plants.
- Exceptions/Alternates:
  - Alternate: There’s no one on duty during the time.
- Related UI: Figure 1a

**Use Case 11a: Creating notes - Brynnli Borrowman**

- Actor/User: Caretaker
- User Story: As a user of CareCoord, I want to be able to store information that is important to my caretaking in the CareCoord app.
- Course of Events:
  1. Open the CareCoord app on my phone
  2. Navigate to the “Notes” section
  3. Click on the button that looks like a “+”
  4. Enter a title
  5. Enter the information I want to save
- Exceptions/Alternates: none
- Related UI: Figures 7a and 7b

**Use Case 11b: Viewing notes - Brynnli Borrowman**

- Actor/User: Caretaker
- User Story: As a user of CareCoord, I want to be able to view stored information that is important to my caretaking in the CareCoord app.
- Course of Events:
  1. Open the CareCoord app on my phone
  2. Navigate to the “Notes” section
  3. There is a list of previously created notes, and I can see their title and a sample of the information they contain
  4. When I click on one, it enters into the page for that note, displaying the title and information
- Exceptions/Alternates: none
- Related UI: Figures 7a and 7b

**Use Case 11c: Searching notes - Brynnli Borrowman**

- Actor/User: Caretaker
- User Story: As a user of CareCoord, I want to be able to search for stored information that is important to my caretaking.
- Course of Events:
  1. Open the CareCoord app on my phone
  2. Navigate to the “Notes” section
  3. Click on the “search” button
  4. Enter a keyword
  5. Look through the list of results until I see what I need

- Exceptions/Alternates: none
- Related UI: Figures 7a and 7b

### **Use Case 12: Creating a CareCoord Account - Brynnli Borrowman**

- Actor/User: Caretaker
- User Story: As a caretaker, I want to create an account on CareCoord
- Course of Events:
  1. Download the CareCoord app from the app store
  2. Open the CareCoord app
  3. Register for an account with Google OAuth
  4. Enter required profile information
- Exceptions/Alternates: none
- Related UI: Figure 2a

### **Use Case 13a: Creating a CareCoord Group - Brynnli Borrowman**

- Actor/User: Caretaker
- User Story: As a caretaker, I want to create a Group on CareCoord
- Course of Events:
  1. Open the CareCoord app
  2. Navigate to the “Group” page
  3. Click on the “Create CareCoord Group” link
  4. Enter a group name
  5. Click the “Create” button
- Exceptions/Alternates: none
- Related UI: Figures 5a, 4a and 4b

### **Use Case 13b: Inviting Others to Your CareCoord Group - Brynnli Borrowman**

- Actor/User: Caretaker
- User Story: As a caretaker, I want to invite other people to my group on CareCoord
- Course of Events:
  1. Open the CareCoord app
  2. Navigate to the “Group” page
  3. Click on the “Invite others to join your group” link
  4. Enter the person’s phone number and/or email
  5. Click the “Invite” button
- Exceptions/Alternates: none
- Related UI: Figures 5a, 4a and 4b



**Use Case 13c: Joining a CareCoord Account - Brynnli Borrowman**

- Actor/User: Caretaker
- User Story: As a caretaker, I want to join a Group for the person I take care of on CareCoord
- Course of Events:
  1. Open the message or email containing the link your friend sent you
  2. Click on the link
  3. The CareCoord app will open
  4. A pop-up will open, asking you to confirm you would like to join the group
  5. Click “Yes”
- Exceptions/Alternates: none
- Related UI: Figures 5a, 4a and 4b

**Use Case 14: Viewing Contact Information for Group Members - Brynnli Borrowman**

- Actor/User: Caretaker
- User Story: As a caretaker and CareCoord user, I want to find contact information for the members of my group.
- Course of Events:
  1. Open the CareCoord app
  2. Navigate to the “Group” page
  3. Under the “Group Members” section, there is a list of the group members
  4. Click on the name of the group member you want to view
  5. Their profile will pull up, which contains their contact information.
- Exceptions/Alternates: none
- Related UI: Image 5a and 4a

## Appendix C: Revisions

### **Table of Contents**

No changes.

### **Executive Summary**

No changes.

### **Background and Technical Requirements**

No changes.

### **System Architecture**

No changes.

### **Personnel**

No changes.

### **System Features**

No changes.

### **Software Engineering Tools and Techniques**

No changes.

### **Timeline**

No changes.

### **Appendix A: UI Sketches**

No changes.

### **Appendix B: Use Cases**

No changes.