<Compass>

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Christina Darstbanian 40097340 (Team Leader)	Chr728	15
Imran Ahmed 40172931	imran1289-ah	17
Xavier Morgan-Tracy 40129775	XavierKMT	3
Jiayi Chen 40110997	JIAYI615	5
Divleen Kaur Ahluwalia 40116121	Divleen12	15
Jan Mikhail Alexei Ong 40154849	janong24	10
Julien Phan 40133814	MrMelon1232	3
Adir Ben-David 40190551	beezzyy	7
Jonathan Abitbol 40190550	yoniabitbol	5
Reuven Ostrofsky 40188881	Reuven1203	5

Project summary

Compass is a medical wellness app that targets specifically people above 40 years old who might be interested in having some type of assistance to keep their healthy habits and lifestyles. Compass offers features of managing medical reminders, booking appointments,tracking user's medications and treatments all in one consolidated application. In addition, having features of medical journals such as diabetic journals allows some patients to easily note their daily doses and treatment details making them able to follow the history of their treatments and use it as a reference for themselves or to show to their medical professional. Additionally,with the speed dial fast option to contact relatives during some emergency situations patients would be able to contact their relatives in a faster and easier way. Not to forget to mention that the app also solves issues related to translation while speaking to a doctor or having difficulties moving to the medical center by having assistance and translation requests that can be made. With many features compass aims for users to be healthier and function hassle free.

Risk

The most pertinent risk to Compass is that of user privacy. Failure to ensure privacy can have multiple negative consequences like unauthorized data access, identity theft and data misuse. It can also lead to a loss of trust between the users and Compass. In order to tackle this risk, Compass will, first of all, be transparent with the users about how their data is being used and who has access to the data. Secondly, the data will be stored directly in Compass's database and will not be shared with any third party applications. Compass will collect only the minimum amount of data necessary. For example, while registering, sensitive information like health card details are not asked. Users will have the ability to access, correct and/or delete their data. Further, since privacy cannot be ensured without security, Compass uses Firebase authentication for authentication services. This service has several layers of security built-in. For example, it automatically hashes and salts passwords for protection, and uses https for data transmission. This provides an immeasurably more secure authentication mechanism than one built from scratch. The following are the links to the login and registration features that use Firebase Auth.
Login Feature
Registration Feature

Legal and Ethical issues

- **Privacy and security of user data**: Users who register or use journal features willingly supply all the information that is contained in the system.To keep commitment to upholding user privacy, terms and conditions must be clearly presented to users and make it clear how their data will be handled and processed. Personal health information about users must be treated with care and won't be disclosed without their permission.

- **User consent**: Users must be informed explicitly of all features that use personal data and also of any potential risks before providing it.
- **Notifications**: All notifications must respect user privacy. All private information should not be displayed directly through notifications. Consent must be given by the user regarding any reminders for medication or appointments.
- **PWA standards**: The web app must be accessible and functionable across all smart devices that can access the browser and has the necessary specs. Also, the web app must comply with all web standards for a PWA.
- **Data deletion/retention**: Users must be advised of all policies regarding how long data is stored and which data can be requested for deletion. Procedures must be in place in case of unintentional data loss such as data backup protocols.
- **Ethical equity**: PWA must ensure that all groups of users may not be discriminated against or be disadvantaged. Users must be addressed of any potential use or issues of misinformation or misunderstandings of any medical terms or advice.

Economic

The goal of this project is to ensure patients are able to follow their health status by having some medical journals that help them track their daily health lifestyles, habits, manage medications and treatments.The measurable impact by the user would be having a consolidated app that offers many services all in one application. Having an easier and more efficient way to manage their health would make users more comfortable in their daily lifestyles. Current features are all offered at no cost. The ultimate goal would be in future to have some features that are not free under some premium subscription plan. Users would be able to enroll to a premium plan and have access to some additional features. Progressively developers would be able to make some income with the increase of numbers of users and continuously working on adding/improving features to the application.

Contractor Estimate

It was agreed and confirmed that the stakeholder would not own the project and rather it would be an open source project under AGPL (Affero General Public License) license. The stakeholder agreed to volunteer for the hours worked for the project including the meetings, discussions, questions and the signoff of each story worked by the team. The stakeholder is not paid throughout the whole project.

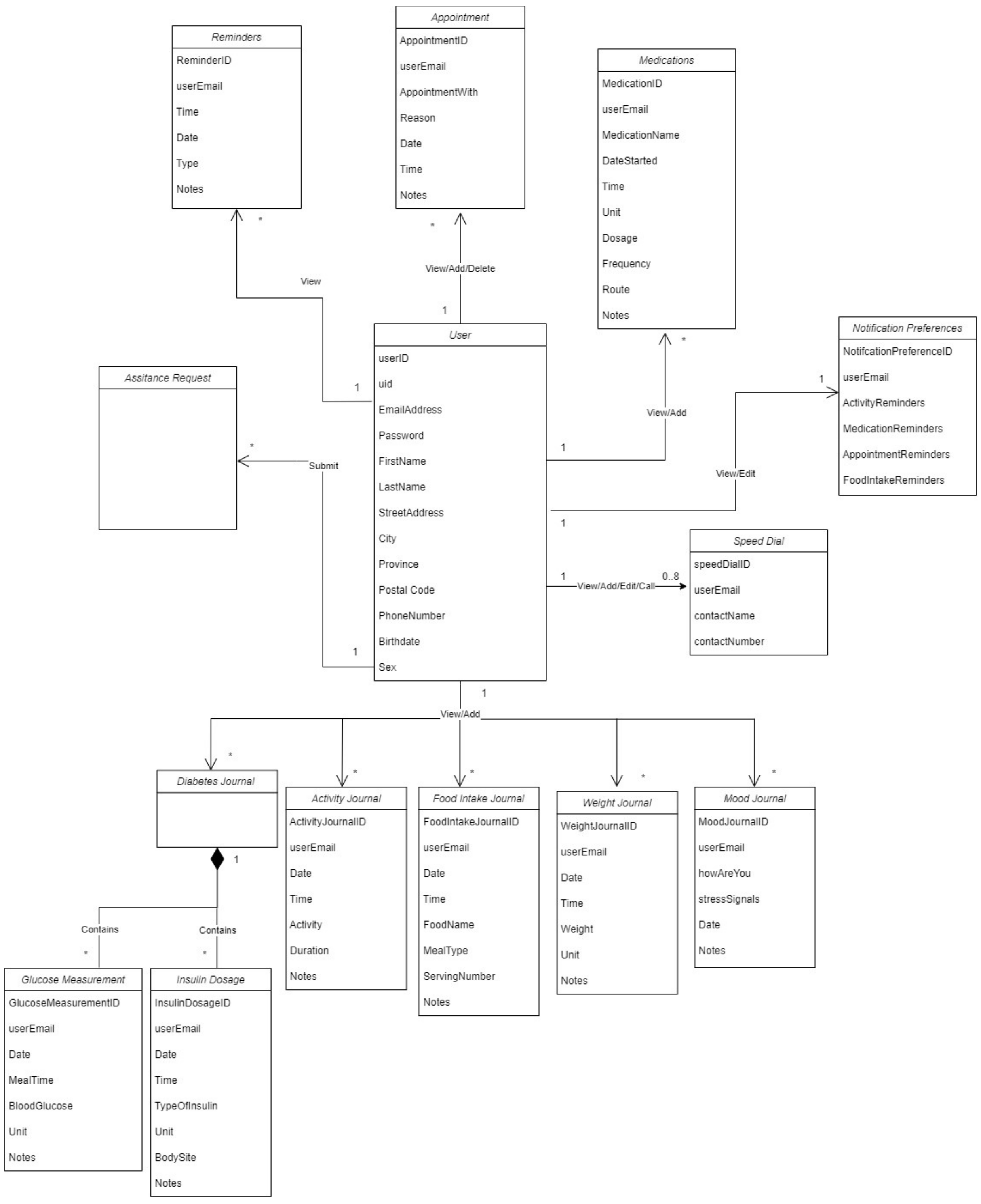
Velocity

Iteration 1 (8 stories, 31 points)

The goal of the first iteration was to set up the main pages that will be included in the application. This includes a splash page, main menu, login, logout, sign up, settings, and forgot password pages. Furthermore, the CI/CD pipeline was set up as well as tests in both the front end and the back end.

Overall Arch and Class diagram

The Domain Model of Compass



Infrastructure

Front End

1. Next.js framework:
    a. Server-side rendering
    b. Routing
    c. File system routing by having components automatically become routes when named page.tsx
2. Tailwind CSS framework:
    a. Utility first approach where classes directly apply styles to the HTML elements
    b. A responsive design for our progressive web app

Back End

1. Node JS:
    a. A runtime environment to run our server-side code
    b. Data synchronization and event-driven, non-blocking flow
2. Express JS:
    a. Middleware allows express to process incoming HTTP requests, as well as routing, and data validation
3. Firebase Authentication:
    a. Authentication as a service is provided which simplifies user management
4. Postgres SQL:
    a. Relational database
    b. Data models to help store our application's data and organize it through tables and relationships
    c. Uses SQL language to perform operations

DevOps and Testing

1. Docker:
    a. Offers containerization that allows our application to be packaged into a container, ensuring consistency between development, testing and production environments
2. Jest
    a. Testing framework for unit tests and integration tests
    b. Allows mocks and spies on function calls
3. Supertest
    a. HTTP testing: tests HTTP requests and API's in Node.js application while also allowing simulation of requests and validation of responses
    b. Integration with Jest

Name Conventions

For our project, we use a camelcase naming convention with a pascal case convention for component names in our application. Function names also reflect actions.

Code

Key files: top **5** most important files (full path). We will also be randomly checking the code quality of files. Please let us know if there are parts of the system that are stubs or are a prototype so we grade these accordingly.

File path with clickable GitHub link	Purpose
Docker-compose-dev.yml	This file is a Docker Compose configuration file for setting up development containers for a PostgreSQL database, a server, and a client application, along with their dependencies, volumes, and environment variables.
server/index.ts	This file is an Express.js application setup file that configures middleware, routes, and database synchronization, including logging and handling environment variables, and exports the configured Express app.
server/config/config.js	This file is a Sequelize configuration file that specifies database connection settings for different environments (development, test, production) and loads environment variables from a .env file to populate these settings dynamically.
client/app/layout.tsx	This file is a React component that defines the layout structure for the root of a web application. It includes context providers for authentication and user data, as well as global CSS styles. The children prop allows rendering of the main content within this layout component.
client/app/config/firebase.tsx	This file sets up and configures Firebase for a web application, including Firestore and Authentication, using environment variables for the configuration, and exports these Firebase services for use in the application.

Testing and Continuous Integration

Test File path with clickable GitHub link	What is it testing
server/test/userController.test.ts	This test contains all the unit tests for the backend user API.
client/app/settings/settings.test.js	This test contains the unit test for frontend settings
client/app/login/login.test.js	This test contains the unit test for frontend logging in
server/test/index.test.ts	This test contains the unit test for the default backend router.
client/app/register/register.test.js	This test contains the unit test for the frontend register

The team chose nextJs and react for the frontend, and express, sequelize (a Node.js_based ORM to query the database for the backend), PostgreSQL, and firebase authentication for the backend. Docker was chosen as the container.

- For the backend: The backend team will mainly focus on writing the unit test for each API written, which means that the functions inside all controller files and the routers of express will be tested. The team decided to use jest to mock the database connection and the database querying results that are returned by using sequelize functions. Supertest was selected to mock the frontend behavior of sending requests to the backend, which will help test the router.
- For frontend: The frontend team decided to focus on the UI testing to ensure all the user workflows are well-designed and work normally. The unit testing will test if the content of the component or page is rendered correctly on the screen. All the components and the pages will have their own separate UI tests.

Both the frontend team and the backend team will try to reach a test coverage that is as high as possible. All the tests will run automatically when a new feature is added to make sure the new code won't break old features.

In the case that some methods/classes are missing unit tests, to make sure those methods/classes are functioning, the team will run all the acceptance tests that involve those methods/classes. The team will also assign members to check the reason why their unit tests are not written when the code is committed and try to add unit tests for them.

The link to CI:
https://github.com/janong24/Compass/actions
The CI file:
https://github.com/janong24/Compass/tree/main/.github/workflows

The team chose Github Action to build the CI/CD pipeline. The integration and version control are automated by the control system git. Developers make changes on different branches and commit the changes to git to ensure that all the changes are trackable. Every time when the code is pushed to remote repositories, the automated build and testing are triggered by our continuous integration environment.

The process goes in the following order: setting up the running environments, building the backend, building the frontend, running the test, and cleaning the credential file. Our continuous integration environment provides rapid feedback to the developers. Developers will be informed immediately if their code fails in any part of the CI process. The mechanism of the version control system lets the developers be able to roll back to the previous version to eliminate critical problems.