Project Report

Group WED-16.30-6

VERDOUW, Midori (s3575912)
TJIONG, Julian (s3786866)
SONG, Jason (s3744335)

PHAM, Van (s3788106) – missing/left group
CHEONG, Hon Khuin Jonathan (s3642842) - missing/left group

Vision

Our product enables businesses to have a bigger impact and our customers to enjoy the freedom of booking whoever they want, whenever they want.

The product allows businesses to be placed on an online platform on top of their existing online or offline presence. We allow businesses of all types, ranging from retail to the consulting and construction industries. Businesses are able to easily select who they want to work in a given time slot, providing better flexibility for worker time slot allocation as well as better tracking of their timesheets.

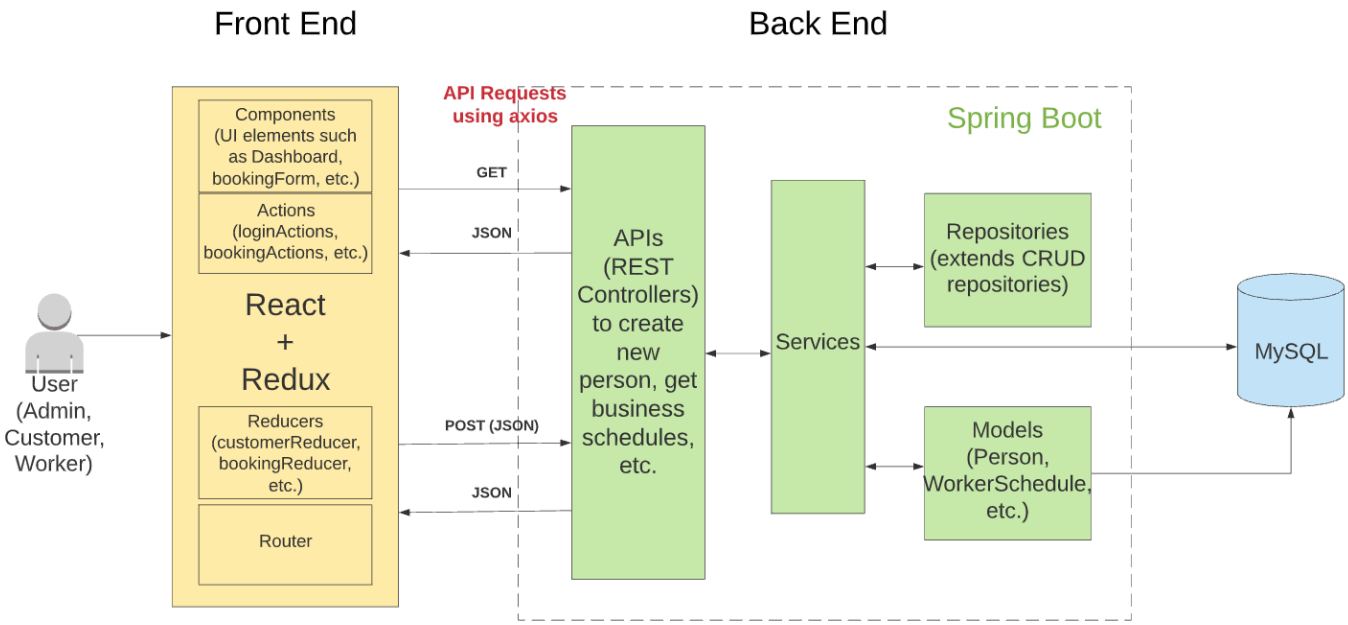
As a customer, you are able to not only book a time for the service you want, but also choose the person from the business who is working at that time. This makes the experience of rebooking the same person who cut your hair, did your dental checkup or provided assistance at your favourite retail store.

Our platform also empowers workers to easily manage their schedule, and inform the business owner of any changes required for sick leave, annual leave etc. Workers are able to input their availability any time and the business owner will see that reflected live on their end and adjust the worker allocation for customers.

This platform provides exposure to businesses, customers a hassle free online experience and workers to better manage their time.

System Architecture/Design

For the three tiers (frontend, backend, datastore) we have used React, Spring Boot and MySQL respectively. The React application uses Redux for the state management, which involves elements such as actions, reducers, store, and the view (components and containers). The front end sends API requests to the back end for getting all business schedules, for adding new person (i.e., worker and customer), for authenticating user inputs for login, etc. The back end provides such APIs as the Spring Boot REST controllers, which are supported by other layers of the back-end application, namely, services, repositories and models. Our application has models such as Person, WorkerSchedule, and Business, each of which correspond to a table in the database.



Refactoring

The majority of the refactoring occurred in the way we set up our database on the backend. Throughout the semester we used an H2 database with a single “Person” table and “Worker Schedule” table.

Ideally the person table would be broken down, and it would become an abstract class in Java where we could create various user types.

This would look something like:

- Abstract Person
    - Worker
    - Business Owner
    - Customer

As a result of the people being stored under one collective table, this had knock-on effects for our worker schedule. Since we were required to provide the user type in our Person table (w for worker for example), our schedule was constrained towards only for workers.

In reality, business owners may look at expanding the requirements of the scheduling aspect of the program to include features such as:

- Meetings with other businesses (B2B)
- Scheduling when stock comes in and out
- Tracking projects

A single schedule table could be expanded to include these features similar to our generic “person”. An example of this could be:

- Abstract Schedule
    - Business internal schedule
    - Worker schedule
    - Calendar/meetings schedule

Gitflow Organisation

We have used the following types of branches:

- master
- deployment
- feature (e.g., feature/US#1_customer)
- release (e.g., release/milestone2)

Each feature branch is used to implement one user story and is active only during implementation. It is based from the development and merged into development upon the completion of the implementation of the feature. A special feature branch named “feature/docker_configuration” exists only for preserving the docker and database configuration needed for demo and deployment. The release branch is created based from the development when all the user stories for the milestone finished and their branches are merged.

Although we have had a short period of time (approximately 7 days) at which no commit was made, overall we have committed regularly throughout our development since Sprint 1. At the time of writing we have 247 commits in total in the development branch.

Scrum Process

Our scrum process followed the traditional scrum framework based off the Scrum Guide. It included multiple sprint artifacts, as well as maintaining our product backlog and multiple standups every week.

We began by assigning roles to each member of the group. Please note that we were all on the development team and we were also 2 members short.

- Scrum Master: Midori
- Development Team: Midori, Jason, Julian
- Product Owner: Jason

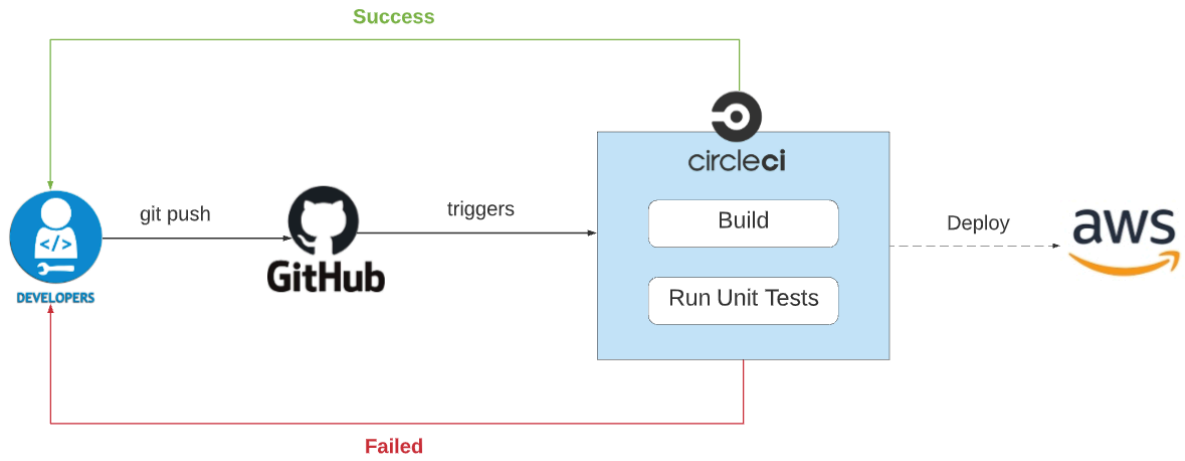
Following that, each milestone consisted of approximately 1 – 2 sprints which were broken down into:

- Sprint planning: We would plan what we were going to do throughout the sprint, prioritise the backlog and discuss each item within the development team
- Daily standups: Each week we would hold 3 standups to ensure everyone was on the same page. They were approximately 30mins in duration as we were only meeting 3 times a week instead of daily. We would ask the following questions each standup:
    - What did I complete since our last standup?
    - What will I work on until the next standup?
    - Am I blocked by anything?
- Sprint review (during tutelab sessions): At the end of every sprint/milestone, we would showcase our progression to our tutor. We were able to celebrate our accomplishments and demonstrate our work so far live to the tutor. This was extremely useful to confirm what we had done to date, and get immediate feedback from the stakeholders (Mohamad)
- Retrospective: The retro enabled us to determine what was working well and what was not. The consensus was that we were able to communicate effectively despite 2 of the members not actively contributing. However, we were not proactive in reaching out to those members earlier to provide support for them, and as a result inform our tutor. We developed our action plan every week, and this drove us towards agile development.

CI/CD Pipeline

Our CI/CD pipeline is illustrated as follows. We used CircleCI for continuous integration; during implementation, any change in our source code pushed to our GitHub repository (i.e., to development and feature branches) triggers build and unit tests in the CI system. CircleCI then provides the result of the build such that the developers are notified whether or not any unit test has failed. In this way it is ensured that the system works correctly whatever pieces of implementation is added or changed. In our case, however, this applies only to the build and unit tests of the front end.

In addition, the development steps should be extended such that continuous deployment is also possible. Our initial plan was to perform automated deployment to AWS through CircleCI, however, it could not be achieved for this milestone and thus the project has been manually deployed.



Testing Documentation

Please refer to the documentation provided for our testing on our GitHub repository.

This is included under “docs/Test Reports”.