Project Report

Majorproject-3-thurs-10-30-2

CAVE, Luca (s3787946)

GAO, Richard (s3784324)

HODGEN, Chris (s3031209)

MCKEOWN, Ryan (s3718888)

1. Our Vision
2. Basic System Architecture Diagram
3. Refactoring Description
   1. Worker Dash Class to Function
4. Gitflow Overview
5. Scrum Process Description
6. Deployment Pipeline Diagram
7. Acceptance Test Cases

## **Our Vision**

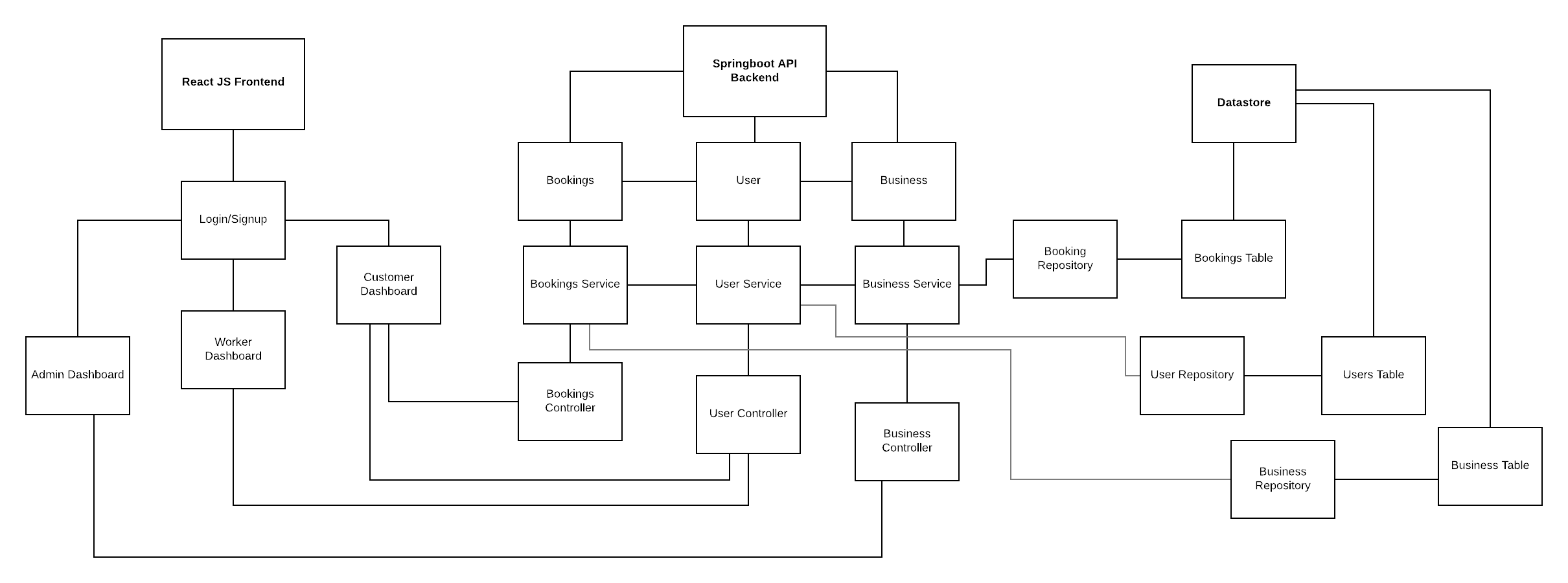
To create a flexible new booking system; providing a frictionless experience for customers to make appointments with their favourite businesses.

## **Our Product**

AGME is a highly flexible, extensible and lightweight bookings API, paired with a clean React JS web application. It simplifies the process of creating a business, adding employees and scheduling bookings so that businesses and customers can focus on the things that matter.

Future product enhancements will include integration of geographic data to recommend local businesses to users as well as the creation of a business rating system.

**(this space intentionally left blank)**



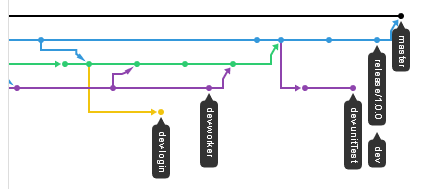
## **Refactoring**

During sprints 3 and 4, a great deal of refactoring took place in the front end. This was predominantly concerned with the conversion of various components from classes to the more modern current standard of functions. This also necessitated a great number of changes in how information is communicated between components and the way API calls and state updates are handled.

In most cases this involved removing constructors, using hooks and using const variables instead of this.state and this.props. As a result of this refactoring the code became much easier to read, extend and maintain. It also solved an issue in one component which was causing a massive memory leak.

## **Gitflow**

We adhered to Atalssian’s Gitflow processes throughout the project. In doing so we avoided any massive merge conflicts and were able to keep code highly segmented while under development. The use of pull requests which were confirmed by at least one other team member ensured that new code was of good quality and did not cause any feature regression. During development commits occurred at 20-60 minute intervals, generally after a feature/modification has been implemented without compilation errors.



## **Scrum Processes**

Luca Cave served as the Scrum Master for the duration of the project. On average, the team got together for official meetings twice a week, although in some situations extra meetings were scheduled to go over requirements and blockers or to conduct task estimation.

In addition to official meetings, some or all members of the team regularly met to conduct peer programming sessions. These meetings were particularly useful when a team member encountered issues when implementing one of their assigned tasks.

Overall, our use of Scrum Processes was effective and ensured the team stayed on task and met our required deadlines.

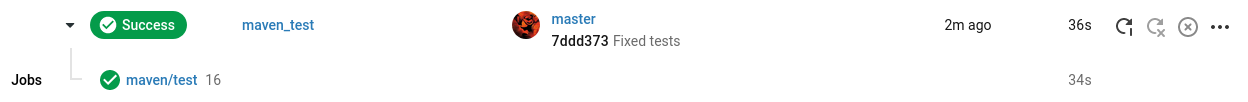
## **Deployment Pipeline**

Due to issues with the CircleCI and AWS education accounts, our deployment pipeline was less than ideal. While we successfully integrated CircleCI (as shown by figure 1 below), the issue with credits meant that we were unable to actually automate deployment with it.

Despite this setback, each aspect of our development pipeline worked in isolation. Our docker containers and composition worked very effectively and made it easy for team members to get development up and running without installing native dependencies. Deployment of the Spring application on AWS using the ElasticBeanstalk service was generally smooth after a few initial hiccups.

Unfortunately we encountered a number of issues when deploying the React application on AWS. After a great deal of time investment we were able to get the application to compile and run (shown in the server logs) but it could not communicate with nginx. We suspect a port issue.

As a backup we deployed the front end application to Heroku, which proved to be extremely straightforward and quick.

Figure 1. CircleCI tests passing successfully

## **Acceptance Testing**

For details of Acceptance Testing see the final acceptance testing document.

Figure 2. Unit tests passing in IntelliJ