**Project Report**

**AGME Booking System**

**Group 3**

**Kaushal Gawri, Ryan Kelly, Muhammad Ali Tariq, Prabhav Mohit Mehra, Mark Johnson**

**Vision Statement**

The AGME Booking System allows for an easy-to-use system for three tiers of user: administrators (business owners), workers, and customers. The system allows for the registration of and signing into these three account types through intuitive interfaces. Business owners can register their company and administer data relevant to their company.

The system is broken into three core components: administrator, worker, and customer.

**Administrator (Business Owner)**

Business owners can register an account with the booking system, allowing them to then sign in and utilize the full range of administrative tools provided. Administrators can create worker accounts, view all past and future bookings, delete bookings from the system, and view their profile.

**Worker**

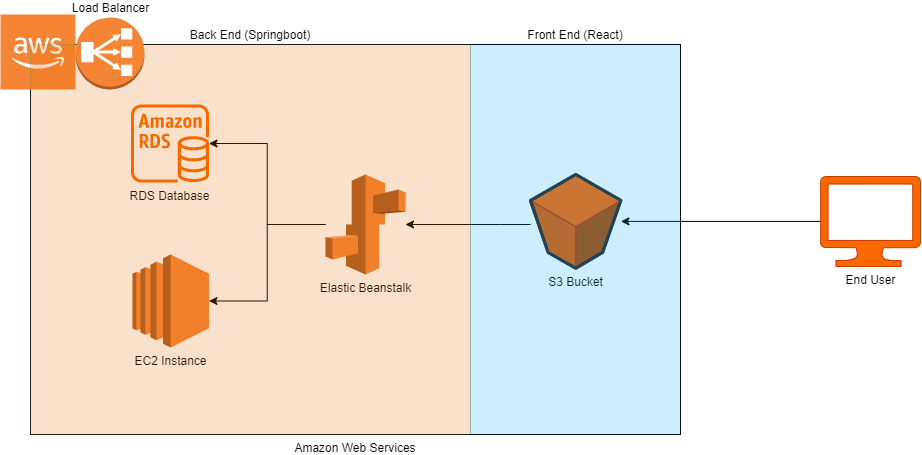
Workers can sign in with an account registered by them by an administrator. Workers have limited functionality compared to administrators: they can view their upcoming and past shifts, view their profile, and edit their profile.

**Customer**

Customers can register an account and use that account to sign into the system. Customers can make a booking with companies, which is stored in the database. Customers are also able to view and edit their profile, as well as view their past and future bookings.

**Architecture Design**

The AGME Booking System makes of Amazons’ Elastic Beanstalk for backend and frontend deployment. The front end files are stored within an S3 bucket, which communicates with Elastic Beanstalk and, in turn, the EC2 instance hosting the back end, and the RDS database containing the application data.



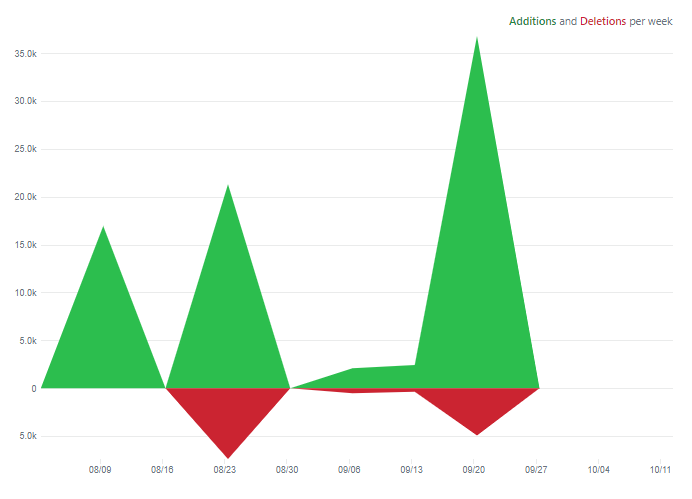
**Refactoring**

The backend team refactored some of the code by assigning the database password to a variable, reducing the number of times the person running the code needs to enter it through the application.

**GitFlow Organisation**

The team utilized GitFlow for the development of the application. The **master** branch always contains the most recent production-ready code, as agreed upon by at least one member of the team via approval of a pull request.

New features are pushed to their own branches with names reflecting the new code. The developer then creates a pull request to the master branch, which is reviewed by another, or multiple members of the group, before being approved and merged into the master branch, ready for deployment.

The team’s code frequency graph outlines the additions and deletions per week:

**Scrum Process**

The team’s Scrum Master was Ryan Kelly. Each sprint (except for the final sprint) lasted two weeks, with the final sprint lasting three. The team met at least once weekly to conduct a Sprint Retro meeting, where the team outlined what went well, what could have gone better, what surprised them and any lessons learned in the previous sprint. Afterward, the team conducted a Sprint Planning meeting, where the team was instructed by the Product Owner (Ujjwal Batra) which tasks were to be completed for the upcoming sprint.

While the team was unable to perform daily standups due to external commitments, the team substituted the process for regularly Discord communications to address any issues that arose. On numerous occasions, group members working on solving a problem met independently of the rest of the group, allowing them to solve issues without the need for a scheduled meeting.

**Deployment Pipeline**

