**Project report**

MON-1530-6

**Vision**

Our vision for this project was creating an application that is useful for businesses or originations who require a booking service to have an intuitive, simple and easy to use service readily available to them.

It only requires the organisation to change the name of the product and change the available services and they have a fully-fledged service up and running instantly.

This use case for any origination is invaluable as they can easily, quickly and cheaply keep tracking of all bookings and keep track of all employees and what they’re up to, who they’re servicing, when they’re servicing them and how they’re servicing them.

Managing the appointments and employees is made much easier when using this application for any organisation which would desire to use it.

The final product speaks for itself in value as its extremely user friendly and extremely useful for managing all a manger could want to manage.

**Basic System Architecture**

A picture containing diagram

Description automatically generated

**Refactoring Report**

SpringBoot:

A lot of refactoring had to be gone through to make the code simpler and easily understandable. One of it was that we had a method in the controller class that was used locally by other controllers to get the details of a booking using the id as follows:

A picture containing graphical user interface

Description automatically generated

This method was used to get the details of all the booking using the booking ID. Initially this was used as a separate method to invoke the database, then we found it much more simpler to perform the function body implicitly from the calling method, to reduce the number of controllers and make it simpler in understanding the functionality and use of each controller. The above method was removed, and the following was used instead:

Text

Description automatically generated

ReactJS:

The following code is used in some of the JS classes wo assign the state variable with the input data. This function is called onClick for a field to assign the values using the field id and the field value.

Text

Description automatically generated

This is efficient as we can reduce the number of functions to do the same task just be defining the id value. But for some functionalities, where the fields have to appear or change visibility upon click or input change, this does not seem to efficient. So, we had to change to the following where requiredText

Description automatically generated

Text

Description automatically generated

From the above code, the line:



changes the visibility of each fields which is done when the onClick is called for the previous field input. This makes it easier to fetch the data from the database based on the input fields given stepby-step rather than reading all the data from the database at once. The above changes were implemented to as many onClick actions where the data had to be fetched and displayed each time for a user input from the database.

Of course there were many other functions that were implemented and called implicitly with other functions to reduce the number of functions and maximise the simplicity in both frontend and backend codes.

The structure we have followed is very simple for spring boot, which we believe is very efficient, the frontend talks with the controller through the REST API’s which in turn calls the corresponding service classes to change/access/delete entries in the database which is in the Repository class.

Commenting out the functions by adding relevant information to make it more easily readable and understandable was really important. The work that one teammate does must be commented with relevant information to make the fellow teammates easy to understand the status of work that is done or being progressed to avoid conflicts and redundancy. This makes it easy readable rather than a complex lines and lines of code with no clue on what it is doing.

**GitFlow Organisation**

Contributions to Master:

Chart

Description automatically generated

Commits and additions/deletions from each team member to Master branch:

Graphical user interface

Description automatically generated

Commits per week:

A picture containing table

Description automatically generated

On average the team committed 5 times per week.

**Scrum process**

Ashraf Miari acted as Scrum master

The scrum process was relatively consistent throughout the 12 weeks of working on the project.

At the start of every sprint we also met for a sprint planning meeting and discussed our vision for the sprint and what we needed to get done. We assigned the scrum points and the general tasks and set a meeting for our first meeting of the spring after the planning.

During the meetings we also asked each other questions and helped each other if we were stuck on any part. If one of us was finding a task too difficult or thought that they could be of more help in another area of expertise, then we would agree on a swap or assign one group member to help the team mate on what they needed help with.

Typically, we met 2 times a week and discussed our progress as well as what we wanted to get done in the time between now and the next meeting. During these biweekly meetings we also set the time for the next meeting. This acted as a deadline to all the tasks we discussed about getting done.

At the end of the sprint the team meets one last time for the sprint retro and we discuss and come up with a consensus of how to answer all the key components and questions brought forth by the retrospective.

**Acceptance testing**

Acceptance tests are in the acceptance test docs