Bookeroo

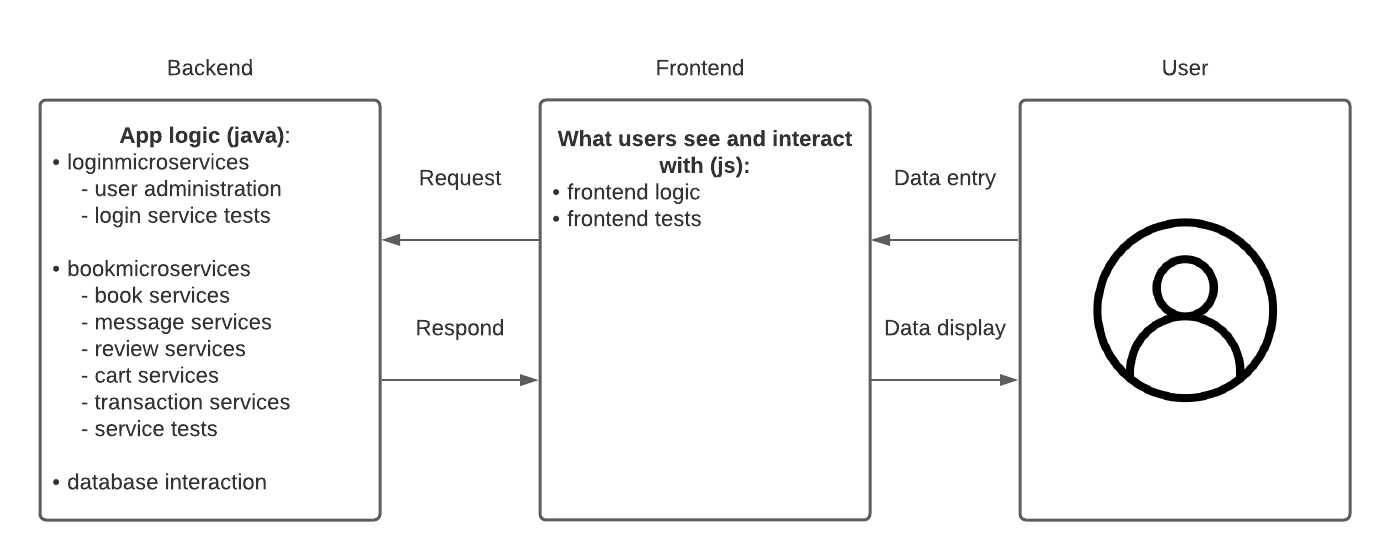
**September 04, 20XX**

# Vision Statement

The bookaroo service will allow users to not only buy their books from safe and secure sources, but also allow for second hand selling in a safe and secure way. Through utilising PayPal’s payment services, card information is stored safely. Additionally, as every business needs to be approved before allowing users to make purchases from them, this allows for the administrative team to monitor who will use the application ensuring there will be no fraudulent services utilising the platform. The combination of these two concepts has meant that users can securely make purchases without worries of authenticity. From the seller’s end, Bookaroo’s services ensure their payments go smoothly, and orders are made and updated as necessary.

Bookaroo allows for users to sell second hand books only when purchased originally from the site. This means that every book, both brand new and used, is an approved safe book and does not have inappropriate/unapproved content.

# Specifications

**System Architecture / Design**

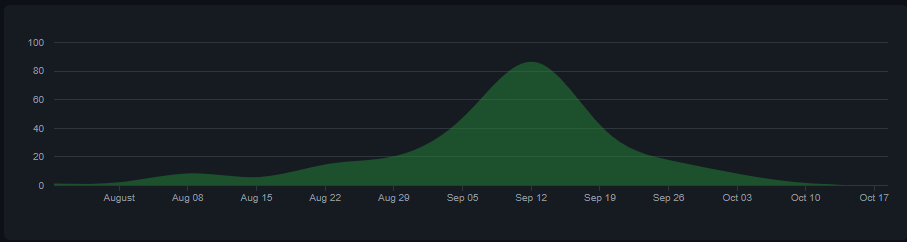
# 

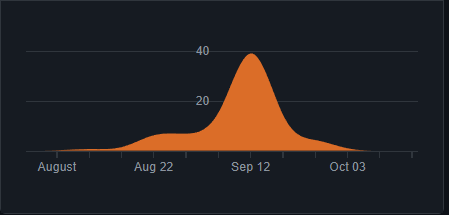
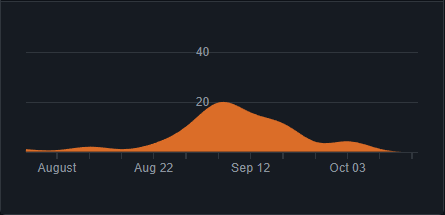
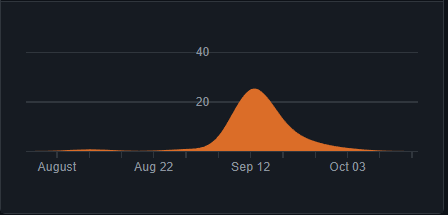
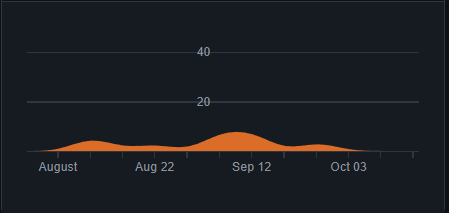
# 

# 

# GitHub flow and statistics

Our projects started off with a Master and Development branch. At the start of each sprint, we create new branches off of the Development branch, and when we are finished coding on the branch, we create a Pull Request to merge into the Development branch. Our feature branches are titled specifically relating to the chosen task the developer is working on.

After the sprint comes to an end, we have a meeting and clean up all of our feature branches, clearing branches that are not needed anymore, and deciding which branches are still being worked on. Finally, when we are satisfied, we make a Pull Requestfrom the Development branch to the Masterbranch. This gets repeated each sprint.  
  
Here are statistics of total commits per contributor for the project:  


* **135** commits Danny Pham   
   (**451** Files Changed, **24356** Lines inserted, **18899** Lines deleted)   
  
* **92** commits Harris Charalambous   
   (**355** Files Changed, **2661** Lines inserted, **3710** Lines deleted)   
  
* **69** commits Clinton Thai  
   (**84** Files Changed, **2420** Lines inserted, **19588** Lines deleted)   
  
* **48** commits Tessa Podbury  
   (**298** Files Changed, **41167** Lines inserted, **421** Lines deleted)   
  
* **10** commits Mazda Shahzadi  
   (**93** Files Changed, **2414** Lines inserted, **39** Lines deleted)   
  

**Please note:** Some files which were meant to belong to .gitignore were pushed to the repository so some statistics may be inaccurate

# Scrum Processes

At the beginning of this project, Tessa was allocated as the Scrum Master. We never officially changed that role, but realistically, Harris took the role of Scrum Master and coordinated a vast majority of the project.

Our Scrum Process followed the Agile model where we were constantly iterating through the different phases of development and planning.

At the beginning of every Sprint we would have a Sprint Planning meeting where we would all discuss what items we wanted to include in the Sprint Backlog, how much weight we believed they held and who was going to be responsible for the different tasks associated with every PBI.

Following that, we also had a once weekly meeting with the Product Owner where we would ask any questions that we had and where he could make requests from us as the development team.

Every week we would have two formal stand up meetings where each team member would explain what they had been working on, what they were going to be working on and if there were any roadblocks that were impeding their progress moving forward. This usually occurred on a Monday night and Friday night depending on team member’s and their commitments.

In between the formal stand up meetings, the team was in contact through Slack and Teams calls where we would help each other out with problems that arose.

It was every team member’s responsibility to update the Sprint Backlog. Tessa was in control of the Jira board.

At the end of every Sprint, the team would have a formal meeting to discuss the Sprint Retro. This was a space where everyone could talk about what they thought was important for the sprint. This included positive feedback, negative feedback and any discussion over what we could improve for our next Sprint.

Prior to every milestone submission, we would have a formal meeting to go over documentation and ensure that everyone had submitted their part for what was required.

# 

# 

# 

# 

# Acceptance Testing and Test Execution

**Acceptance Testing**

Acceptance testing was done using a few set steps. First of all, we would list out the PBI and its appropriate acceptance criterion. For example, PBI 11 is “As a customer, I want to view the availability of a book, So that I can purchase it”. Its acceptance criteria is:

1. Given I am a customer, when I click a book, the availability of the book is shown as: “In stock” or “Out of stock”.

Then, by the acceptance criteria, a tester would follow the exact steps and if the availability of the book is shown properly, then the test passes, and if not it fails. This is all recorded in a document with images that goes through each step of the process.

**Frontend Test Execution**

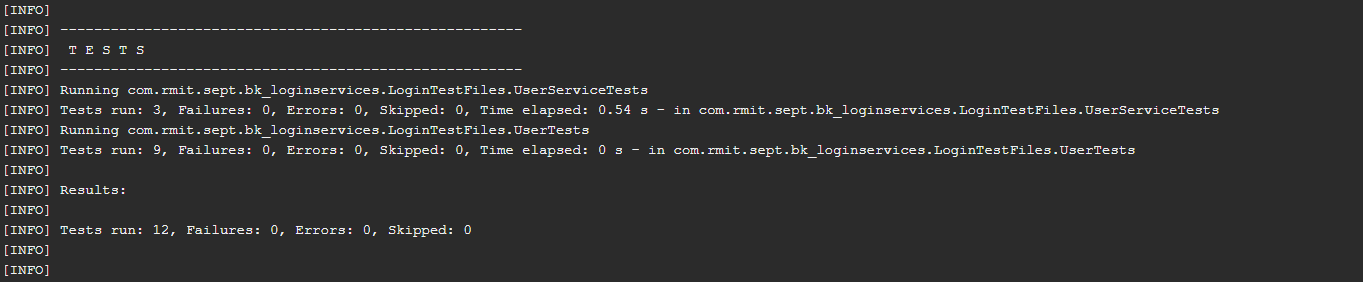
In order to test the application, tests for the frontend were necessary. The frontend was tested using jest as well as enzyme, by creating test files for action and reducer classes. The tests all passed on CircleCI as shown in the image below.



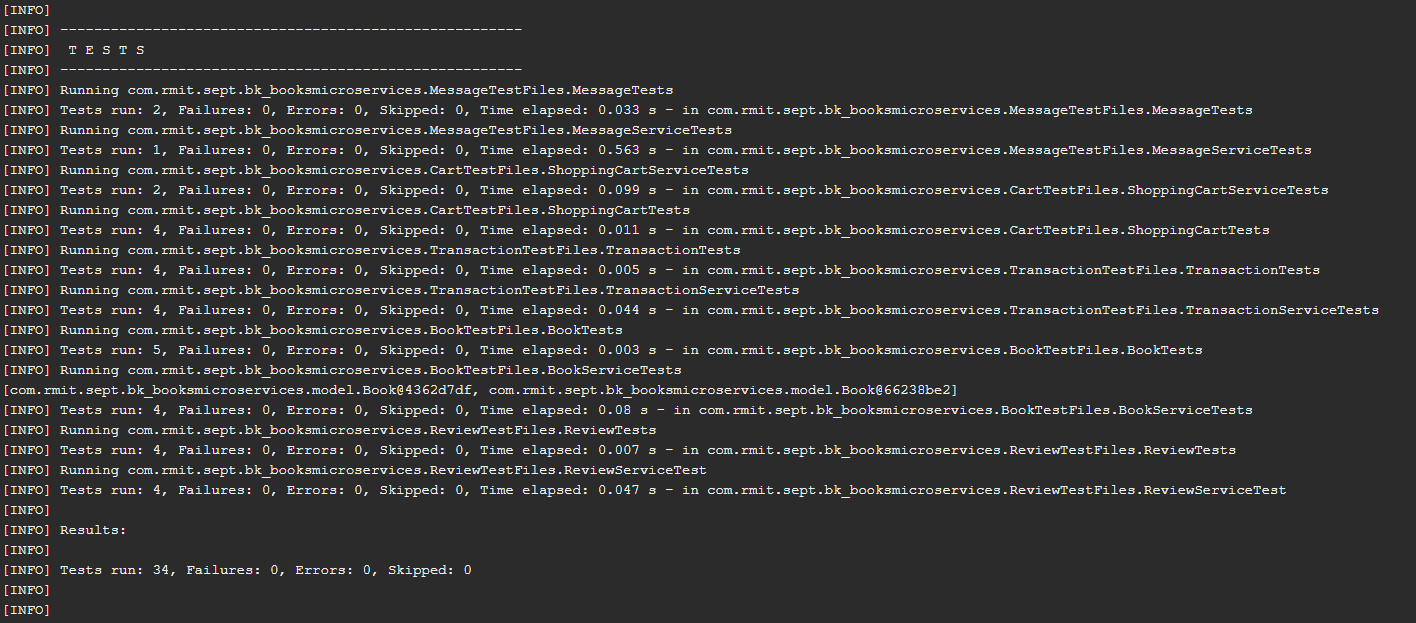
**Backend Test Execution**

The backend was tested using JUnit and Mockito. The main model classes as well as the service classes were tested whilst the controller classes were tested through the frontend tests. The backend was tested by using Mockito to create a dummy implementation of the service classes, and then using JUnit to assert the return values. Below are the appropriate tests being run and passed on CircleCi.

**Loginmicroservice tests**



**Bookmicroservice tests**



# Refactoring approach

After working on the application we quickly realised the importance of refactoring the code because as more functionality was added it became more difficult to make effective development. We noticed that as more utility around the book object was added it became harder to follow and bad smells were coming from code so to clean up our code we merged our back end api together to minimise coupling and increase simplicity. We employed a facade design pattern to hide the unnecessary complexity from our users and keep coupling to a minimum with our different objects like books, orders and users.

Before every sprint review we made sure to remove any unnecessary branches and imports that were no longer needed and added comments so the logic could easily be followed. Also having reusable components from the react framework made the code more flexible and minimised redundancy further refactoring the code. It was crucial that when we were refactoring when we made changes that the functionality was the same but logic was simplified. We made sure refactoring was a priority as it greatly affects the application flexibility in the future.

# CI/CD Pipeline

