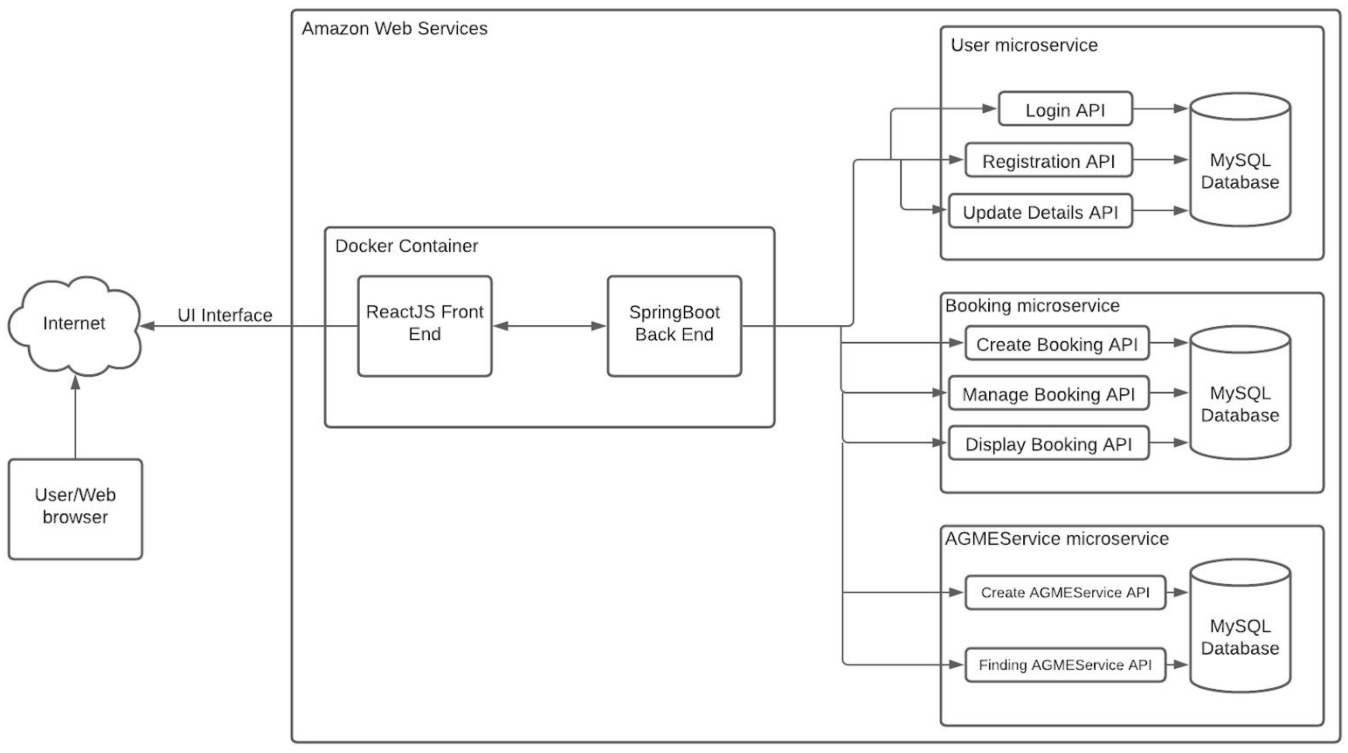
Project Report - 7.TUES-1430 Group 5

Vision Statement

The AGME Booking Service product provides a convenient and easy-to-handle usage of any kind of booking system for any kind of service, to both administrators and end-users. For business owners, AGME Booking Service provides a blank template to substitute whatever service their business offers. As such, they can easily and quickly set up a suite of available services without any coding knowledge required at all. For the customers, registration is an effortless process, in which they will be able to book their desired service in minimal time.

The user-interface was created with the intention to be flexible and easily adjustable to any type of business but can also be specialised for businesses with certain services in mind. They are clearly shown to the user, prioritising their attention to booking a desired service. The value of this product is clearly shown with its portability - enabling store owners to rapidly set up services for the end users, who are able to then book with minimal effort.

Final Basic System Architecture/Design

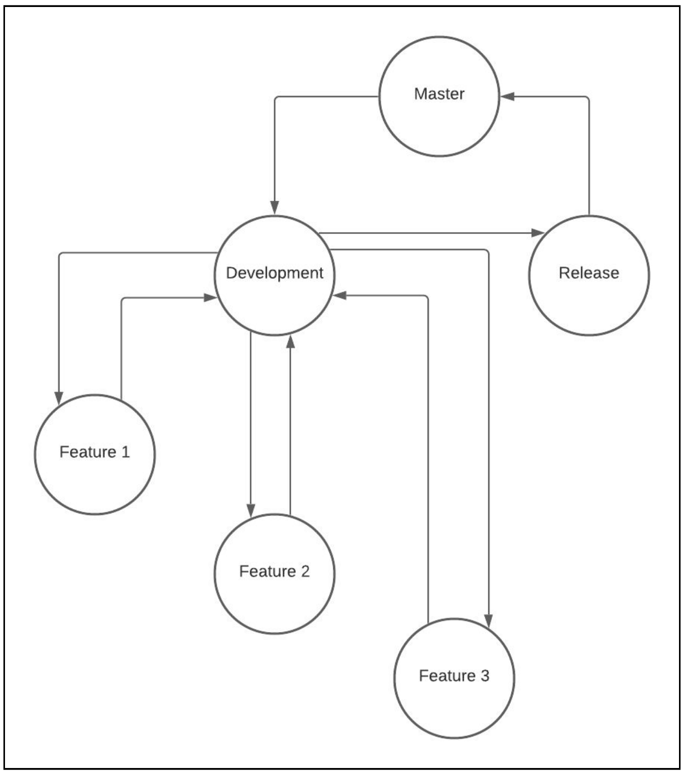


Documentation of Refactoring

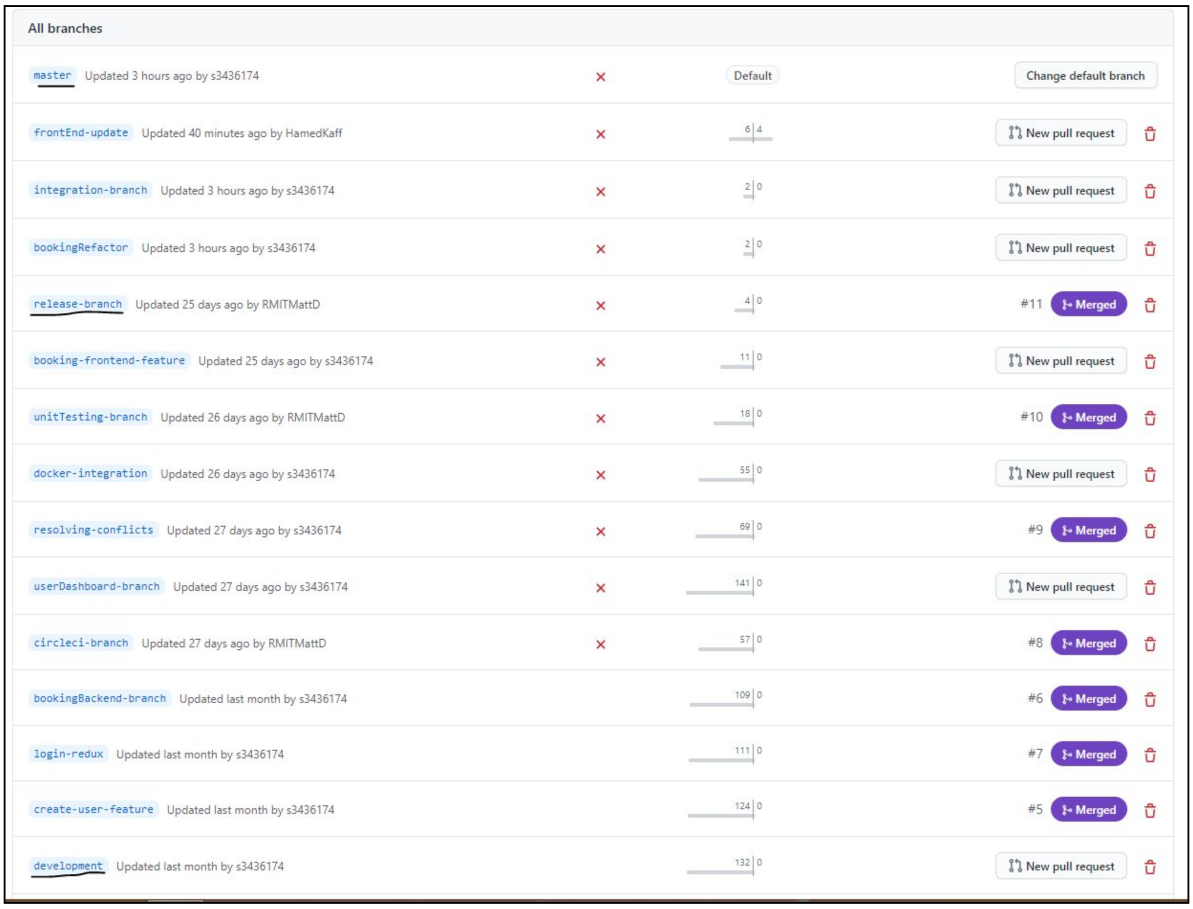
Refactoring Job	Code “Smells”/Reason for Refactoring	Effect
Changing all instances of class names starting with “Service” into “AGMEService”.	Having relevant model, service and controller classes for a business ‘service’ had been rather confusing. For example, the service class for a business service was called “ServiceService” and the controller was named “ServiceController”. It was simply not readable, both at an initial glance and for extended periods of time.	All types of classes; model, service, controller etc, for every part of the backend is now easily identifiable in both name and function. “AGMEService” much more clearly defines what they represent; a business ‘service’.
Refactored front end src files into their respective folders	Front end files for each component were not organised in folders and was a jumbled mess. It was hard to navigate and made importing specific components a chore. Also to separate each component so we knew what the functionality of the code was for.	Each component is properly organised allowing for easier code maintenance. The functionality of each file is also better defined.
Worker Model/Booking functionality to include available dates	We did not have a way to include availability for a worker in an easily accessible and modifiable method. Therefore we decided to modify the worker model to include a list of available dates related to their role/business.	The worker/user model includes a list that stores available dates for future bookings. This allows the front end to easily access and display availabilities for the worker selected.

GitFlow Overview and Commit Frequency

GitFlow Overview



Master Branch - Contains the latest deployable build of the product, always accompanied with release notes if coming from Release Branch.
Release Branch - An intermediary branch between merging Development to Master, in which any bugs or missed code are finished and finalised. This is only merged to master once it is deemed to be release-worthy.
Development Branch - Contains the most recent cumulation of feature branches that are at least in the first stages of being finalised. At the start of a project or after a new release on Master, the Development branch pulls over the changes from Master before any Feature Branches can be created.
Feature Branches - Contains one or more implementations of a certain feature/improvement by the project team. Any new feature branches almost always branch off from Development first.
The policy of feature branches is to usually implement many things in many areas, then merge them together. If a potential feature branch includes a change or addition that may affect other areas of the codebase, all other branches would ideally need to be finished/be in the first stages of being finalised and merged back to Development before this new branch can be created. The idea of this is to prevent as many merge conflicts as possible.

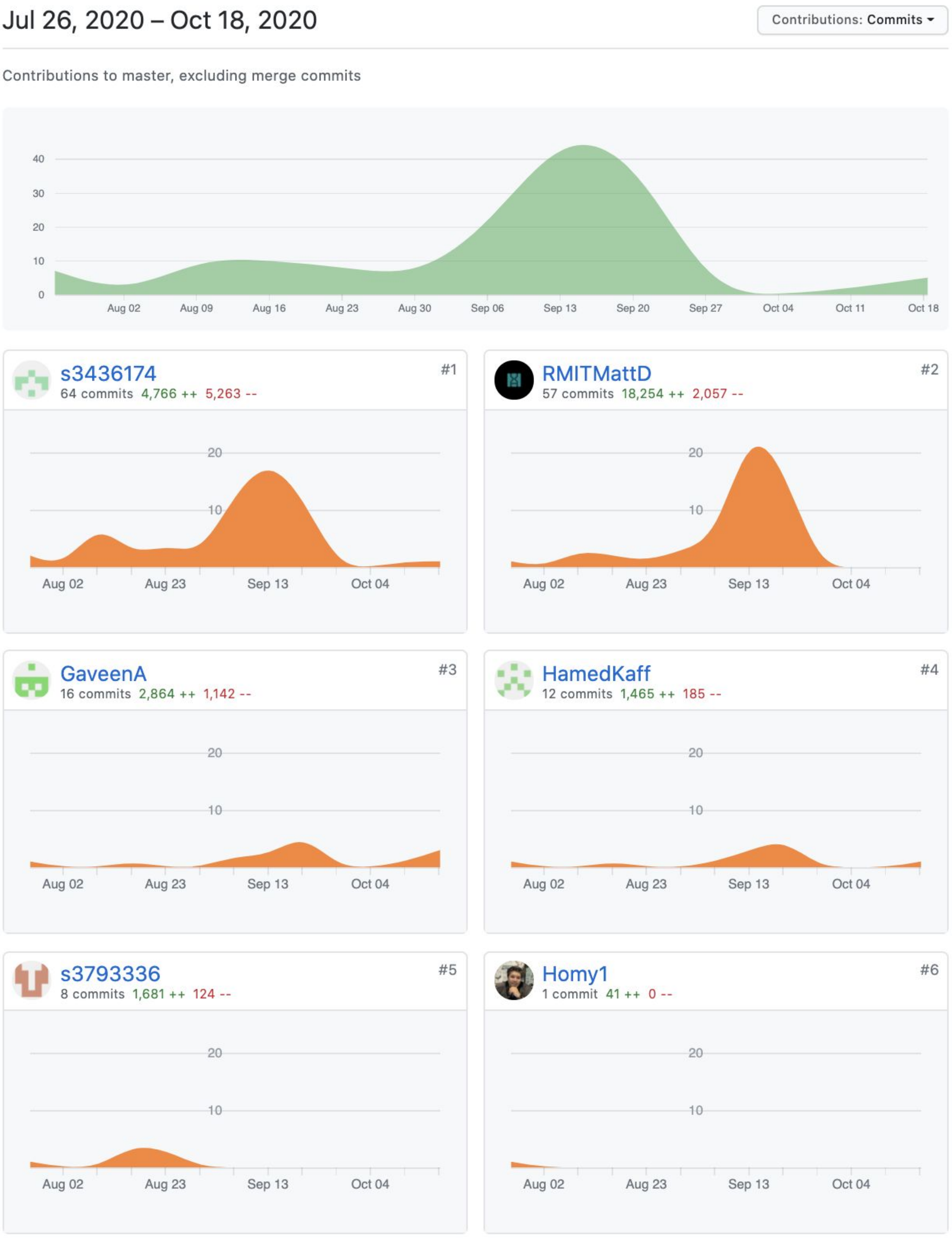


All branches

master Updated 3 hours ago by s3436174	✗	Default	Change default branch
frontEnd-update Updated 40 minutes ago by HamedKaff	✗	6 | 4	New pull request
integration-branch Updated 3 hours ago by s3436174	✗	2 | 0	New pull request
bookingRefactor Updated 3 hours ago by s3436174	✗	2 | 0	New pull request
release-branch Updated 25 days ago by RMITMattD	✗	4 | 0	#11 Merged
booking-frontend-feature Updated 25 days ago by s3436174	✗	11 | 0	New pull request
unitTesting-branch Updated 26 days ago by RMITMattD	✗	18 | 0	#10 Merged
docker-integration Updated 26 days ago by s3436174	✗	55 | 0	New pull request
resolving-conflicts Updated 27 days ago by s3436174	✗	69 | 0	#9 Merged
userDashboard-branch Updated 27 days ago by s3436174	✗	141 | 0	New pull request
circleci-branch Updated 27 days ago by RMITMattD	✗	57 | 0	#8 Merged
bookingBackend-branch Updated last month by s3436174		109 | 0	#6 Merged
login-redux Updated last month by s3436174		111 | 0	#7 Merged
create-user-feature Updated last month by s3436174		124 | 0	#5 Merged
development Updated last month by s3436174		132 | 0	New pull request

Figure Above - The list of most branches created within the Github repository.
Underlined are the Master, Development and Release branches, with all other branches, and some other unseen ones, being feature branches. Although told to leave them in for the purposes of marking, they would usually be deleted as soon as they have been finalised and merged back to the Development branch.

Commit Frequency



Jul 26, 2020 – Oct 18, 2020

Contributions: **Commits**

Contributions to master, excluding merge commits

s3436174 #1
64 commits 4,766 ++ 5,263 --

RMITMattD #2
57 commits 18,254 ++ 2,057 --

GaveenA #3
16 commits 2,864 ++ 1,142 --

HamedKaff #4
12 commits 1,465 ++ 185 --

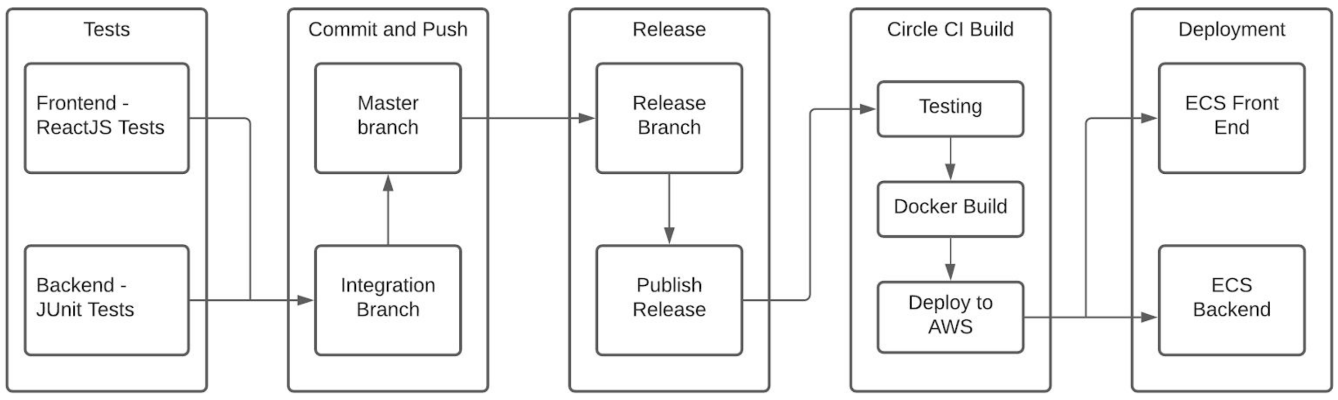
s3793336 #5
8 commits 1,681 ++ 124 --

Homy1 #6
1 commit 41 ++ 0 --

Description of Scrum Process

Gav was the Scrum Master for Sprints 0, 1 and 2. For Sprints 3 and 4, Matthew was the Scrum Master. The minimum number of Scrum Meetings per week was aimed to be two, including the lab meeting. For all Scrum Meetings, documentation of what happened was kept in the “Scrum Meeting Log” document in the team’s Google Drive folder.

Deployment Pipeline Diagram Setup



Documentation of Acceptance Test Cases/Test Execution

Refer to Acceptance Testing Document.