Team Members

Shreya Samanta, s3724266
Anthony Nguyen, s3783719
Catherine Thomas, s3784710
Stephanie Renata, s3726196
Jeremy Qing Siang Leong s3804383

Peer Assessment

Shreya

Contribution: 20.6%
Tasks:

- Implemented bookings table in Worker_Dashboard.js
- Created unit tests Worker_Dashoboard.test.js
- Implemented homepage enhancement feature in Home.js
- Edited all .css files in project to ensure consistency
- Redeployed project to AWS

Anthony

Contribution: 20.6%
Tasks:

- Implemented Service start_time, end_time, available_days
- Added unit tests for backend
- Added booking validation in the backend
- Added user validation in the backend
- Added service validation in the backend
- Things that surprised us slide
- One interesting thing slide

Catherine

Contribution: 17.6%
Tasks:

- Implemented table that shows available services in Booking.js
- Fixed service drop down menu to match service details shown in table above
- Implemented frontend validation when booking with unavailable services and default service option in Booking.js
- Fixed booking page to refresh after booking is made
- Created unit tests in AvailableServices.test.js
- Acceptance tests results 3a, 3b, 4a, 4b, 4c
- What went well slide
- One interesting feature slide

Stephanie

Contribution: 20.6%
Tasks:

- Implemented add availability page
- Implemented add availability front end validation
- Implemented booking time and date validation
- Created unit tests for add availability page
- Fixed input phone validation problem in registration page

Jeremy

Contribution: 20.6%
Tasks:

- Created a unit test for add worker (add_worker.test.js)
- Created add worker section
- Field validation for worker
- Connect add worker front end to backend
- Fixed booking double click issue

Links

Github: https://github.com/RMIT-SEPT/majorproject-7-tues-14-30-2
AWS: http://ec2-54-208-156-197.compute-1.amazonaws.com:3000/
Trello: https://trello.com/b/vSa4dbeY
Microsoft Teams:
https://teams.microsoft.com/l/team/19%3a1145220f1b6e49a9a4ac03edfee93837%40thread.tacv2/conversations?groupId=fe3fa0ad-be2d-425e-969a-10ff9b8f747b&tenantId=d1323671-cdbe-4417-b4d4-bdb24b51316b
Google Drive:
https://drive.google.com/drive/folders/10-KoLYkS51KQj6X_7uuhwXtt-BApLh3n?usp=sharing

New Code, Tests & Functionality

During this sprint the new user story "as a customer I want to view the application homepage so that I can see a landing page from which I can view available services before I log in or register" was created for the purpose of adding an enhancement feature to the project.

New code/test files:
BackEnd:

- ServicesServiceTests.java

FrontEnd:

- Home.js
- Home.css

- Add_Availability.js
- Add_Availability.css
- Add_Worker.js
- Add_Worker.css

Test Files:

- Worker_Dashboard.test.js
- AddAvailability.test.js
- Add_Worker.test.js
- AvailableServices.test.js

During this milestone we added the final functionality to ensure the application had a logical flow.
A table was added to the worker dashboard so that they could view a timetable of all of their bookings easily. As part of the enhancement feature, a home page was added to the application to replace the login page as the root page. This means that when users first open the application, they are greeted with a splash page welcoming them to the app and showing the current available bookings.
In the backend, three new attributes were added to the service model, as these were essential to defining the availability of a service. Additional booking validation was added, although some were removed in the end as there were issues with Postman and the front end sending different times. An additional user validation, to check for duplicate usernames, was added to prevent the creation of users with duplicate usernames.
In the booking page, a table that displays all of the available services with their respective worker, days, start time and end time was added. The service drop down was also changed to match the service displayed above.
In the worker dashboard, a button for adding new availability was added. When clicked, the product will show an add availability page where the worker can fill in the form to add their availability.
On the admin dashboard, there is a button for adding a new worker. When it has been clicked it will redirect to the add worker page which allows admin to insert the information of the new worker.

Communication

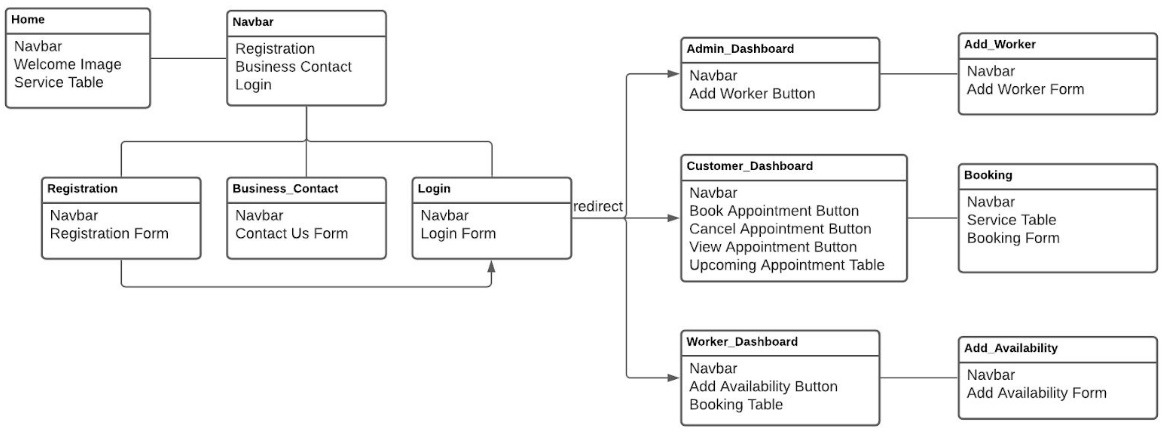
We have been using Microsoft Teams for our main form of communication amongst the team. As advised, we have added our tutor Sachin to the Team as we are unsure how to provide logs as of yet. The recordings links for some of our meetings are below:
https://web.microsoftstream.com/video/44d3e253-6046-437f-8360-b151dc3e4800
https://web.microsoftstream.com/video/15ed04bc-1e78-4404-95ad-3464653cf121
https://web.microsoftstream.com/video/61462bf5-f8c3-47ef-b871-0c5e9a2e2c2a
https://web.microsoftstream.com/video/cd015c24-ddcc-427f-a6c1-c27617528bc5
https://web.microsoftstream.com/video/28bebd93-c344-4f82-a016-ea0ee615e203

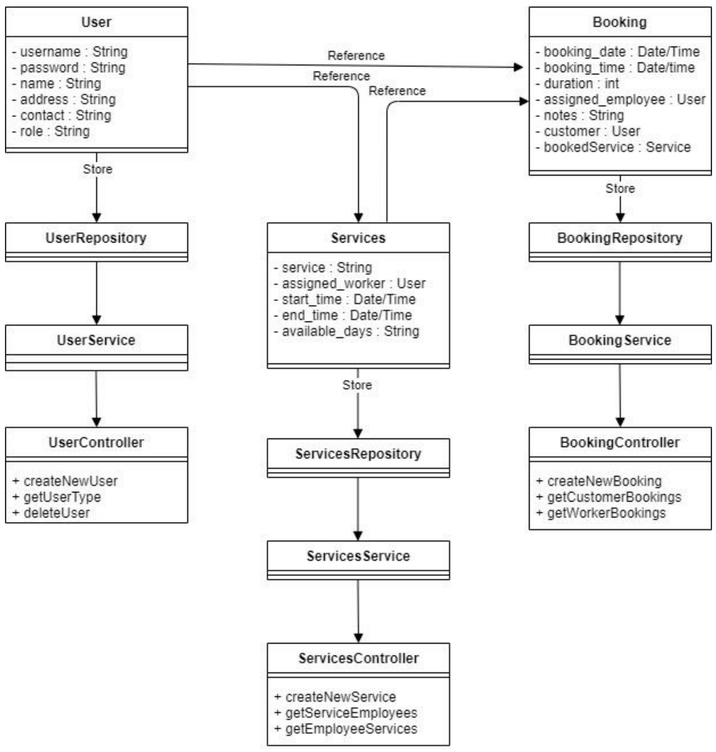
Vision Statement

The purpose of our application is to allow multiple businesses to use a collaborative platform to advertise their services. This is valuable to businesses as they can use our website to organise their services rather than have to create their own, and they receive website traffic they might not normally receive as more customers will visit our website than might have visited their own. This is also beneficial for users as they can view multiple services on a singular application rather than having to search out specific businesses for each service they require. Our application also offers benefits to workers, as they can offer multiple services and be employed by multiple businesses and have their timetable and bookings all located within our application.

What we aim to do to set our product apart from others in the market is increase the availability of our application. This would be done by integrating the application with popular social media services. As most users of the internet have some form of social media, integrating our application with social media apps already in use would raise traffic flow to our website as well as make it much more convenient to use for customers. This would be valuable to both users and businesses as an ideal integration would allow customers to book services without needing to navigate away from the social media application. We believe that this would give our application an advantage and distinguish it from similar applications.

Architecture / Design





Refactoring Report

At the beginning of the Sprint, the Services model did not have enough attributes. As such, new attributes were added, specifically start_time, end_time, and available_days. The three attributes defined the times that a user can make a booking and the days that they can make a booking. The needed to be refactored to accommodate for the times and days needed to properly define a service. The BookingService service was also refactored to fix some of the validation. As the validation was causing issues between the front end and the backend, as well as with Postman and the backend, the validation was removed as the issue could not be fixed to accommodate for both the frontend and Postman. Some other minor refactoring that occurred was that comments were added to mainly the services and controllers, as comments were missing.

At the beginning of the Sprint there are two booking validations for ensuring that the selected time is between the start and end time of the available service and the selected date is in the available days. However after we changed the service drop down from service id to service name, we cannot get the start time, end time and available days using the service id, so we need to get it using axios but it returns null to start time, end time and available days. As a result, the code needed to be refactored so that it only performs the validation if the start time, end time or available days is not returning an empty string. The other refactoring that is done was to add the comments to the code.

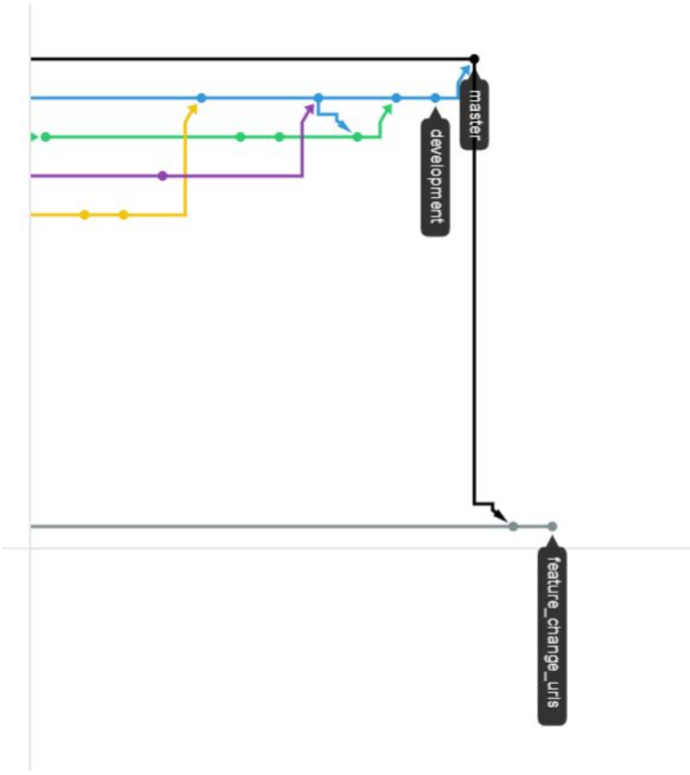
A new task which was created for this sprint was adding an enhancement feature to the project. This took the form of a home page which would replace the login page as the root page for the application. This required the refactoring of several files as the routing of the application needed to be changed. The code in App.js needed to be refactored to account for the addition of a new page, and the root “/” page needed to be changed from the Login.js page to the new Home.js page. The NavBar.js file also needed to be refactored to change the redirected page once a user has logged out to the home page, as well as creating a link to the homepage in the navbar when no user is logged in.

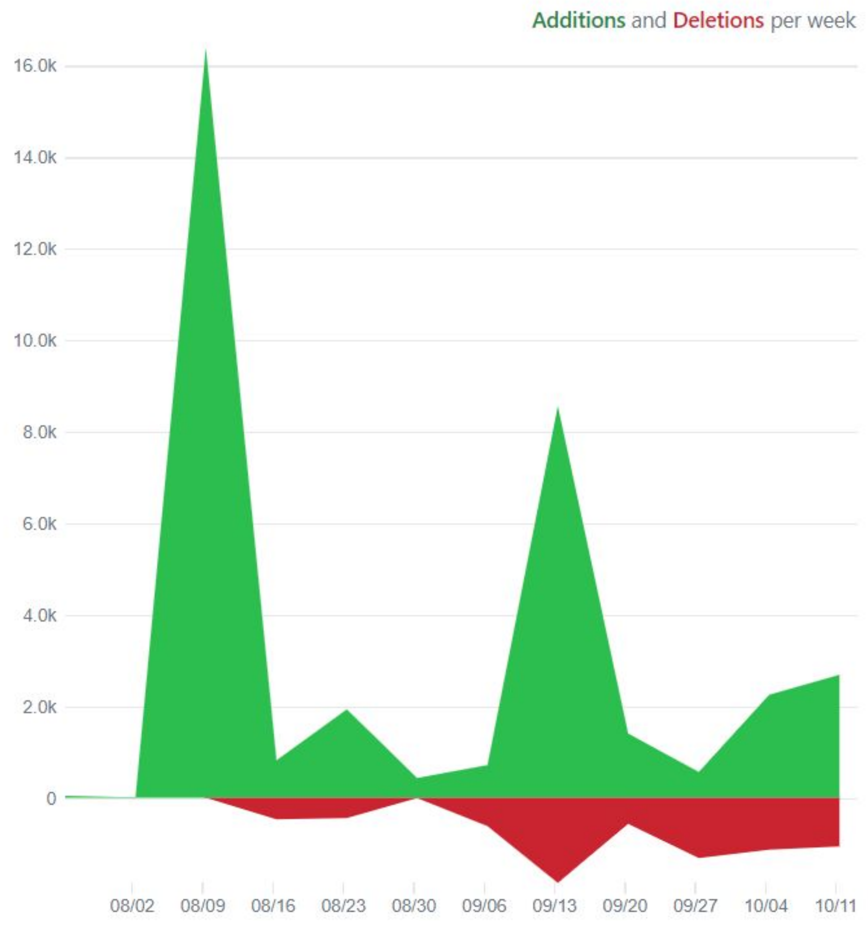
At the beginning of this sprint, there were two drop down menus on the booking page for selecting a service and a worker, however, this was changed just to one drop down menu for choosing the service. This was refactored because of the complications in the code whilst validating that the customer selects the correct worker for a service. This meant that when making a booking the worker name was assigned to null as it was not being selected by the user. Therefore, the code needed to be refactored to ensure the worker name was taken using axios after the user submits the booking form. Other refactoring that was done was adding comments to the code.

As passing worker name into booking API requires another axios request to get the worker name by doing so it stops the whole process of booking after it retrieved worker name from the backend. Which causes the book button to be pressed twice in order to make a booking successfully, therefore, it needs to be refactored so that the user is able to make a booking with one click. The other refactoring that is done was to add the comments to the code.

Gitflow organisation

Our Github repository was organised with the following branches master, development, hotfix and feature. The development branch is created off of the master branch and feature branches are opened off of the development branch and each of the feature branches will include unit tests and small commits that form up a functionality. The feature branch will get closed and merge into development once it has been tested out and functionality is working for every team member. Before merging the development branch into master we will do final testing on the development branch. If there is an error to fix in master branch a hotfix branch was opened off of it and it will get close after the error has been fixed.





Scrum Process

For the first milestone, the team had one sprint, Sprint 0, which lasted one week. In the second milestone, there were two sprints, Sprint 1 and 2, which were both three weeks long each. In the final milestone, there were also two sprints, Sprint 3 and 4, lasting two weeks and one week respectively.

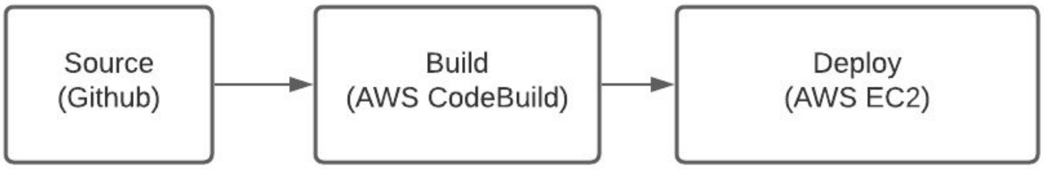
Every sprint began with a sprint planning, where items to be implemented in the sprint, were chosen from the product backlog, and were added to the sprint backlog. Each sprint ended with a sprint retrospective where the team discussed what went well, what could have gone better, any things that surprised us, and any lessons learned.

Our scrum master was Stephanie and the product owner was Shreya while the others, Anthony, Jeremy and Catherine, were part of the development team.

The team had meetings once a week with the client to discuss the progress of the product. Aside from this meeting, the team met twice more during the week, which followed the process of a regular scrum meeting where each member discussed the state of their progress and also raised any issues or clarifications. There were also some additional meetings before the submission of the milestones to ensure everything required for the milestone was completed and documented before submission.

During every meeting, meeting minutes were recorded to ensure that a proper agenda was followed and to document what each member will be working on and the timeframe taken to complete that task.

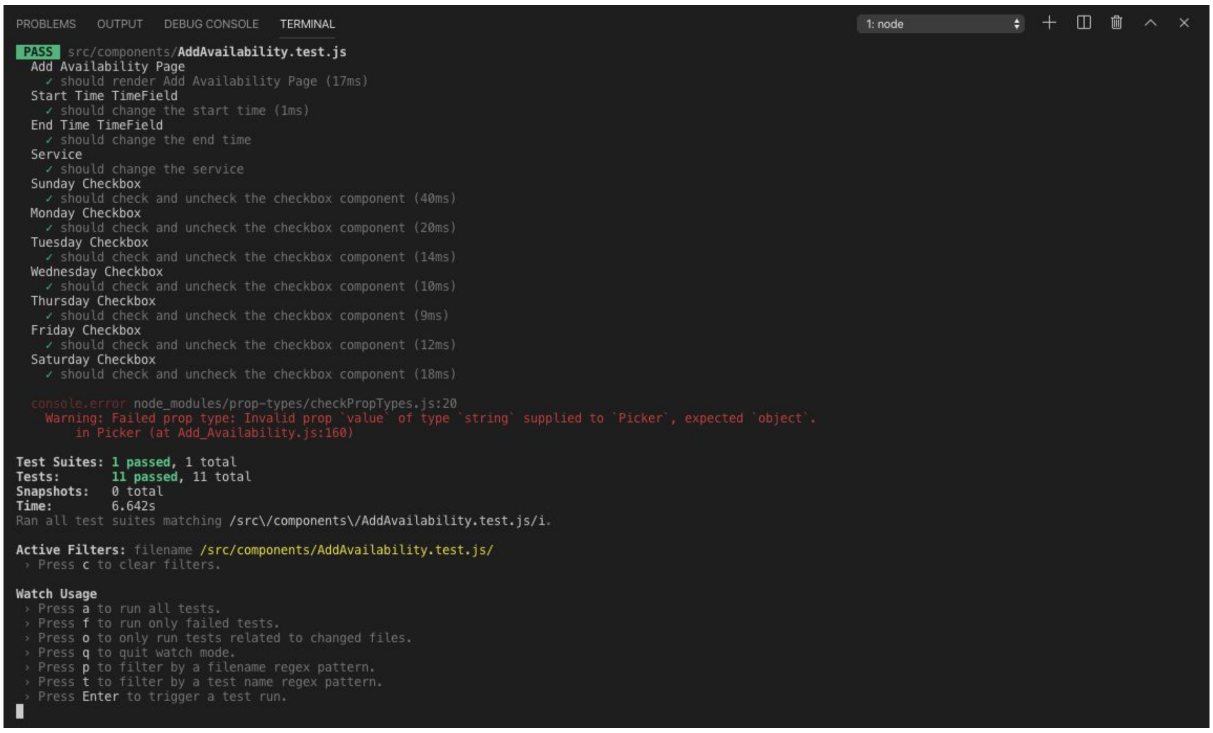
Deployment pipeline



Acceptance Test Documentation & Evidence of Test Execution

The documentation for acceptance tests is in the file “Acceptance Test Result” on GitHub.

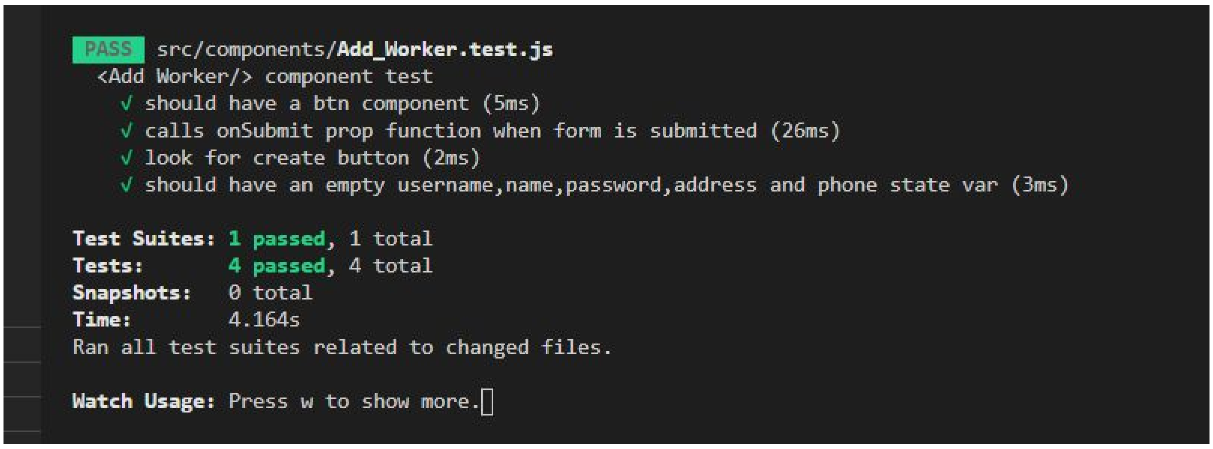
AddAvailability.test.js
This test file contains 11 test cases for the Add Availability page. The first test is to test the page render successfully. The second and third test is to check if the start time and end time is changed when the user selects a time. The next test is to test if the user can enter the service. The rest of the tests are used to test if the checkbox can be checked and unchecked.



```
 PASS  src/components/AddAvailability.test.js
  Add Availability Page
    ✓ should render Add Availability Page (17ms)
  Start Time TimeField
    ✓ should change the start time (1ms)
  End Time TimeField
    ✓ should change the end time
  Service
    ✓ should change the service
  Sunday Checkbox
    ✓ should check and uncheck the checkbox component (40ms)
  Monday Checkbox
    ✓ should check and uncheck the checkbox component (20ms)
  Tuesday Checkbox
    ✓ should check and uncheck the checkbox component (14ms)
  Wednesday Checkbox
    ✓ should check and uncheck the checkbox component (10ms)
  Thursday Checkbox
    ✓ should check and uncheck the checkbox component (9ms)
  Friday Checkbox
    ✓ should check and uncheck the checkbox component (12ms)
  Saturday Checkbox
    ✓ should check and uncheck the checkbox component (18ms)

  console.error node_modules/prop-types/checkPropTypes.js:20
    Warning: Failed prop type: Invalid prop `value` of type `string` supplied to `Picker`, expected `object`.
        in Picker (at Add_Availability.js:160)

Test Suites: 1 passed, 1 total
Tests:       11 passed, 11 total
Snapshots:   0 total
Time:        6.642s
Ran all test suites matching /src\/components\/AddAvailability.test.js/i.

Active Filters: filename /src/components/AddAvailability.test.js/
 › Press c to clear filters.

Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
 › Press o to only run tests related to changed files.
 › Press q to quit watch mode.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
▌
```

Add_Worker.test.js

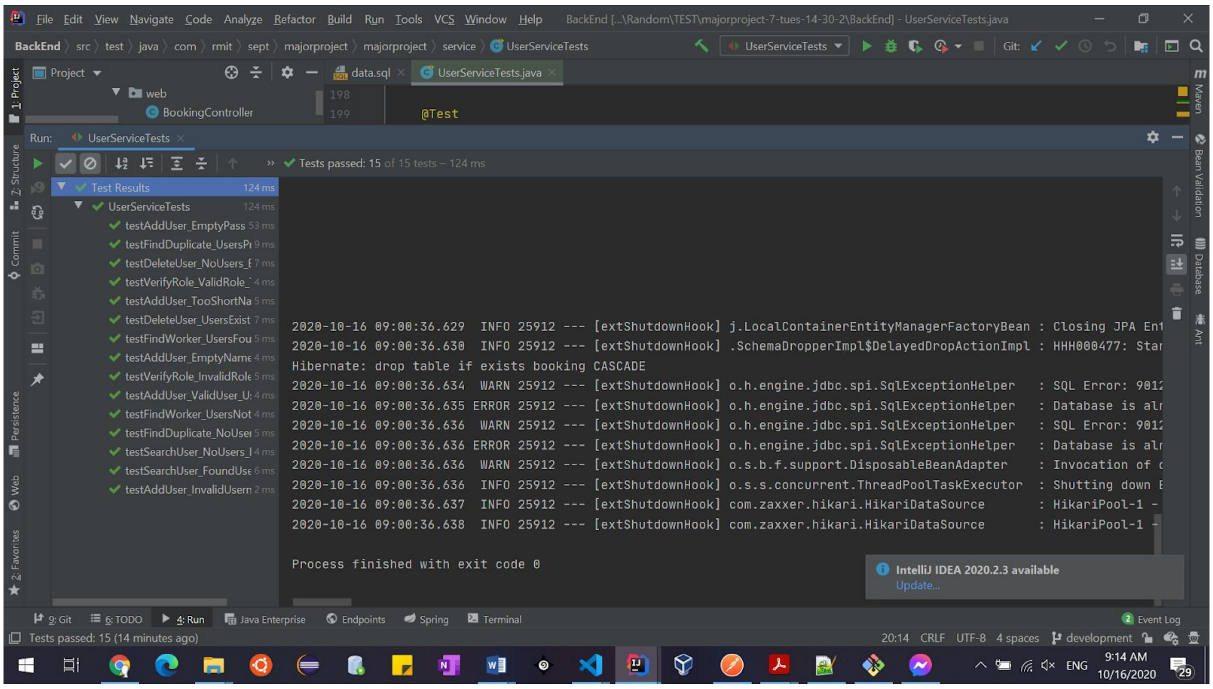
This test suite contains 4 tests the first test look for the only button on creating worker form the second test whether the form has been submitted once then button has been click. The third test whether did the button on create worker form have the matching text. The fourth test is to check whether username, name, password, address and phone are empty strings on setup.



```
PASS  src/components/Add_Worker.test.js
  <Add Worker/> component test
    √ should have a btn component (5ms)
    √ calls onSubmit prop function when form is submitted (26ms)
    √ look for create button (2ms)
    √ should have an empty username,name,password,address and phone state var (3ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        4.164s
Ran all test suites related to changed files.

Watch Usage: Press w to show more.
```

UserServiceTests.java:

The UserServiceTests.java contains a total of 15 tests, each one focusing on a method in the UserService.java. Before each of the tests are run, the init() method creates three test users, as we are mocking the return from certain UserRepository functions. The first test, testAddUser_ValidUser_User() tests that a user can be added when the values are valid, each assertion attempts to add a user to the mockUserService and expects it to return the user. The second test is testAddUser_EmptyPassword_Null() which tests that when given an invalid username, in this case the username is too short, then the return will be null. The third test is testAddUser_EmptyPassword_Null(), which test that when adding a user to the user service, if the password is left blank, then the return will be null. The fourth test, testAddUser_EmptyName_Null(), test that when the user has a blank name, then when adding the user to the user service, then the return will be null. The fifth test, testAddUser_TooShortName_Null(), test to see that when the name of the user is shorter than expected, than the return will be null. The sixth test in UserServicesTests.java is testSearchUser_NoUSers_Null(), tests that when searching for a user, based on username, that is not in the repository then the return is null. The seventh test, testSearchUser_FoundUser_User(), is the opposite of the previous test, where it tests that when the user is present in the repository then it will return the user. The eight test, testDeleteUser_NoUsers_Exception(), test that when attempting to delete a user that is not present in the repository, then a UserException exception is thrown. The ninth test, testDeleteUser_UserExists_Void(), tests that when deleting a user that exists in the repository then an exception is not thrown. The tenth test, testVerifyRole_InvalidRole_False(), unlike previous tests, doesn't test users themselves but checks the verifyRole() method in UserService.java. In this test, it checks that when parsing an invalid role to the method, then the function will return false. The eleventh test, testVerifyRole_ValidRole_True(), tests that when parsing a valid role to the verifyRole() function the return will be true, opposite to the previous test. The twelfth tests, testFindDuplicate_NoUsers_False(), tests that there are no users that already exist with the username parsed, and the expected result is true. The thirteenth test ,testFindDuplicate_UsersPresent_True(), is similar to the previous, but it will be expecting there to be a duplicate user. The last two tests, testFindWorker_UsersNotFound_Null() and testFindWorker_UsersFound_User2(), test that when parsing a worker name, the corresponding worker will be returned.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
BackEnd [...\Random\TEST\majorproject-7-tues-14-30-2\BackEnd] - UserServiceTests.java
BackEnd src test java com rmit sept majorproject majorproject service UserServiceTests
Project
data.sql
UserServiceTests.java
web
BookingController
Run: UserServiceTests
Git
TODO
Run
Java Enterprise
Endpoints
Spring
Terminal
Event Log
IntelliJ IDEA 2020.2.3 available
Update...

```
@Test
```

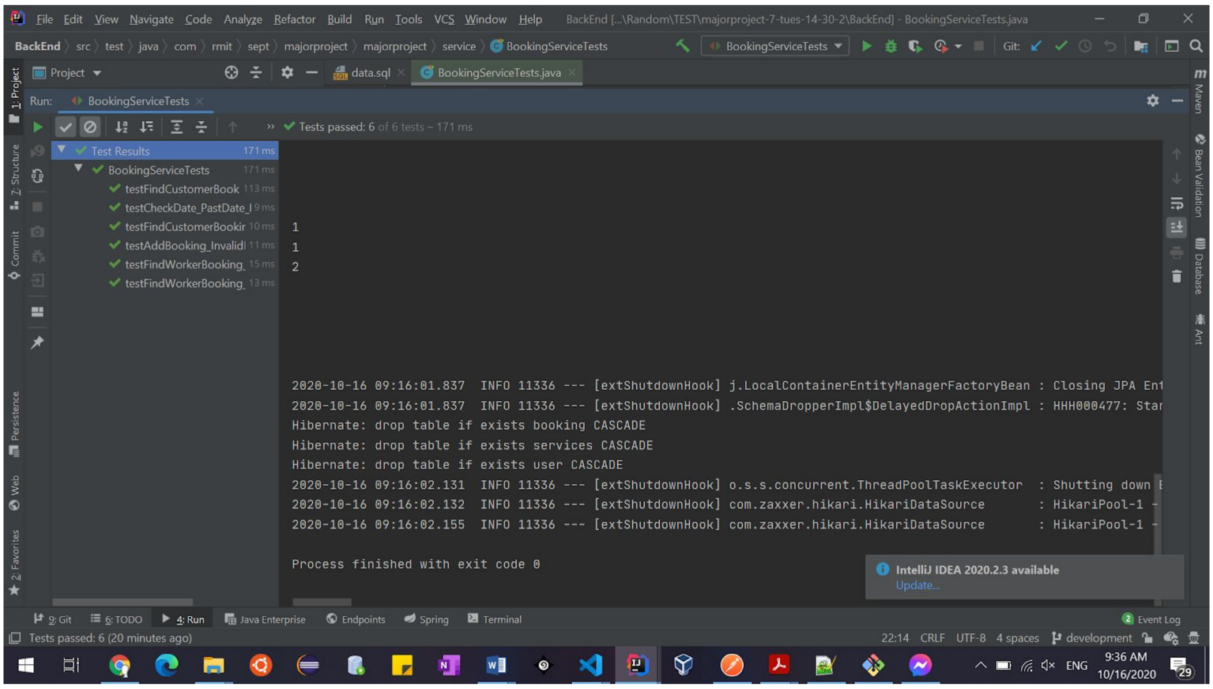
Tests passed: 15 of 15 tests – 124 ms

```
Test Results                     124 ms
  UserServiceTests               124 ms
    testAddUser_EmptyPass         53 ms
    testFindDuplicate_UsersPı      9 ms
    testDeleteUser_NoUsers_E       7 ms
    testVerifyRole_ValidRole_      4 ms
    testAddUser_TooShortNa         5 ms
    testDeleteUser_UsersExist      7 ms
    testFindWorker_UsersFou        5 ms
    testAddUser_EmptyName          4 ms
    testVerifyRole_InvalidRole     5 ms
    testAddUser_ValidUser_U        4 ms
    testFindWorker_UsersNot        4 ms
    testFindDuplicate_NoUseı       5 ms
    testSearchUser_NoUsers_I       4 ms
    testSearchUser_FoundUse        6 ms
    testAddUser_InvalidUsern       2 ms
```

```
2020-10-16 09:00:36.629  INFO 25912 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Ent
2020-10-16 09:00:36.630  INFO 25912 --- [extShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Star
Hibernate: drop table if exists booking CASCADE
2020-10-16 09:00:36.634  WARN 25912 --- [extShutdownHook] o.h.engine.jdbc.spi.SqlExceptionHelper   : SQL Error: 9012
2020-10-16 09:00:36.635 ERROR 25912 --- [extShutdownHook] o.h.engine.jdbc.spi.SqlExceptionHelper   : Database is alr
2020-10-16 09:00:36.636  WARN 25912 --- [extShutdownHook] o.h.engine.jdbc.spi.SqlExceptionHelper   : SQL Error: 9012
2020-10-16 09:00:36.636 ERROR 25912 --- [extShutdownHook] o.h.engine.jdbc.spi.SqlExceptionHelper   : Database is alr
2020-10-16 09:00:36.636  WARN 25912 --- [extShutdownHook] o.s.b.f.support.DisposableBeanAdapter    : Invocation of c
2020-10-16 09:00:36.636  INFO 25912 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down E
2020-10-16 09:00:36.637  INFO 25912 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
2020-10-16 09:00:36.638  INFO 25912 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -

Process finished with exit code 0
```

Tests passed: 15 (14 minutes ago)   20:14 CRLF UTF-8 4 spaces development   9:14 AM 10/16/2020

BookingServiceTests.java:
BookingServiceTests.java currently contains a total of six tests, focusing mainly on date validation methods in BookingService. Before each test, the test class mocks a return, so that when the bookingRepository class function, save(), is called it will return the booking. The first test, testCheckDate_PastDate_Minus1(), tests that when the checkDate() function is called on a date that occurred in the past, then it will return -1. This assertion is performed three times, on three different dates, one which has a past year, one that has a past month, and one that has a past day. The second test, testAddBooking_InvalidDate_Null, tests that when a booking with an invalid date is added through saveOrUdateBooking(), then the function will return null. The next test was testFindCustomerBooking_NoneFound_EmptyList(), it tests the findCustBooking() function. It expects an empty list to be returned as there are no bookings belonging to the parsed user. The next test, testFindCustomerBooking_Found_NotEmptyList(), is similar to the previous test, however it is the reverse, and expects there to be two bookings in the returned list. The next two tests, testFindWorkerBooking_NoneFound_EmptyList() and testFindWorkerBooking_BookingFound_FilledList(), are similar to the two previous tests, however instead of searching for customer bookings, it searches for worker bookings.

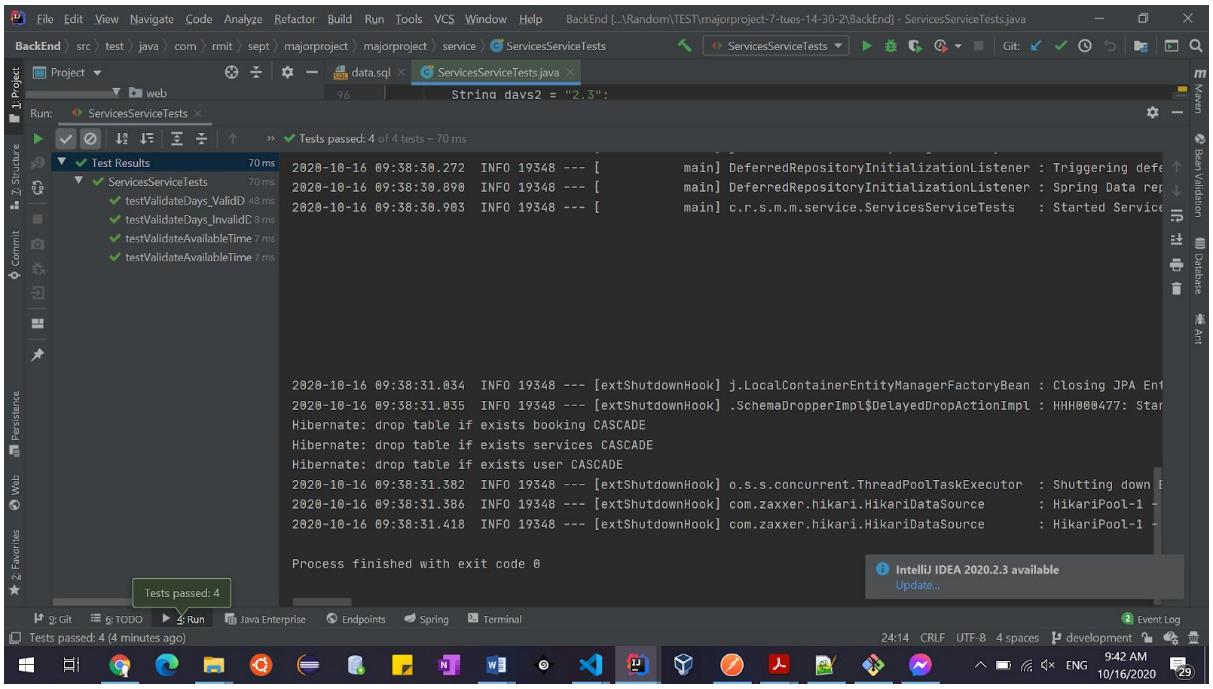


```
Tests passed: 6 of 6 tests – 171 ms

Test Results                        171 ms
  BookingServiceTests               171 ms
    testFindCustomerBook            113 ms
    testCheckDate_PastDate_I          9 ms
    testFindCustomerBookir           10 ms
    testAddBooking_InvalidI          11 ms
    testFindWorkerBooking_           15 ms
    testFindWorkerBooking_           13 ms
```

```
1
1
2



2020-10-16 09:16:01.837  INFO 11336 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Ent
2020-10-16 09:16:01.837  INFO 11336 --- [extShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Star
Hibernate: drop table if exists booking CASCADE
Hibernate: drop table if exists services CASCADE
Hibernate: drop table if exists user CASCADE
2020-10-16 09:16:02.131  INFO 11336 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down E
2020-10-16 09:16:02.132  INFO 11336 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
2020-10-16 09:16:02.155  INFO 11336 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -

Process finished with exit code 0
```

Tests passed: 6 (20 minutes ago)

ServicesServiceTest.java:
ServicesServiceTests.java has a total of four tests, focusing on functions in ServicesService.java. The first two tests, testValidateAvailableTimes_ValidTimes_True() and testValidateAvailableTimes_InvalidTimes_False(), test the validateAvailableTimes() function. They expect the parsed services to have valid available times and invalid available times respectively. The last two tests, testValidateDays_ValidDays_True() and testValidateDays_InvalidDays_False(), test the validateDays() function. The first one expects the services parsed to have valid days (i.e. integers between 1-7, inclusive, and separated by commas). While the second one expects the days to be invalid, for example, an integer outside the expected range.

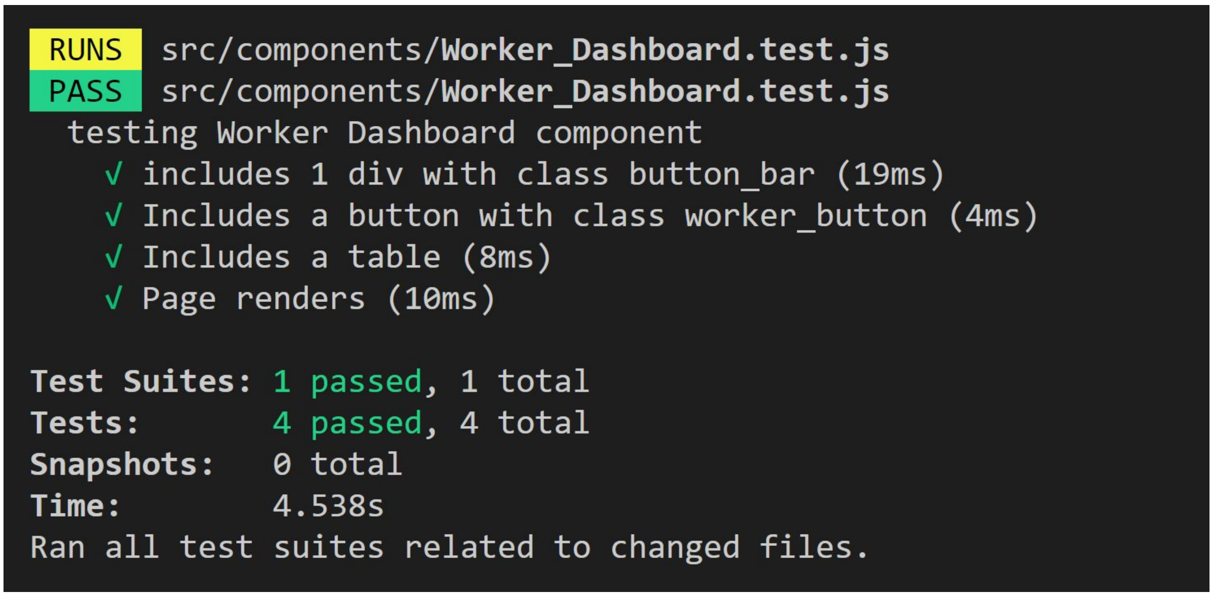


```
Tests passed: 4 of 4 tests – 70 ms

Test Results                        70 ms
  ServicesServiceTests              70 ms
    testValidateDays_ValidD         48 ms
    testValidateDays_InvalidD        8 ms
    testValidateAvailableTime        7 ms
    testValidateAvailableTime        7 ms
```

```
2020-10-16 09:38:30.272  INFO 19348 --- [           main] DeferredRepositoryInitializationListener : Triggering defe
2020-10-16 09:38:30.890  INFO 19348 --- [           main] DeferredRepositoryInitializationListener : Spring Data rep
2020-10-16 09:38:30.903  INFO 19348 --- [           main] c.r.s.m.m.service.ServicesServiceTests   : Started Service



2020-10-16 09:38:31.034  INFO 19348 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Ent
2020-10-16 09:38:31.035  INFO 19348 --- [extShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Star
Hibernate: drop table if exists booking CASCADE
Hibernate: drop table if exists services CASCADE
Hibernate: drop table if exists user CASCADE
2020-10-16 09:38:31.382  INFO 19348 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down E
2020-10-16 09:38:31.386  INFO 19348 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
2020-10-16 09:38:31.418  INFO 19348 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -

Process finished with exit code 0
```

Tests passed: 4 (4 minutes ago)

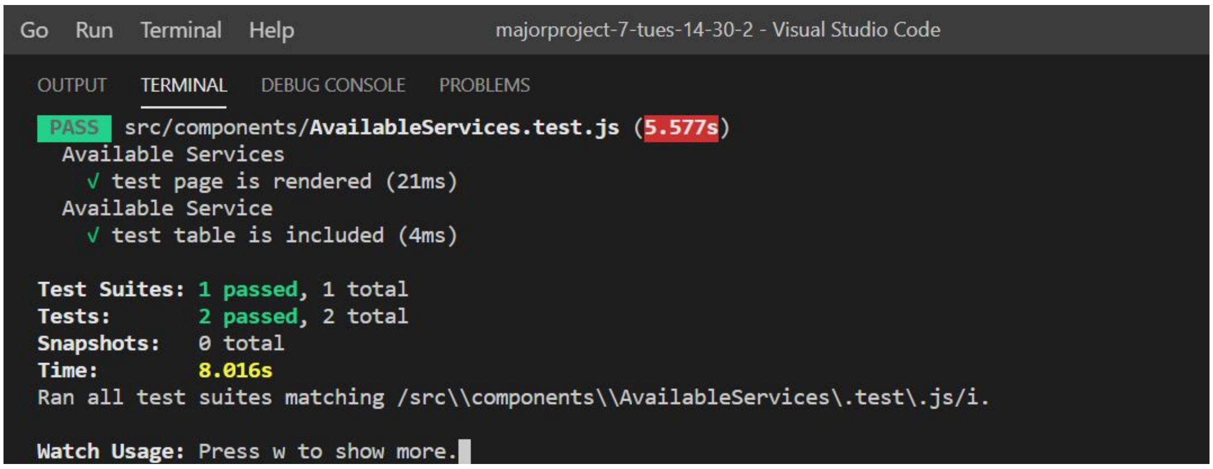
Worker_Dashboard.test.js
The worker dashboard tests test the formatting of the worker dashboard is what is required. These react tests test that the page has rendered and that it has rendered each element correctly. Aside from testing that the page has rendered, they test that the page contains one button of the correct styling and a table which is used to display the worker's bookings.



```
 RUNS  src/components/Worker_Dashboard.test.js
 PASS  src/components/Worker_Dashboard.test.js
  testing Worker Dashboard component
    √ includes 1 div with class button_bar (19ms)
    √ Includes a button with class worker_button (4ms)
    √ Includes a table (8ms)
    √ Page renders (10ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        4.538s
Ran all test suites related to changed files.
```

AvailableServices.test.js

This test file has two tests that test the available services table on the booking page. The first one tests if the page has rendered correctly, as currently the page will not render if the available services have not been successfully fetched from the back end. The second tests that the available services table is included in the booking page.



```
 PASS  src/components/AvailableServices.test.js (5.577s)
  Available Services
    √ test page is rendered (21ms)
  Available Service
    √ test table is included (4ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        8.016s
Ran all test suites matching /src\\components\\AvailableServices\.test\.js/i.

Watch Usage: Press w to show more.
```