Team Members

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

Peer Assessment

Shreya

Contribution: 21.85%
Tasks:

- Created pages for Customer Dashboard, Worker Dashboard and Admin Dashboard for redirection after login
- Created Booking page
- Created table to display worker services
- Created request to the backend to retrieve worker services
- Unit tests for Customer Dashboard
- Dockerized project
- Deployed project through AWS EC2
- Documented all meetings in meeting minutes

Anthony

Contribution: 21.85%
Tasks:

- Created models, repositories, services, and controllers for User, Booking, Services
- Implemented backend functionality for:
  - Login
  - Registration (i.e. create new users)
  - Creating bookings
  - Creating services
  - Getting bookings based on customer/worker
  - Getting services based on name or by worker detail
- Basic backend validation
- Unit Tests for User and Booking service functions
- Helped with CircleCi yml file for backend

Catherine

Contribution: 12.6%
Tasks:

- Unit tests for confirmation message on booking page
- Implemented confirmation message on booking page
- Implemented table to display available services
- Implemented drop-down menu of services
- Implemented front-end validation for booking feature

Stephanie

Contribution: 21.85%
Tasks:

- Unit test for booking field components
- Implemented booking fields frontend
- Set up circleci for frontend and backend
- Implemented registration form
- Backend and frontend validation for registration
- Connect frontend and backend for registration
- Burndown charts

Jeremy

Contribution: 21.85%
Tasks:

- Implemented functionality:
    - Redirect to the appropriate page based on the type of user
    - Authentication and authentication for user type
    - Connection between frontend and backend for booking feature
    - Unit tests for login component
    - Login page frontend
    - Page for business contact
    - Linked navbar button with appropriate page

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/
Docker Images: shreysaman/majorproject-7-tues-14-30-2_backend, shreysaman/majorproject-7-tues-14-30-2_frontend
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 and Functionality

As there was little functionality made in Sprint 0, a lot of the functionality was made in Sprints 1 and 2, where Sprint 1 was more focused on building the base code for functionality, such as basic dashboard pages, rather than full on functioning dashboards. Sprint 2 introduced some of the more important and major functions, as well as linking the front end to the back end.

After completing the sprints in this milestone we now have functionality to login as a user and register as a new customer, then be redirected to the relevant dashboard. Customers are also able to navigate to the bookings page from their dashboard and book an appointment. Some of the back end functionality that will be integrated in future sprints are functionality to create services, retrieving a list of bookings made by certain customers or bookings made with certain workers.

New code/test files:
Backend:
- UserException.java
- JwtRequestFilter.java
- AuthenticationReques.java
- AuthenticationResponse.java
- Booking.java
- Services.java
- User.java
- BookingRepository.java
- ServicesRepository.java
- UserRepositiory.java
- BookingService.java
- ServicesService.java
- UserSerivice.java
- JwtUtil.java
- SercurityConfigurer.java
- BookingController.java
- LoginController.java
- ServicesController.java
- userController.java

FrontEnd
- Admin_Dashboard.js
- Admin_Dashboard.css
- Booking.js

- Booking.css
- Customer_Dashboard.js
- Customer_Dashboard.css
- Worker_Dashboard.js
- Worker_Dashboard.css

Test Files:
- BookingServiceTests.java
- UserServiceTests.java
- Login.test.js
- Customer_Dashboard.test.js
- BookingField.test.js
- BookingMessage.test.js

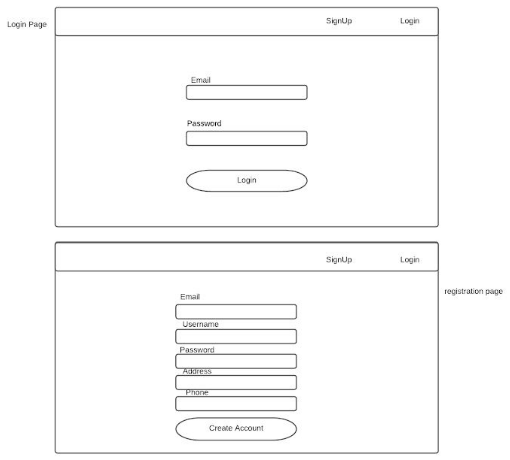
Communication

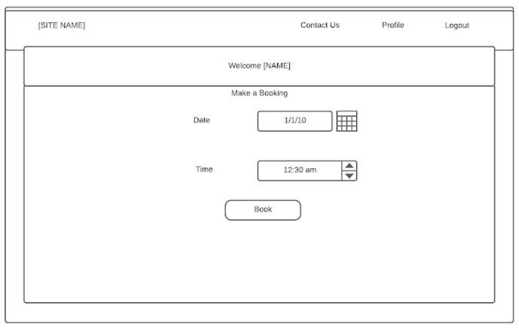
We have been using Microsoft Teams for our main form of communication amongst the team. As advised, we have added our tutor Sachin to the Team as we are unsure how to provide logs as of yet. The recordings links for some of our meetings are below:

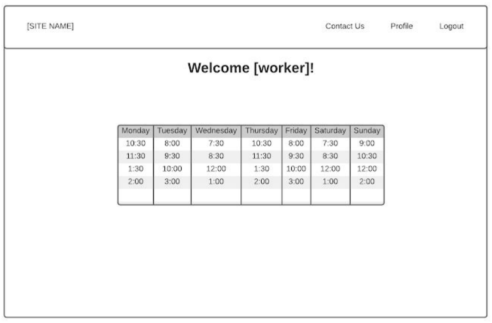
https://web.microsoftstream.com/video/a03f08b3-6f4a-4c0b-8d78-bfbaaad174f0
https://web.microsoftstream.com/video/cae2d4b2-619e-4fc8-903f-8c93a826073b
https://web.microsoftstream.com/video/19036205-41c6-422f-9375-545c2dcae977
https://web.microsoftstream.com/video/84d9c48c-534c-4da1-96c9-bb2396250e22

Design Description

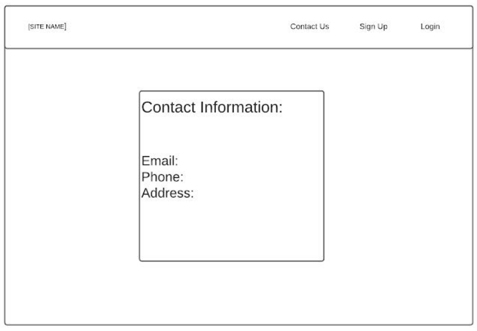
As we had multiple team members working on developing the frontend at the same time, we created a rough wireframe that details what the UI should look like in order to ensure it remained consistent.







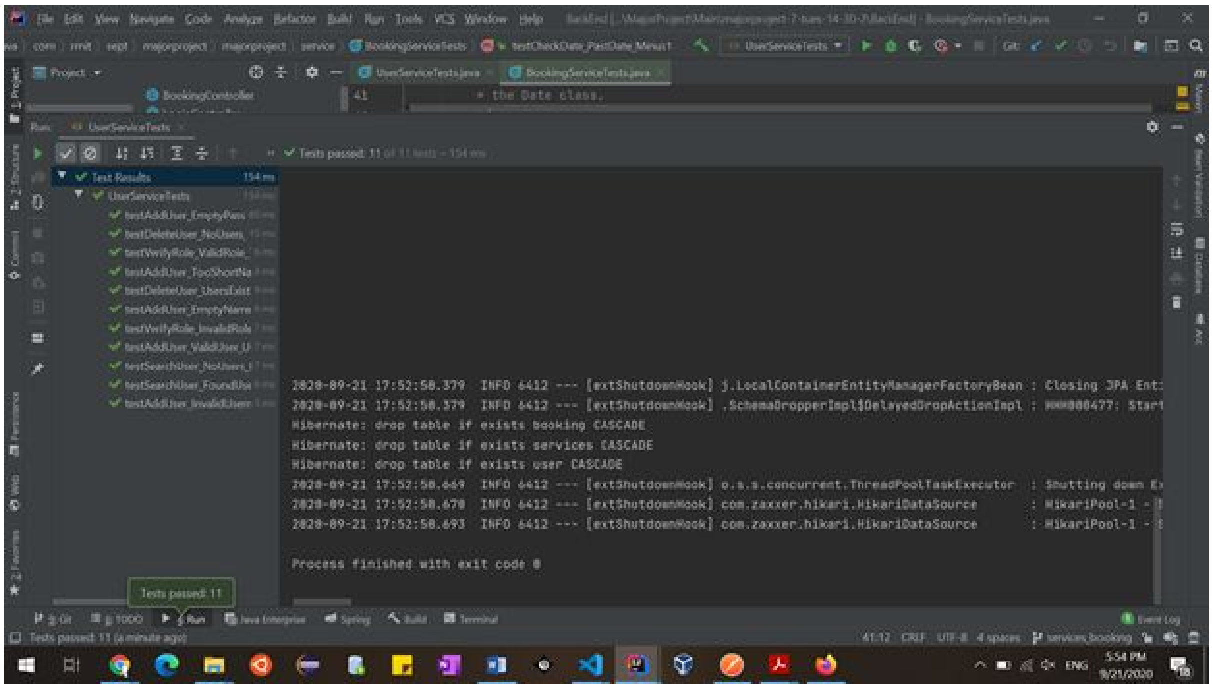
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
10:30	8:00	7:30	10:30	8:00	7:30	9:00
11:30	9:30	8:30	11:30	9:30	8:30	10:30
1:30	10:00	12:00	1:30	10:00	12:00	12:00
2:00	3:00	1:00	2:00	3:00	1:00	2:00



Test Documentation

UserServicesTests.java:
The UserServicesTests.java test class currently contains a total of 11 tests. 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 thown. 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.



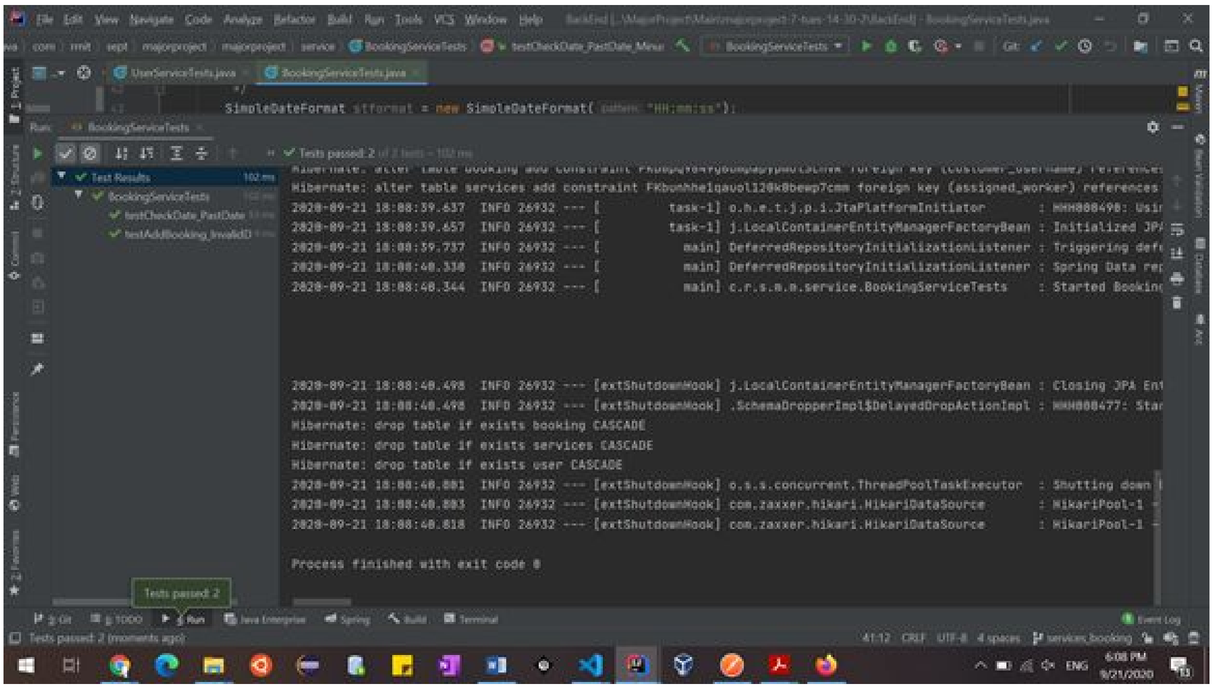
```
Tests passed: 11 of 11 tests – 154 ms

Test Results
  UserServiceTests
    testAddUser_EmptyPass
    testDeleteUser_NoUsers_
    testVerifyRole_ValidRole_
    testAddUser_TooShortNa
    testDeleteUser_UsersExist
    testAddUser_EmptyName
    testVerifyRole_InvalidRole
    testAddUser_ValidUser_U
    testSearchUser_NoUsers_
    testSearchUser_FoundUse
    testAddUser_InvalidUsern
```

```
2020-09-21 17:52:50.379  INFO 6412 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Ent
2020-09-21 17:52:50.379  INFO 6412 --- [extShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Start
Hibernate: drop table if exists booking CASCADE
Hibernate: drop table if exists services CASCADE
Hibernate: drop table if exists user CASCADE
2020-09-21 17:52:50.669  INFO 6412 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down E
2020-09-21 17:52:50.670  INFO 6412 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
2020-09-21 17:52:50.693  INFO 6412 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -

Process finished with exit code 0
```

BookingServiceTests.java:
BookingServiceTests.java currently contains a total of two 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 checkDtae() 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 final test, testAddBooking_InvalidDate_Null, tests that when a booking with an invalid date is added through saveOrUdateBooking(), then the function will return null.



```
SimpleDateFormat stformat = new SimpleDateFormat( pattern: "HH:mm:ss");
```

✓ Tests passed: 2 of 2 tests – 102 ms

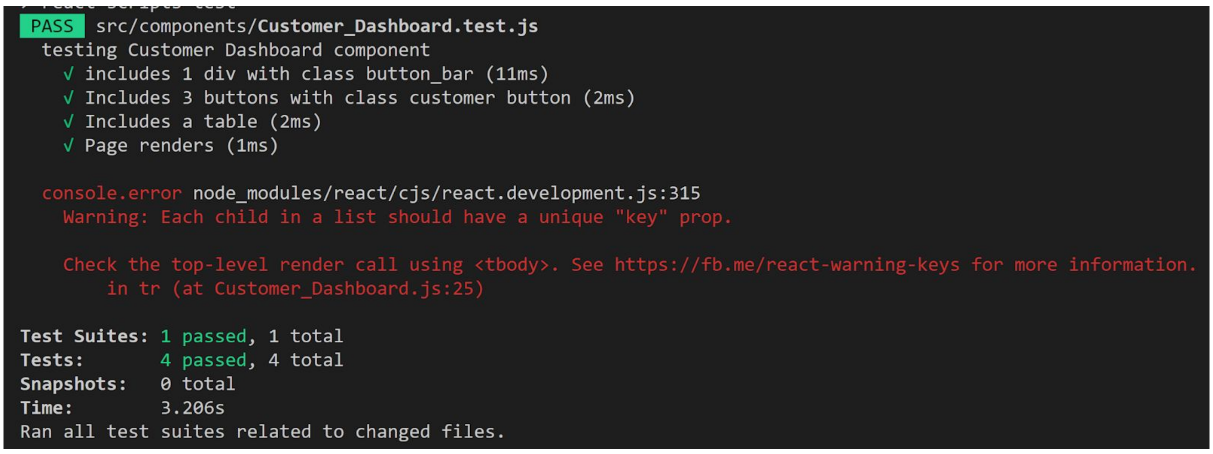
```
Hibernate: alter table services add constraint FKbunhhe1qauol120k8bewp7cmm foreign key (assigned_worker) references
2020-09-21 18:08:39.637  INFO 26932 --- [         task-1] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Usin
2020-09-21 18:08:39.657  INFO 26932 --- [         task-1] j.LocalContainerEntityManagerFactoryBean : Initialized JPA
2020-09-21 18:08:39.737  INFO 26932 --- [           main] DeferredRepositoryInitializationListener : Triggering defe
2020-09-21 18:08:40.330  INFO 26932 --- [           main] DeferredRepositoryInitializationListener : Spring Data rep
2020-09-21 18:08:40.344  INFO 26932 --- [           main] c.r.s.m.m.service.BookingServiceTests    : Started Bookin



2020-09-21 18:08:40.498  INFO 26932 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Ent
2020-09-21 18:08:40.498  INFO 26932 --- [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
2020-09-21 18:08:40.801  INFO 26932 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down
2020-09-21 18:08:40.803  INFO 26932 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
2020-09-21 18:08:40.818  INFO 26932 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -

Process finished with exit code 0
```

Customer_Dashboard.test.js
These tests test the rendering of the Customer Dashboard and ensure it has all the requirements necessary of the page as specified in the assignment specification. The tests check that the page itself renders, as well as contains the necessary buttons and table layout.



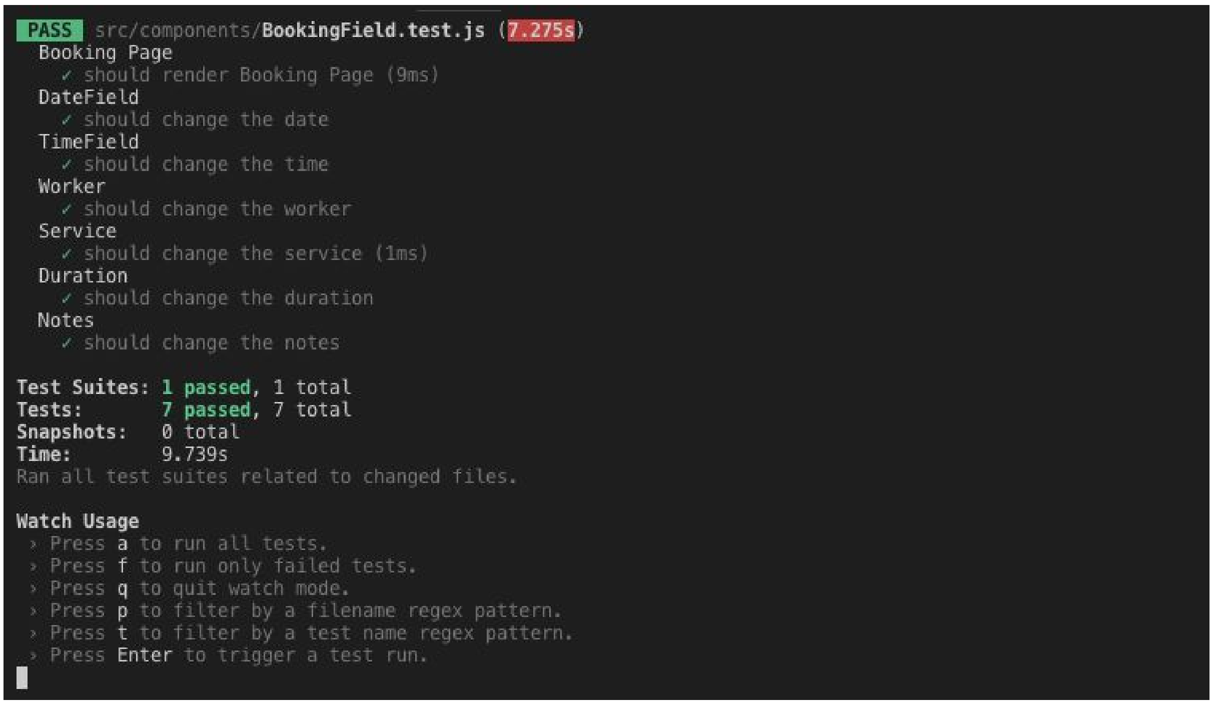
```
 PASS  src/components/Customer_Dashboard.test.js
  testing Customer Dashboard component
    √ includes 1 div with class button_bar (11ms)
    √ Includes 3 buttons with class customer button (2ms)
    √ Includes a table (2ms)
    √ Page renders (1ms)

  console.error node_modules/react/cjs/react.development.js:315
    Warning: Each child in a list should have a unique "key" prop.

    Check the top-level render call using <tbody>. See https://fb.me/react-warning-keys for more information.
        in tr (at Customer_Dashboard.js:25)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        3.206s
Ran all test suites related to changed files.
```

BookingField.test.js
These tests test the rendering of the Booking page and checking each booking field changes when the user typed in the value.



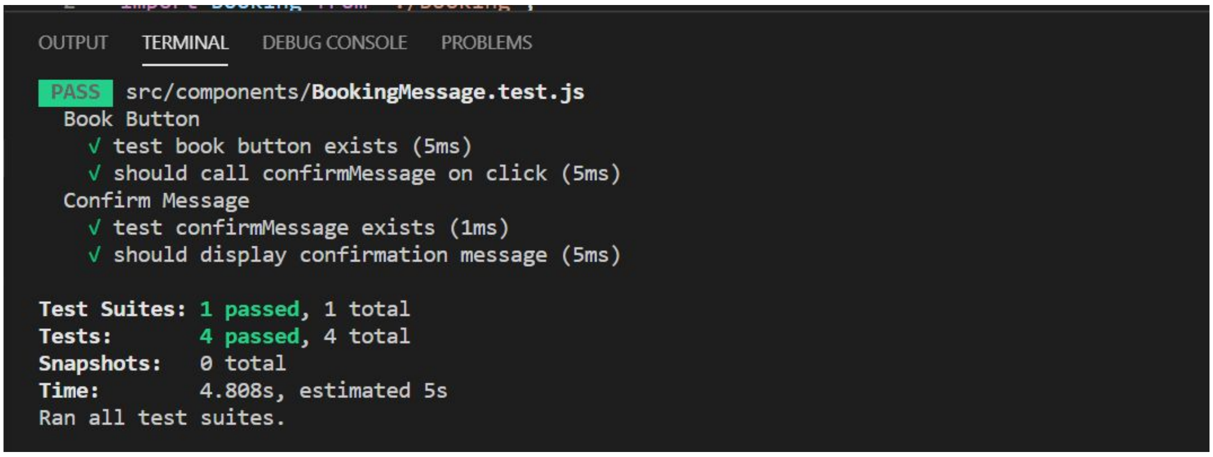
```
PASS  src/components/BookingField.test.js (7.275s)
  Booking Page
    ✓ should render Booking Page (9ms)
  DateField
    ✓ should change the date
  TimeField
    ✓ should change the time
  Worker
    ✓ should change the worker
  Service
    ✓ should change the service (1ms)
  Duration
    ✓ should change the duration
  Notes
    ✓ should change the notes

Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        9.739s
Ran all test suites related to changed files.

Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
 › 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.
```

BookingMessage.test.js

This test file contains 4 tests which test the book button on the booking page. The first two tests test that the book button exists on the page and that it calls the confirmMessage() function when clicked by the user. The other two tests test the confirmMessage() function and that when called, the function displays the confirmation message to the user.

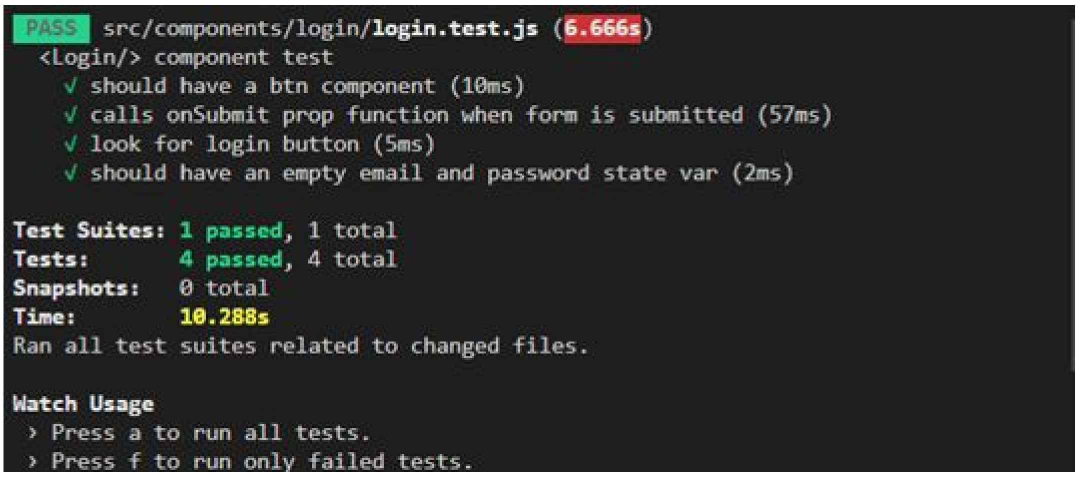


```
PASS  src/components/BookingMessage.test.js
  Book Button
    √ test book button exists (5ms)
    √ should call confirmMessage on click (5ms)
  Confirm Message
    √ test confirmMessage exists (1ms)
    √ should display confirmation message (5ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        4.808s, estimated 5s
Ran all test suites.
```

Login.test.js

This test suite contain 4 tests the first test look for the only one button on the login form the second test test whether the form has been submitted once the button is clicked.The third test whether did the button on login form have the matching text.The fourth test is to check if password and email are empty strings on setup.

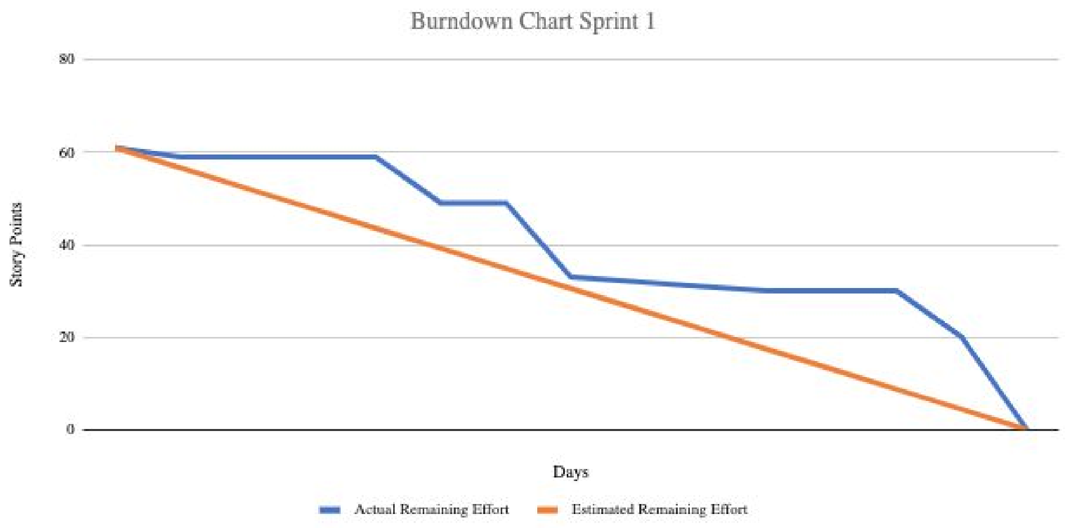


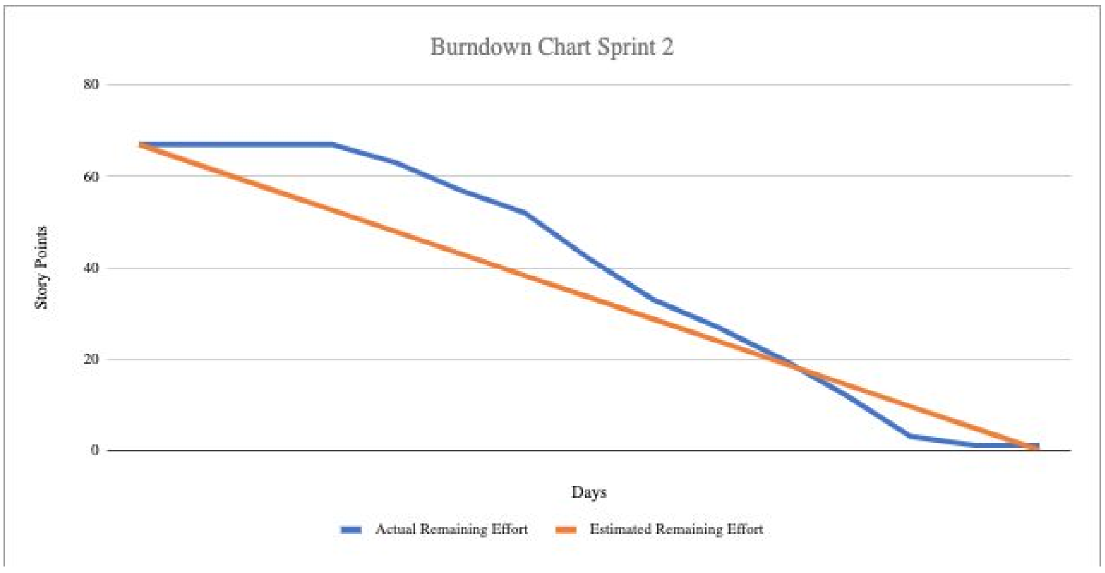
```
 PASS  src/components/login/login.test.js (6.666s)
  <Login/> component test
    √ should have a btn component (10ms)
    √ calls onSubmit prop function when form is submitted (57ms)
    √ look for login button (5ms)
    √ should have an empty email and password state var (2ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        10.288s
Ran all test suites related to changed files.

Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
```

Burndown Chart




Burndown Chart Sprint 2
80
60
40
20
0
Story Points
Days
Actual Remaining Effort
Estimated Remaining Effort