Monday 3:30, Group 1 – Milestone 2 Submission

Name	Student #	Contribution
Abdelmalek, Lawrence	s3656022	20%
Alsagheer, Abrar	s3707180	20%
Tawaf, Elizabeth	s3723812	20%
Yendall, Brodey	s3718834	20%
Wang, Richard	s3720241	20%

Contributions

Abdelmalek, Lawrence

- Assisted other front end developers with various tasks and designs
- · Developed tests for the front end
- · Finalising of Bookings and Hours services

Alsagheer, Abrar

- Development of the circleCI pipeline
- Deployments of front end and back end services
- · Development of the availability service

Tawaf, Elizabeth

- Implementation of admin create users functionality
- Prototyping and designing the interfaces

Yendall, Brodey

- Maintenance of back end services
- · Increasing and fixing backend tests
- · Refactor and improvements to the common library

Wang, Richard

- Implementation of admin assigning hours functionality
- Implement book appointment functionality
- · Investigating how to link back to front end

Links

Github:

https://github.com/RMIT-SEPT/majorproject-9-mon-1530-1

JIRA (Our task board):

https://sept-assignment.atlassian.net/secure/RapidBoard.jspa?rapidView=5&projectKey=AS

CircleCI:

https://app.circleci.com/pipelines/github/RMIT-SEPT/majorproject-9-mon-1530-1

Due to running out of credits we were required to make a private repository and developed the CircleCI pipeline there.

Please observe "CircleCI history.pdf" for screenshots of failure and successes from CircleCI and a screenshot the AWS deployed application.

AWS

Main link:

Use this link to access the application's front end http://sept-env-1.eba-mpb52g22.us-east-1.elasticbeanstalk.com/

Back-end URLs

The following URLs are all for backend services. These links do not have a front end interface but use an API instead. Please observe our API documentation for how to use these APIs Users:

http://users-env.eba-qkspuxrm.us-east-1.elasticbeanstalk.com

Hours:

http://hours-env.eba-mmev4jew.us-east-1.elasticbeanstalk.com/

Bookings:

http://bookings-env.eba-kxxpnwmy.us-east-1.elasticbeanstalk.com/

Availability:

http://availability-env.eba-5wumcyam.us-east-1.elasticbeanstalk.com/

Documentation

You can find our documentation at these links or alternative in the PDFs attached

Meeting minutes:

 $\underline{https://docs.google.com/document/d/1VwznRiQvTXG2edQyKUvhek1T3K1MYGgKsf5rYva7a2s/edit}$

Acceptance tests: https://docs.google.com/document/d/17i3kjH-Sq0F4iVErP9VUwW7l8hW05puLTY2r6v4cgBM/edit

Retro notes:

https://docs.google.com/document/d/1QzsRQ6jkXlGGtWvbo2BbgK1EYT63mxl7qIIjjKftesQ/edit

Sprint planning notes:

https://docs.google.com/document/d/1QsxZ 178HfUrBCNNuHI8zo83-UsvzTBsUG0kFy5tYKI/edit

User stories:

https://docs.google.com/document/d/1hk5YE-FuwBdbOxs24yind3ak 6VOdh7edNO53AaewBA/edit

Additional documentation

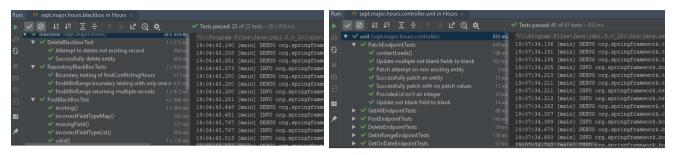
API documentation: https://docs.google.com/spreadsheets/d/11sLVYMP3aJ-I-ocQJIMQttMxrNts48WPDepV-bfG4DY/edit#gid=948496785

Setting up a basic CRUD service with the common library:

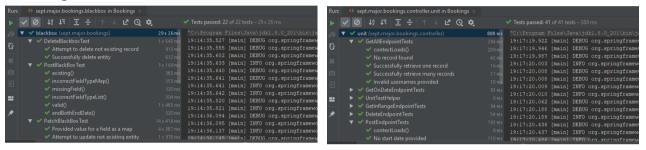
https://docs.google.com/document/d/1wkaIza8SGxosywtLCUmT8agkewFyaA1RV939RXSbK2g/edit#

Testing evidence:

Hours



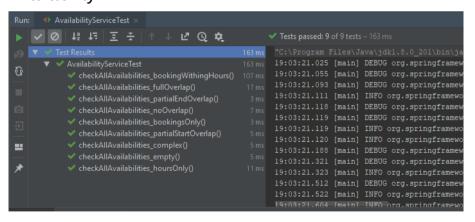
Bookings



Users



Availability



Front end

```
Nusage
ress a to run all tests.
ress of to run only failed tests.
ress of to quit watch mode.
ress of to quit watch mode.
ress of to filter by a filename regex pattern.
ress to filter by a test name regex pattern.
ress to filter by a test name regex pattern.
ress filter to trigger a test run.
ress filter to trigger a test run.
ress filter to trigger a test run.
ress filter to trigger a fest run.
re
              Unable to find an element with the text: 13/01/2022. This could be because the text is broken up by multiple elements. In this case
                                             />
</svg>
</span
class="sc-fzoCCn gIHccX"
                                                               expect(getByText('testWorker')).toBeTruthy();
expect(getByText('13/01/2022')).toBeTruthy();
                                                             expect(getByText('2:00:00 pm')).toBeTruthy();
expect(getByText('3:00:00 pm')).toBeTruthy();
                   at Object.getElementError (node_modules/@testing-library/dom/dist/config.js:34:12) at node_modules/@testing-library/dom/dist/query-helpers.js:71:38 at getByText (node_modules/@testing-library/dom/dist/query-helpers.js:54:17) at Object.<anonymous> (src/Components/Dashboard/DashboardComponents.test.js:64:14)
   FAIL src/Components/Form/Form.test.js
• Test suite failed to run
              Cannot find module '../Components/Form' from 'Form.test.js'
                                    import 'mutationobserver-shim';
import { render } from '@testing-library/react';
import Form from '../Components/Form';
                    at Resolver.resolveModule (node_modules/jest-resolve/build/index.js:259:17) at Object.<anonymous> (src/Components/Form/Form.test.js:4:1)
              src/Components/Toolbar/Toolbar.test.js
src/Components/Main/Main.test.js
Console
               console.warn node_modules/react-dom/cjs/react-dom.development.js:88
Warning: componentWillReceiveProps has been renamed, and is not recommended for use. See https://fb.me/
                   * Move data fetching code or side effects to componentDidUpdate.

* If you're updating state whenever props change, refactor your code to use memoization techniques or mc

* Rename componentWillReceiveProps to UNSAFE_componentWillReceiveProps to suppress this warning in non-sreact-codemod rename-unsafe-lifecycles' in your project source folder.
    PASS src/App.test.js

• Console
              console.warn node_modules/react-dom/cjs/react-dom.development.js:88
Warning: componentWillReceiveProps has been renamed, and is not re

* Move data fetching code or side effects to componentDidUpdate.
                  * If you're updating state whenever props change, refactor your code to use memoization techniques or mo

* Rename componentWillReceiveProps to UNSAFE_componentWillReceiveProps to suppress this warning in non-seact-codemod rename-unsafe-lifecycles` in your project source folder.
Test Suites: 2 failed, 7 passed, 9 total
Tests: 1 failed, 24 passed, 25 total
Snapshots: 0 total
Time: 9.558s
```

Work done since Milestone 1

- Implementation of availability service MVP
 - The availability service is a convenience API for front end that formats Hours and Bookings data into a single payload. Saving the front end from doing extra calculations
 - Back/Availability directory contains the project for this service.
- · Finalisation of the Bookings and Hours services
 - Changes the controllers, services, repositories and junit tests. Each of these can be found in the relevant BackEnd folders for each service
 - As of milestone 1 the Bookings and Hours services were barebones and have bugs a
 plenty. This milestone we finalised these services by completing the functionality, fixing
 the bugs and a great improvement to the testing
- Blackbox testing with a temporary dockerized database
 - Allows for junit tests for backend services to access a database that will be closed and reopened for every test. This functionality allows us to do complex and isolated tests without the use of mocks, allowing us to test the full scope of the application.
 - BackEnd\Common\src\main\java\sept\major\common\testing for all classes used in the implementation of this.
 - Bookings, Hours and Users implement an application-blackbox.properities file and include a startup .sql file in the resources folder relevant to the service.
- · Improvements to User service tests
 - The user service lacked any sort of blackbox testing. This implemented this functionality as well as general improvements to the unit tests.
 - See BackEnd\Users\src\test\java\sept\major\users
- Implement book appointment functionality
 - User story 8
 - In order to accurately handle requests sent to the back-end, we must ensure that every request sent has all the information required. There is no point posting a query to the back-end without the correct parameters. The requests should be handled client-side.
 - Support Firefox date and time selection for making a booking ensure that our application runs smoothly on major supported browsers so that any user can successfully use the app. We need to ensure all components we use are supported, and where they are not, we implement workarounds to address these issues.
 - Handle return status from booking an appointment for the user dashboard, we can start sending requests, which is great and all, but how do we know a request worked? This story will explore updating the UI and view to reflect whether a request made by a user was successful or not.
 - Changes are in FrontEnd/src/Components/Dashboard/Bookings/BookingComponents.js
 - FrontEnd/src/Components/Dashboard/DashboardComponents.js
 - FrontEnd/src/Components/Dashboard/Dashboard.js
- Linking back end to the front
 - We have decided that axios is the best option to link our react based front end to our Java backend.
 - This was don easily by adding the axios library to our react library
 - In order to post or get data; we make an axios object that identifies if it is a get or a post request, then in that method we pass in the data to the backend if we are doing a post request.
 - If we need to make a get request we pass in the key id for the specific data base
- · Admin create worker
 - User story 7
 - Implement a page that allows an admin user to create a new employee, linking with relevant back end services. This involves creating a component to display a form to capture employee information, and when the form is submitted, it will send a POST request to /user.
 - This contains the Amin page /FrontEnd/src/Components/Dashboard/Admin/Admin.js

- This contains a Form that could be reused to signup the user /FrontEnd/src/Components/Form/FormDetails.is
- Admin assign hours to work
 - User story 3
 - Allow an admin to assign hours to an employee, linking with relevant back end service.
 - Have the UI to allow the Admin to set hours for workers.
- Implemented front end testing
 - Add unit tests for Dashboard/DashboardComponents
 - Add unit tests for Booking/BookingComponents
 - Update front-end tests for current page set (sign up)
 - Add unit tests for static front-end pages
 - o Add unit tests for Login
- Implemented circleCI pipeline
 - The circleCI pipeline allows for testing and evaluating commits to the github repository, increasing the quality of code that is merged into develop and consequently release and master.
 - See the circleCI pipeline link given above
- Deployed services to AWS
 - We now have the front end and back end services deployed to AWS so that they may be accessed without setting up in your local environment
 - See AWS links provided above
- Moved to google sheets for database documentation
 - Our previous API documentation was done with Swagger however this was too difficult to maintain and read so we migrated to a google sheet.
 - https://docs.google.com/spreadsheets/d/11sLVYMP3aJ-I-ocQJIMQttMxrNts48WPDepVbfG4DY/edit?usp=sharing