# HOBBY WEB APPLICATION PROJECT RESTAURANT RATING API

BY PHILIP UGONA

# ABOUT ME

- Studied Mechanical Engineering (Undergraduate) – Graduated in 2018
- Studied Engineering Business Management (Masters) – Graduated in 2020
- Went to the University of Sussex
- This course is  drastic change in both structure and workload to my previous education

# MY APPROACH TO THE PROJECT

- Outlined the basic tasks and requirements from the project specification using project management software Notion.
- Brainstormed potential projects and weighed my options against the project spec. Options included Stock Price and Earnings Tracker,  Gym Workout Logger, and Restaurant Meal Ranking System.
- Selected the Restaurant Meal Ranking System as it aligns more closely to the Minimum Viable Product.
- Upon selecting the project, tasks were split these tasks into three categories: Front-End, Back-End and Testing.
- The Front-End was attempted first, and the structure of the API was drafted (using HTML) and some styling options were added (using CSS)

# SPRINT PLAN (RESTAURANT MEAL RATING)

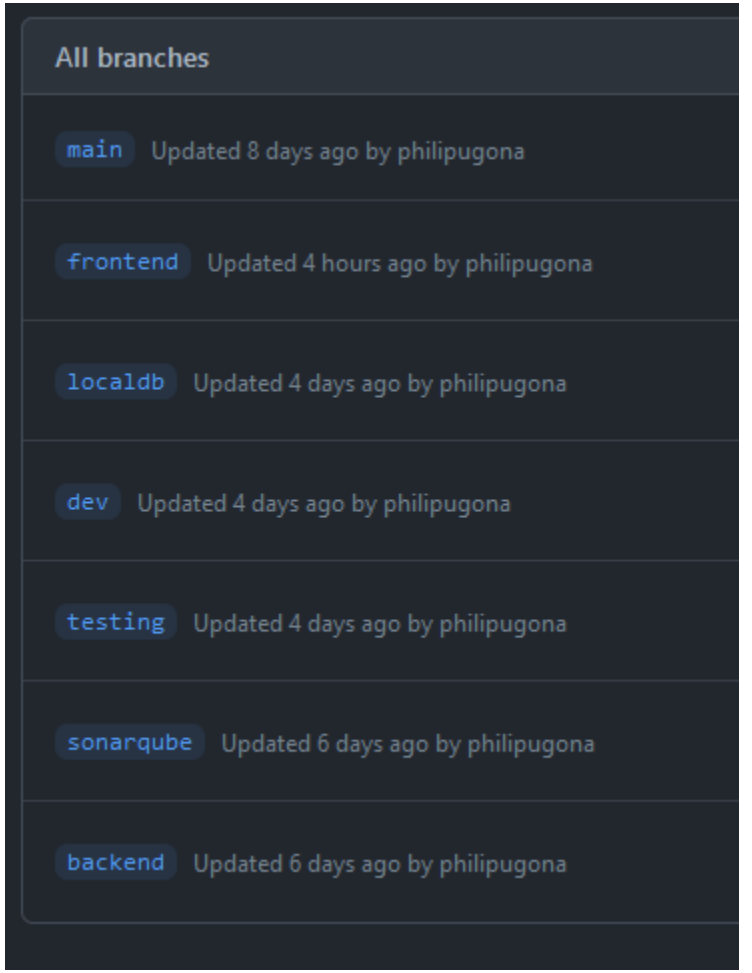
- User Stories were created for the project forming the Backlog
- These user stories were assigned story points, a priority and sub-tasks
- Story points are time estimations for the completion of each user story.
- Priority is shown the arrows on the right of each task. High priority arrows are red, medium priority arrows are orange and low priority arrows are green
- The high priority task were carried out first, then the medium priority tasks
- Low priority tasks are additional features to be added, after the MVP has been fulfilled

# TECHNOLOGIES LEARNT AND APPLIED

- JIRA – Jira is a collaboration tool used in Agile Teams for project planning and task delegation.
- JAVASCRIPT – Java is a general purpose, class based, object orientated programming language.
- MYSQL – A database manipulation language. Essentially used to store large amounts of data. Used in this project to store all the restaurant meals, restaurant location, the dish ordered and the rating of the dish.
- ECLIPSE IDE – Eclipse is an integrated development environment (IDE) for developing applications using the Java programming language and other programming languages. Created A Spring Java Project for the back-end of the project. Testing is also carried out in Eclipse.
- GITHUB & GITBASH– GitHub is a collaboration and Version control tool. Git Bash is used to set up a local repository for version control.
- HTML & CSS – Used to create the structure of the API and add styling to it.
- Selenium – Used to automate tests for web browsers.

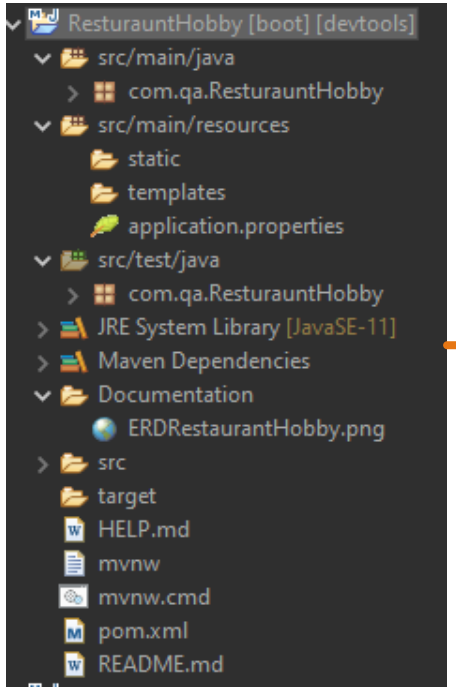
# VERSION CONTROL

- Ensured that with each new feature, a new branch was instantiated
- Each added feature was tested to ensure it was working before making commits.
- Regular commits were being made in the appropriate branches
- Previous commited featured were also testing thourougly to ensure no bugs were encountered which would delay the project.

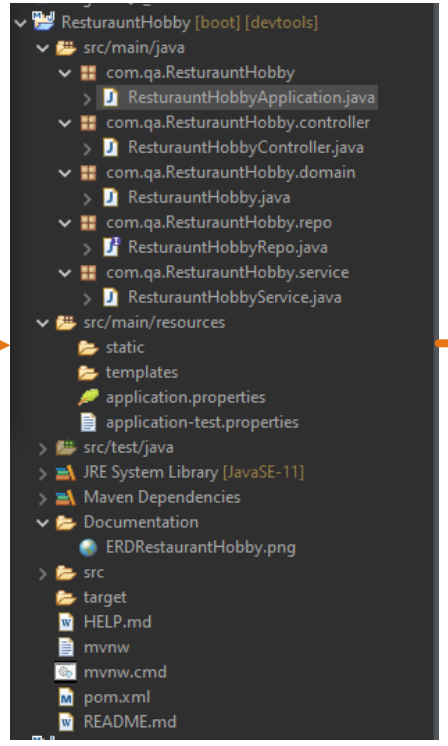


# PROJECT GROWTH

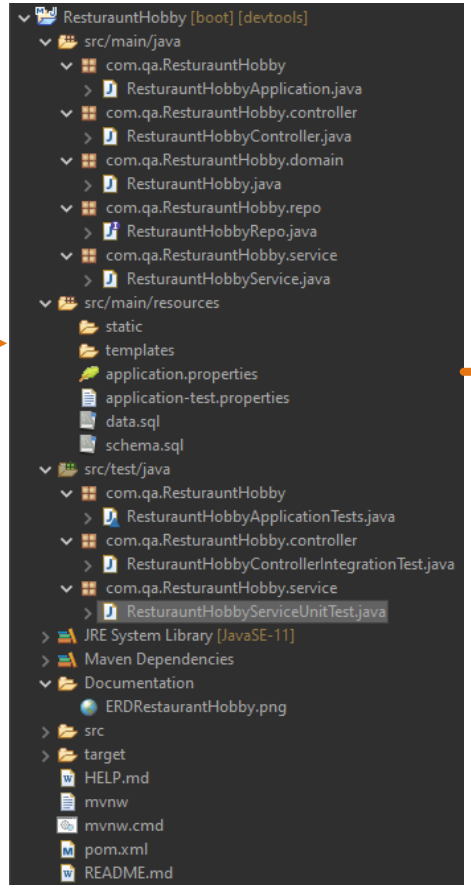
## Start



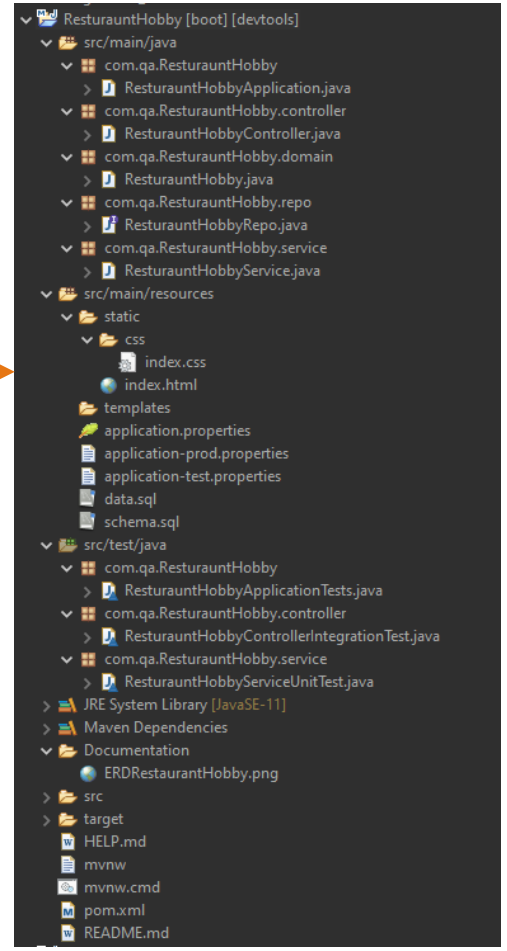

## Back-End and Testing




## Added Front-End




## Current



# TESTING

- Testing for this project required three types of automated tests:
- Unit Testing: These tests are carried out using Junit with the Maven system in Java Eclipse. A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behaviour or state
- Integration Tests: Carried out using Junit in Java Eclipse also. An integration test aims to test the behaviour of a component or the integration between a set of components. Essentially, these tests covert the user stories into tests.
- Web Application Testing: Using Selenium WebDriver to automate web browsers to test website functionality. Carried out also through Java Eclipse.

# TESTING COVERAGE

ResturauntHobby (10 May 2021 17:32:49)

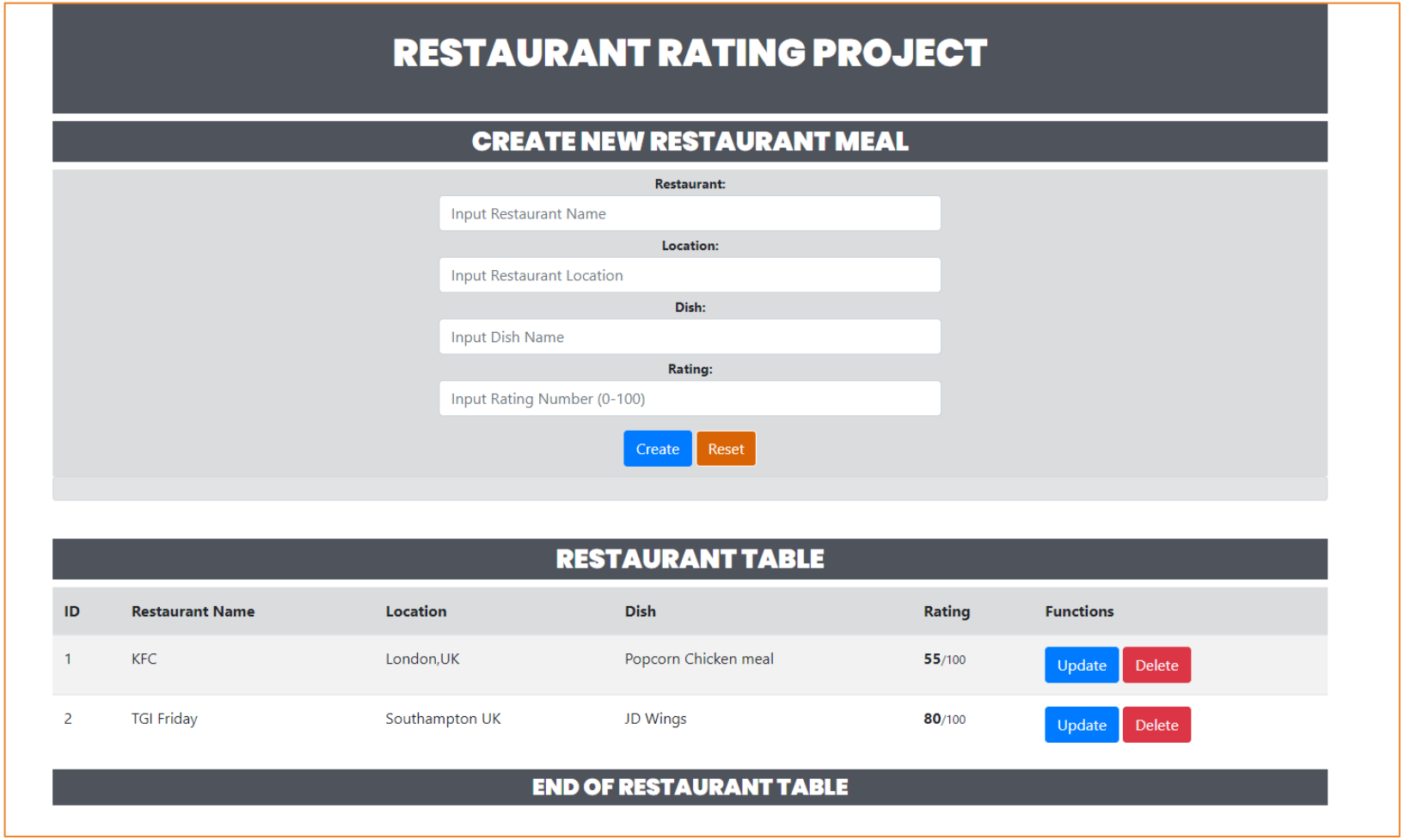
| Element | Coverage | Covered Instructions | Missed Instructions | Total Instructions |
|---|---|---|---|---|
| ResturauntHobby | 86.0 % | 751 | 122 | 873 |
| src/main/java | 67.7 % | 254 | 121 | 375 |
| com.qa.ResturauntHobby.domain | 53.9 % | 131 | 112 | 243 |
| com.qa.ResturauntHobby | 37.5 % | 3 | 5 | 8 |
| com.qa.ResturauntHobby.service | 95.2 % | 80 | 4 | 84 |
| com.qa.ResturauntHobby.controller | 100.0 % | 40 | 0 | 40 |
| src/test/java | 99.8 % | 497 | 1 | 498 |

ResturauntHobbyService.java | application.properties | ResturauntHobbyController

```java
@Service
public class ResturauntHobbyService {

    private ResturauntHobbyRepo repo;

    public ResturauntHobbyService(ResturauntHobbyRepo repo) {
        this.repo = repo;
    }

    public ResturauntHobby create(ResturauntHobby r) {
        return this.repo.save(r);
    }

    public List<ResturauntHobby> getAll() {
        return this.repo.findAll();
    }

    // getById needed to delete or update a field in the table
    public ResturauntHobby getById(Long id) {
        Optional<ResturauntHobby> optionalDish = this.repo.findById(id);
        if(optionalDish.isPresent()) {
            return optionalDish.get();
        }
        return null;
    }

    //Delete
    public boolean remove(Long id) {
        this.repo.deleteById(id);
        boolean exists = this.repo.existsById(id);
        return !exists;
    }

    //Update
    public ResturauntHobby update(Long id, ResturauntHobby rest) {
        Optional<ResturauntHobby> existingOptional = repo.findById(id);

        if(existingOptional.isPresent()) {
            ResturauntHobby existing = existingOptional.get();

            existing.setId(id);
            existing.setDish(rest.getDish());
            existing.setLocation(rest.getLocation());
            existing.setRating(rest.getRating());
            existing.setRestaurantName(rest.getRestaurantName());

            return this.repo.save(existing);
        }
        return null;
    }
```

- Target Test coverage of 80%. This includes both Integration and Unit Testing.
- Current Unit Test coverage of 86%.
- Continually faced with the same error and not being able to troubleshoot
- The Service Class coverage can be improved to attain 100%

# DEMONSTRATION



# SPRINT REVIEW AND RETROSPECTIVE

- The Project went  well and stuck to the schedule.
- Estimation of the User Stories were good. The project did not fall behind schedule.
- The last CRUD function Update needs to be finalized
- The MVP will be met once the above is complete.
- Extra functionality planned at the start of the project cannot be completed.

# CONCLUSION / REFLECTION

- The code needs some tweaks to be fully functional for the deadline. The HTTP methods are not completed yet. Only the PUT request (the update CRUD function) is required for project completion.
- The POST, GET, and DELETE HTTP methods are functional but can also be improved.
- Back-End testing achieved 86% coverage which is acceptable.
- The approach taken for the Front-End should have been simplified, utilising a Div-based layout as opposed to a Table-based layout to display the MySQL data.
- The API will work as intended by the deadline