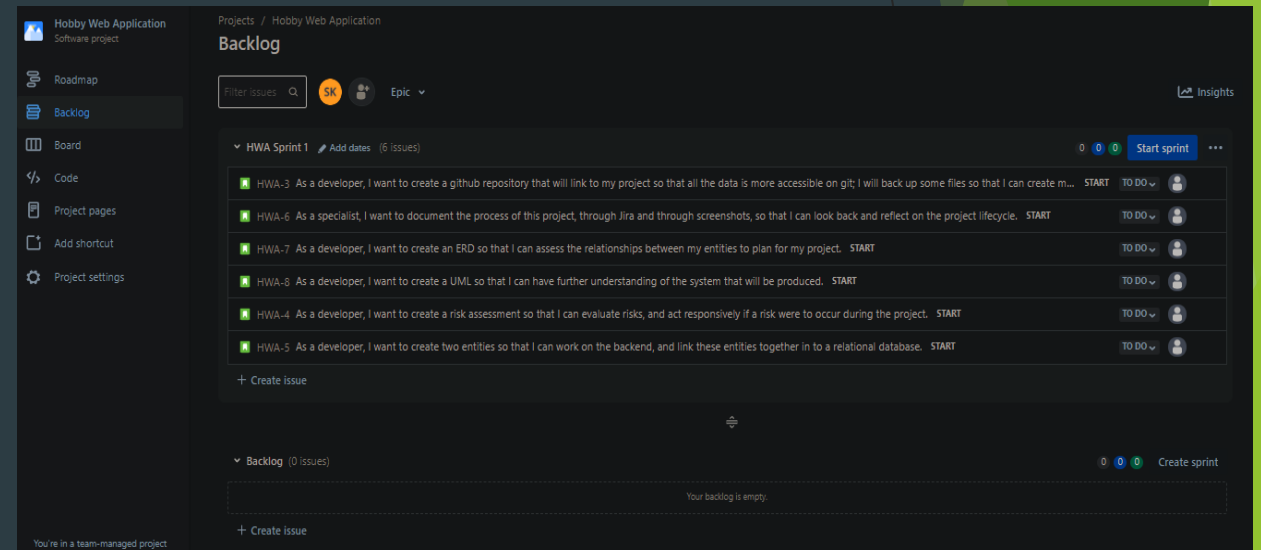
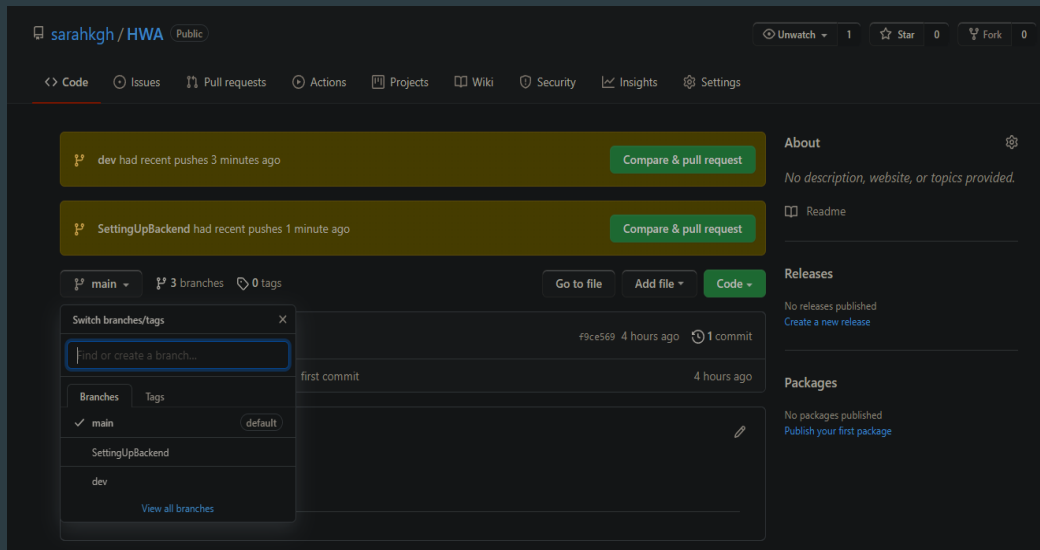


HWA project

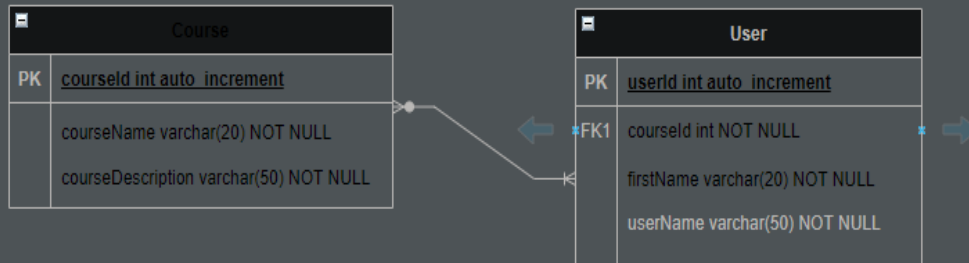
Sarah Khan

Introduction

- ▶ For this project, I approached the specification by:
- ▶ Setting user stories for all tasks within Jira.
- ▶ Preparing the GitHub repository so all the code can be kept together.
- ▶ Reading through and understanding the scope and MVP.



- I prepared ERD and UML diagrams for the project so that I was able to visual the application.



ERD Diagram

UML Diagram



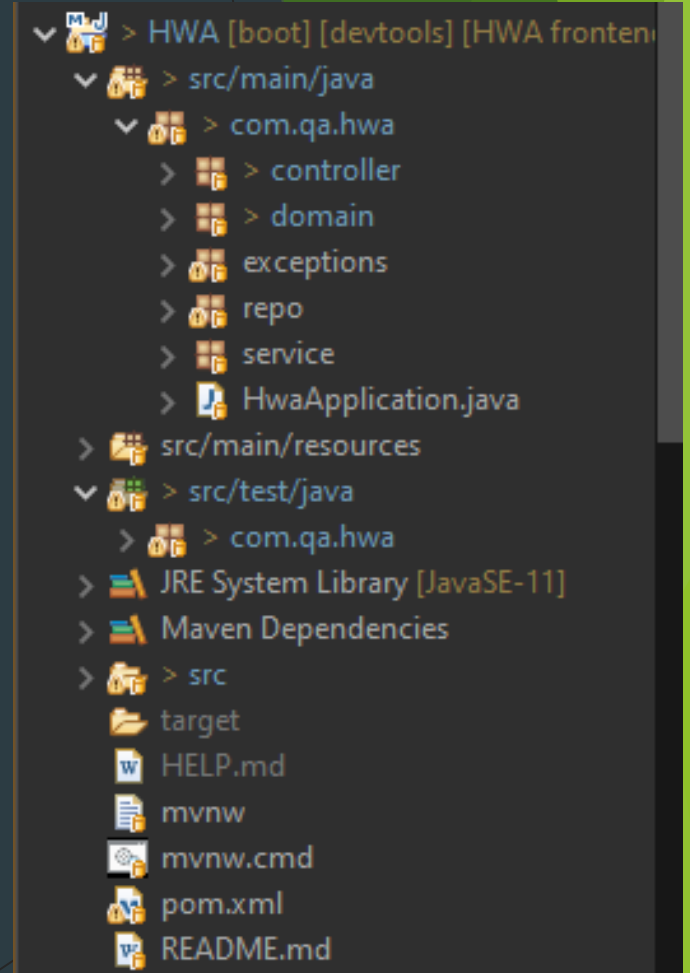
- ▶ I first decided to work on the backend of the application so that I would have some code I could base from. First I created the domain then the repo, service and controller classes.

User.java X

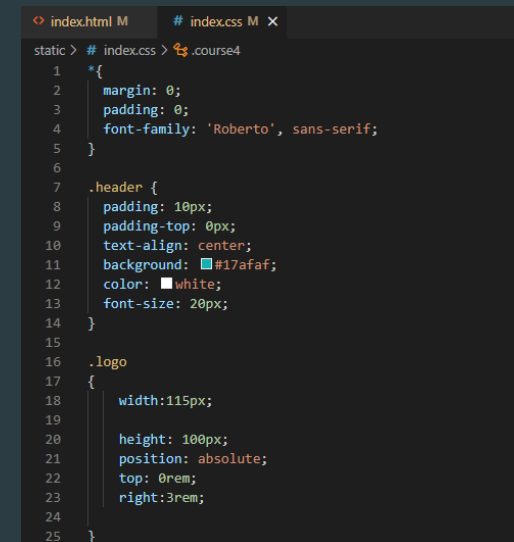
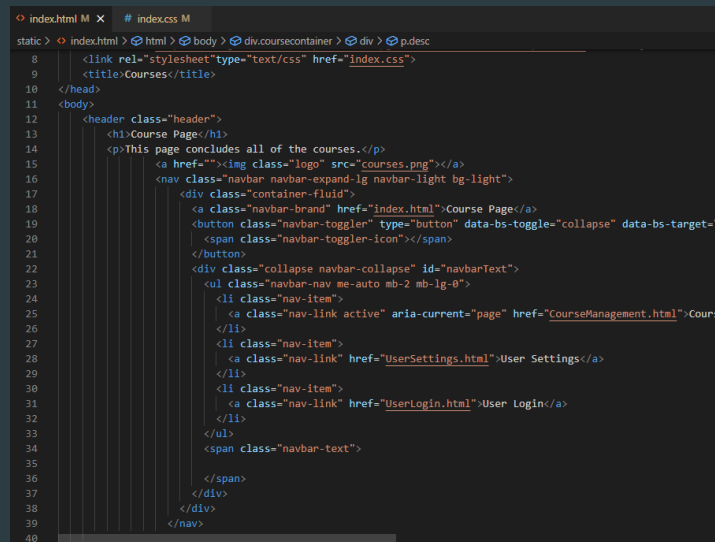
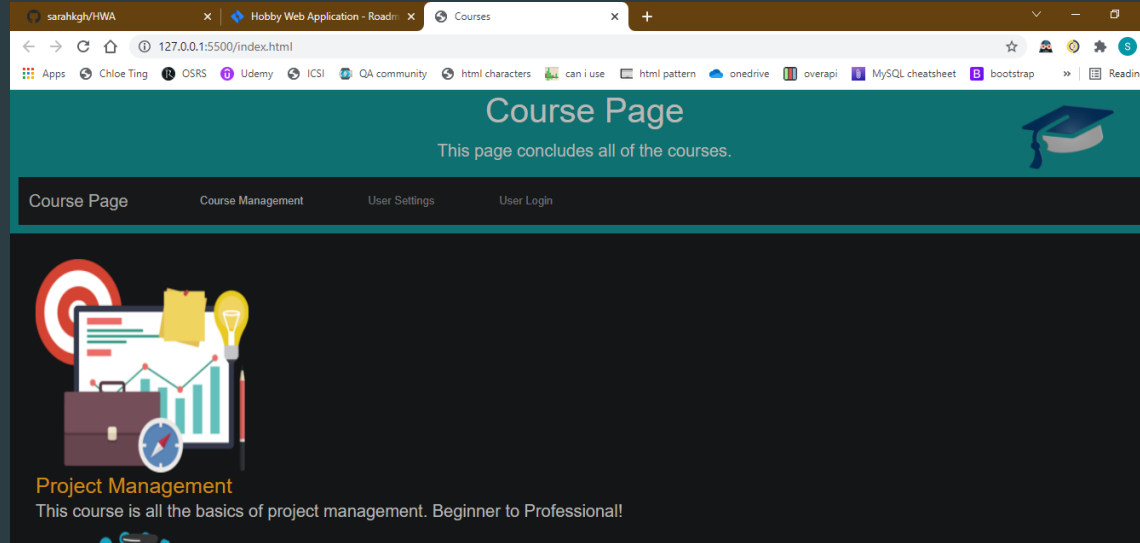
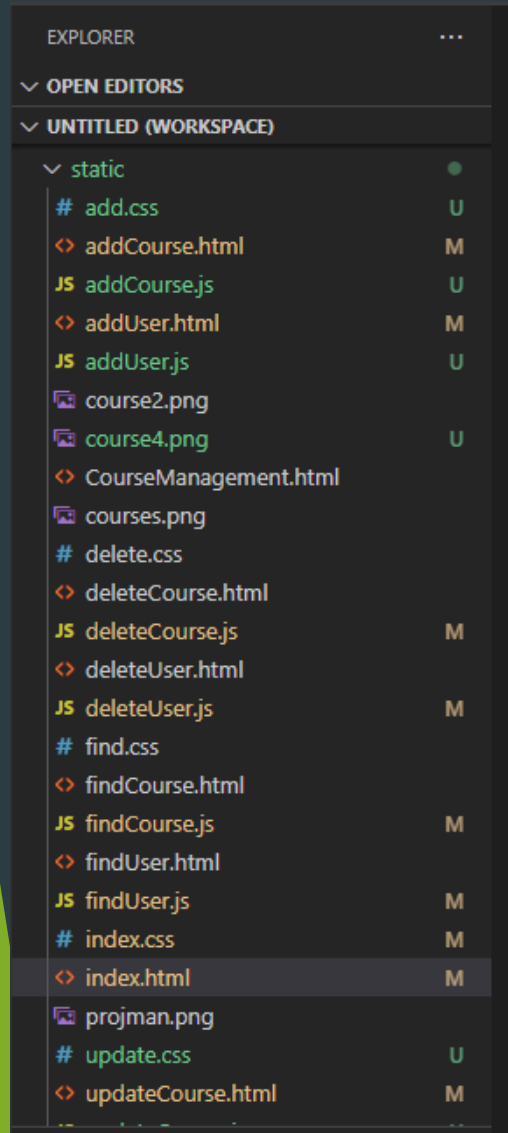
```
1 package com.qa.hwa.domain;
2
3 import javax.persistence.Column;
4
5
6
7
8
9
10
11 @Data
12 @Entity
13 public class User {
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private int userId;
18
19     @Column(name = "firstName", nullable = false)
20     private String firstName;
21
22     @Column(nullable = false, unique = true)
23     private String userName;
24
25     public User(String firstName, String userName) {
26         this.firstName = firstName;
27         this.userName = userName;
28     }
29
30 }
31
32 --
```

Course.java X

```
1 package com.qa.hwa.domain;
2
3 import javax.persistence.Column;
4
5
6
7
8
9
10
11 @Data
12 @Entity
13 public class Course {
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private int courseId;
18
19     @Column(name = "courseName", nullable = false, unique = true)
20     private String courseName;
21
22     @Column(nullable = false)
23     private String courseDescription;
24
25     public Course(String courseName, String courseDescription) {
26         this.courseName = courseName;
27         this.courseDescription = courseDescription;
28     }
29
30 }
31
```



- ▶ I then worked on the front end of my application, first designing a main (index) html page then many pages branching from. There is consistency within all pages of the website. Css and Js pages were created for the html pages.



As I was less confident with front end coding, I found help online and from my tutor to create some of the elements.

- Using GitHub, I could continuously back up all the changes I made to any code and these could be accessed at any time. I used the same commands on GitBash to regularly back up.



```
MINGW64/c/Users/Sarah/eclipse-workspace/HWA
modified: src/main/resources/static/updateCourse.html
modified: src/main/resources/static/updateUser.html
modified: src/test/java/com/qa/hwa/controller/UserControllerIntegrationTest.java



Untracked files:
(use "git add <file>..." to include in what will be committed)
src/main/resources/static/add.css
src/main/resources/static/addCourse.js
src/main/resources/static/addUser.js
src/main/resources/static/course4.png
src/main/resources/static/update.css
src/main/resources/static/updateCourse.js
src/main/resources/static/updateUser.js
src/test/java/com/qa/hwa/controller/CourseControllerUnitTest.java
src/test/java/com/qa/hwa/service/CourseServiceTest.java











no changes added to commit (use "git add" and/or "git commit -a")
g
Sarah@LAPTOP-NV4VFBK MINGW64 ~/eclipse-workspace/HWA (frontend)
$ git add .
Sarah@LAPTOP-NV4VFBK MINGW64 ~/eclipse-workspace/HWA (frontend)
$ git commit -m "HWA-13 #Updates to frontend"
```

Projects / Hobby Web Application

Backlog

Filter issues SK  Epic 

▼ HWA Sprint 1  (5 issues) 0 0 0 [Start sprint](#) 







-  HWA-4 As a developer, I want to create a risk assessment so that I can evaluate risks, and act responsively ... START TO DO 
-  HWA-9 As a developer, I want to work on the backend of my application so that I can ensure the front e... MIDDLE TO DO 
-  HWA-10 As a developer, I want to test the code within my backend so that I can ensure my methods are ... MIDDLE TO DO 
-  HWA-11 As a developer, I want to create the code for the front end of my application so that users can i... MIDDLE TO DO 
-  HWA-13 As a developer, I want to create the JavaScript pages in the front end of my application so that I... MIDDLE TO DO 


[+ Create issue](#)

▼ Backlog (0 issues) 0 0 0 [Create sprint](#)

Your backlog is empty.


[+ Create issue](#)


 Middle /   1    [Insights](#)

To Do 

Description

Add a description...

Details 


Assignee  Unassigned


Labels None

Sprint HWA Sprint 1

Story point estimate None

Development 2 commits 1 minute ago

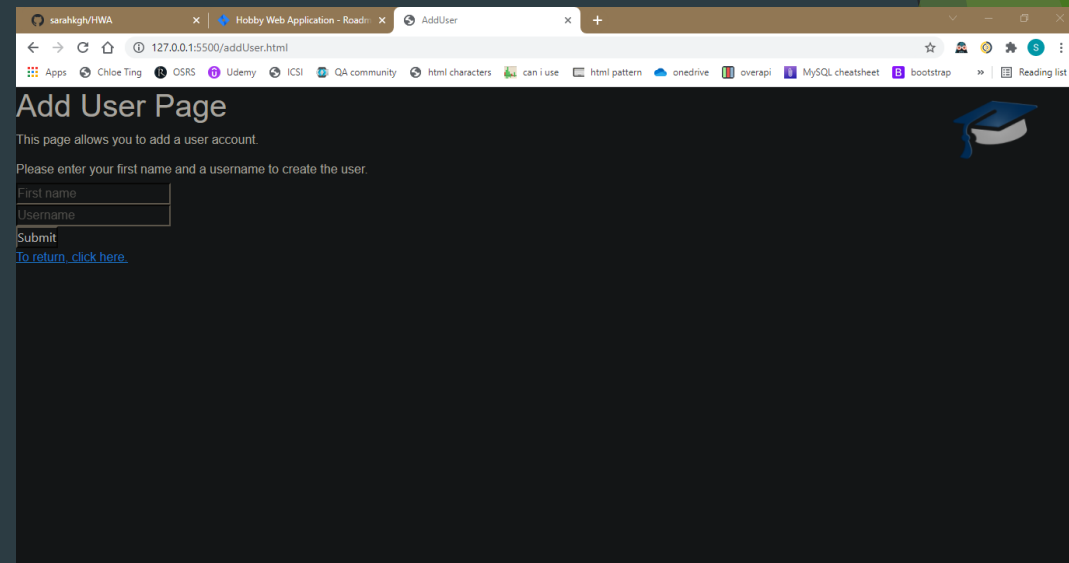
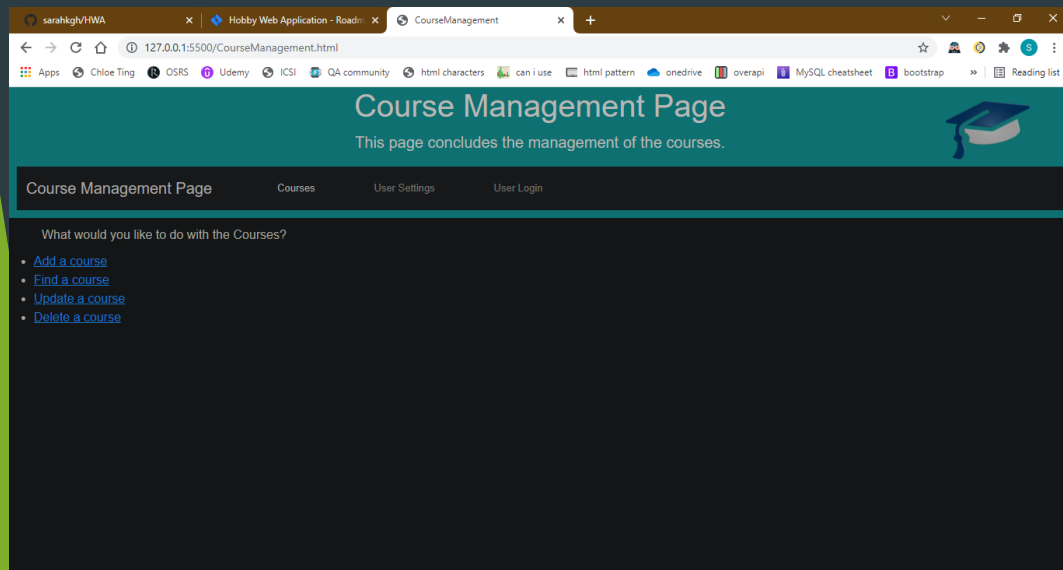
Reporter  Sarah Khan



Pro tip: press **IM** to comment

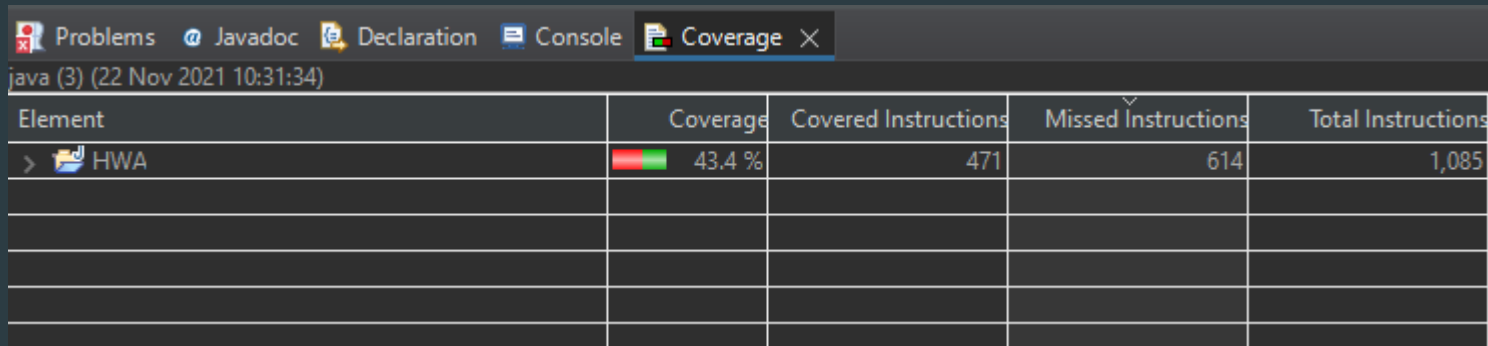
Lessons learnt

- ▶ During my consultant journey, I learnt to:
- ▶ Become more confident with the frontend (html, css, js) of the application.
- ▶ More familiarity with service, controller and repo classes.
- ▶ More understanding of tests.
- ▶ Being able to use postman.
- ▶ Work on a Spring Application.



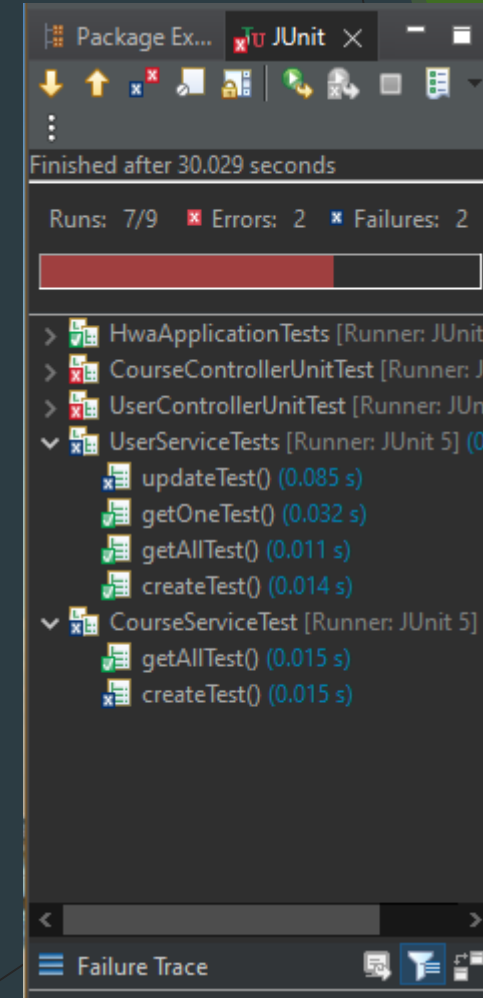
Testing

- Methods within the service and controller classes were run and tested to ensure that they are functioning.



The screenshot shows the Eclipse IDE's Coverage view. The top bar includes tabs for Problems, Javadoc, Declaration, Console, and Coverage. Below the tabs, it says 'java (3) (22 Nov 2021 10:31:34)'. The main area is a table with the following columns: Element, Coverage, Covered Instructions, Missed Instructions, and Total Instructions. The table contains one row for 'HWA' with a coverage of 43.4%, 471 covered instructions, 614 missed instructions, and a total of 1,085 instructions.

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
> HWA	43.4 %	471	614	1,085



The screenshot shows the Eclipse IDE's JUnit test runner window. The title bar says 'Package Ex... JUnit'. The window displays the following information: 'Finished after 30.029 seconds', 'Runs: 7/9', 'Errors: 2', and 'Failures: 2'. Below this is a progress bar. The test results are listed in a tree view: 'HwaApplicationTests [Runner: JUnit]', 'CourseControllerUnitTest [Runner: JUnit]', 'UserControllerUnitTest [Runner: JUnit]', 'UserServiceTests [Runner: JUnit 5] (0.015 s)', and 'CourseServiceTest [Runner: JUnit 5] (0.015 s)'. The 'UserServiceTests' and 'CourseServiceTest' are expanded, showing their respective test methods: 'updateTest() (0.085 s)', 'getOneTest() (0.032 s)', 'getAllTest() (0.011 s)', 'createTest() (0.014 s)', 'getAllTest() (0.015 s)', and 'createTest() (0.015 s)'. The bottom of the window has a 'Failure Trace' button and some icons.

Package Ex... JUnit

Finished after 30.029 seconds

Runs: 7/9 Errors: 2 Failures: 2

HwaApplicationTests [Runner: JUnit]

CourseControllerUnitTest [Runner: JUnit]

UserControllerUnitTest [Runner: JUnit]

UserServiceTests [Runner: JUnit 5] (0.015 s)

- updateTest() (0.085 s)
- getOneTest() (0.032 s)
- getAllTest() (0.011 s)
- createTest() (0.014 s)

CourseServiceTest [Runner: JUnit 5] (0.015 s)

- getAllTest() (0.015 s)
- createTest() (0.015 s)

Failure Trace

Sprint review

- ▶ What went well: created two entities and their methods, created a consistent front end for the application, regularly used github to backup, a Jira storyboard to keep up with user stories.
- ▶ What could be improved: integration testing, a more professional frontend.
- ▶ The HWA has no errors and the application layers all relate to each other. It could be improved with refinement to the Junit testing, extra loggers to the console and further HTML elements to improve the frontend.