Automated Testing document

Test Report

Thomas van der Molen
IPS3-DB03

Project Information	
Project members	Thomas van der Molen (4168003)
Project name	Text Adventure
Version	1.0

Table of Contents

Version History

Version	Date	Change
1.0	17-01-2022	Created File

Introduction

The Text Adventure project is split into 3 services: The front-end service, the back-end entity manager, and the back-end game manager. All services have been tested on their most important functionalities / interactions.

Front-end

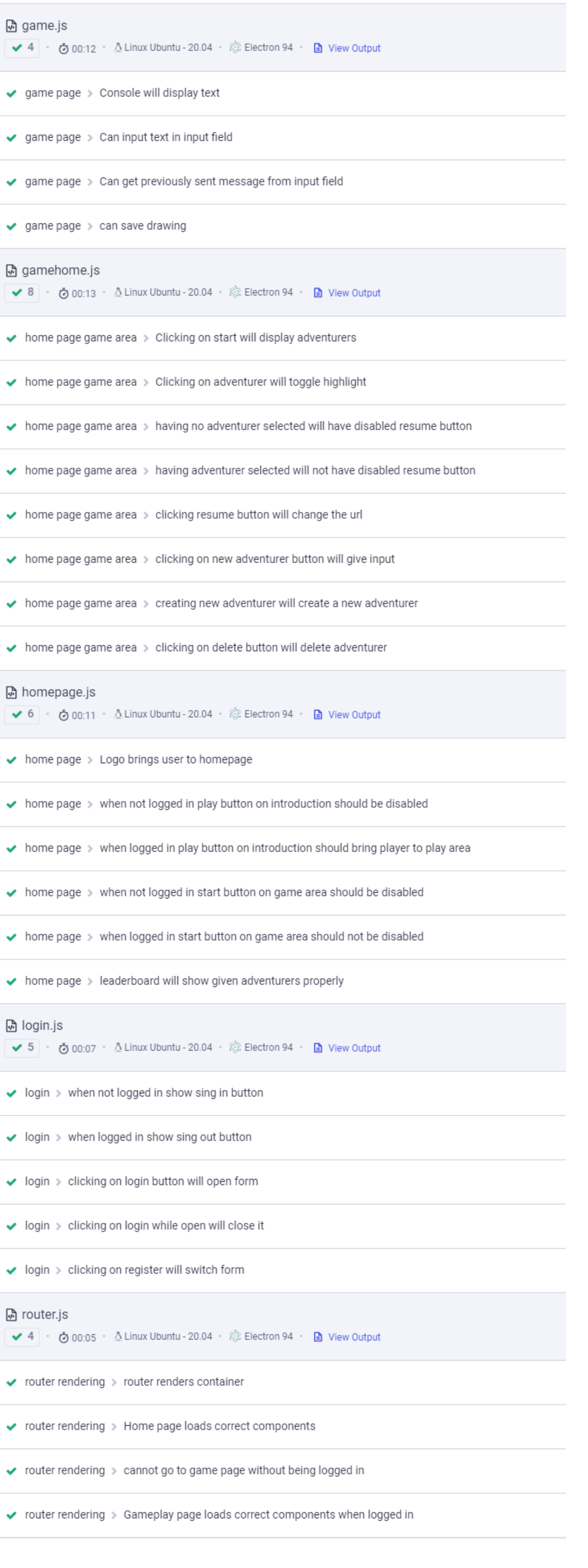
27 tests

For my front-end React App, I have decided to use Cypress to test my logic, I have also decided to write my tests as integration tests with no dependencies on other running services (also because my GP frontend tests are exclusively end-to-end tests).

Because I test all my services and I think having tests rely on as little outside dependencies as possible, I have decided to test my front-end in this structure, which does mean that any interaction like logging in will have to be mocked.

For my testing coverage I have made sure to cover all routes and the basic interactions on set routes, this covers the homepage and game page and any user input possible on set pages, such as logging in, drawing, creating an adventurer, etc.

Furthermore, I have connected my CI/CD to with cypress.io so that all ran tests will be documented on there, including videos and images of the tests.



game.js
✓ 4 · 00:12 · Linux Ubuntu - 20.04 · Electron 94 · View Output

✓ game page > Console will display text
✓ game page > Can input text in input field
✓ game page > Can get previously sent message from input field
✓ game page > can save drawing

gamehome.js
✓ 8 · 00:13 · Linux Ubuntu - 20.04 · Electron 94 · View Output

✓ home page game area > Clicking on start will display adventurers
✓ home page game area > Clicking on adventurer will toggle highlight
✓ home page game area > having no adventurer selected will have disabled resume button
✓ home page game area > having adventurer selected will not have disabled resume button
✓ home page game area > clicking resume button will change the url
✓ home page game area > clicking on new adventurer button will give input
✓ home page game area > creating new adventurer will create a new adventurer
✓ home page game area > clicking on delete button will delete adventurer

homepage.js
✓ 6 · 00:11 · Linux Ubuntu - 20.04 · Electron 94 · View Output

✓ home page > Logo brings user to homepage
✓ home page > when not logged in play button on introduction should be disabled
✓ home page > when logged in play button on introduction should bring player to play area
✓ home page > when not logged in start button on game area should be disabled
✓ home page > when logged in start button on game area should not be disabled
✓ home page > leaderboard will show given adventurers properly

login.js
✓ 5 · 00:07 · Linux Ubuntu - 20.04 · Electron 94 · View Output

✓ login > when not logged in show sing in button
✓ login > when logged in show sing out button
✓ login > clicking on login button will open form
✓ login > clicking on login while open will close it
✓ login > clicking on register will switch form

router.js
✓ 4 · 00:05 · Linux Ubuntu - 20.04 · Electron 94 · View Output

✓ router rendering > router renders container
✓ router rendering > Home page loads correct components
✓ router rendering > cannot go to game page without being logged in
✓ router rendering > Gameplay page loads correct components when logged in

Back-end

For my back-end I have used Xunit to write my tests, in combination with Moq and Shouldly. For my testing I have decided to use both unit tests and integration tests. I have primarily focused on writing integration tests because these cover more code while taking less time.

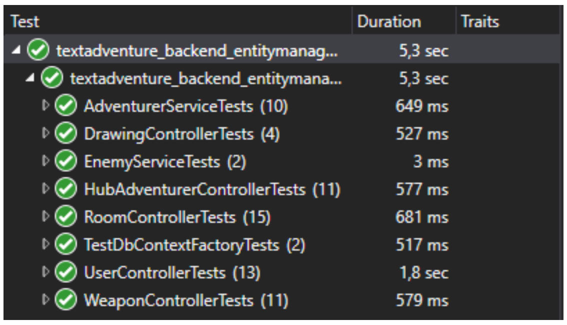
Entity Manager

68 tests

For my coverage I have tested all the API endpoints, this covers most of the project because the entity manager's main job is handling persistent data together with its connected database.

With the relatively large number of tests written I have a code coverage % of 51,90%. This covers 97,44% of the database context, 100% of the controller endpoints accessed directly by users (this excludes endpoints connecting the entity manager and game manager services) and 98,50% of the services.

Test	Duration
Class: WeaponControllerTests Passed (11)	607 ms
Class: UserControllerTests Passed (13)	1,9 sec
Class: TestDbContextFactoryTests Passed (2)	540 ms
Class: RoomControllerTests Passed (15)	719 ms
Class: HubAdventurerControllerTests Passed (11)	609 ms
Class: EnemyServiceTests Passed (2)	5 ms
Class: DrawingControllerTests Passed (4)	550 ms
Class: AdventurerServiceTests Passed (10)	683 ms



```
Test                                          Duration   Traits
◢ ✓ textadventure_backend_entitymanag...       5,3 sec
  ◢ ✓ textadventure_backend_entitymana...      5,3 sec
    ▷ ✓ AdventurerServiceTests (10)            649 ms
    ▷ ✓ DrawingControllerTests (4)             527 ms
    ▷ ✓ EnemyServiceTests (2)                    3 ms
    ▷ ✓ HubAdventurerControllerTests (11)      577 ms
    ▷ ✓ RoomControllerTests (15)               681 ms
    ▷ ✓ TestDbContextFactoryTests (2)          517 ms
    ▷ ✓ UserControllerTests (13)               1,8 sec
    ▷ ✓ WeaponControllerTests (11)             579 ms
```

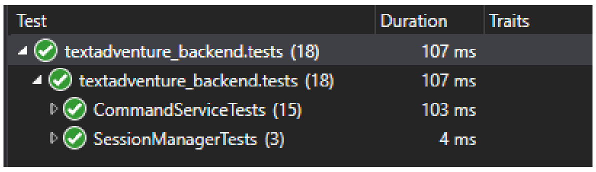
Game Manager

18 tests

For my coverage I have tested the main command flow for a user exploring the world (this means an adventurer in the exploring state), I have chosen to test this because I have spent a relatively large amount of time on previous tests and by testing these interactions, I will have covered the main flow and logic of my game, including handling a connected user's data and handling a command sent from a User.

To test this project, I have had to mock the SignalR connected created when a user resumes a game, this took me quite a bit of time and has taught me a lot about how Moq and mocking in general works. With the tests written I cover 37.72% of the project, with 67% of the session logic, 74,19% of the command service and 49% of the commands.

Test	Duration
Class: SessionManagerTests Passed (3)	4 ms
Class: CommandServiceTests Passed (15)	103 ms



```
Test                                       Duration   Traits
◢ ✓ textadventure_backend.tests (18)        107 ms
  ◢ ✓ textadventure_backend.tests (18)      107 ms
    ▷ ✓ CommandServiceTests (15)            103 ms
    ▷ ✓ SessionManagerTests (3)               4 ms
```