Testing Documentation

Document metadata: LO3 3.1 EVI 1

Contents:

Preface

Initially, we recall tasks directly associated with the implementation of tests:

- T1 (E2E test to prevent unauthorised access to database)
- T4 (DoS attack on API)
- T6 (manual human UI accessibility testing)
- T10 (compilation ability)
- T11 (continuous testing environment)
- T12 (authenticated access of API routes)
- T15 (logical unit testing of scheduling with relevant input data)
- T18 (performance test of scheduling unit)

Omitting of Tasks for this Portfolio

We postponed the performance test of database connection (verifying R6), so this will not be part of the portfolio. We already have another performance test (even though at the unit level of the system). R6 is of lowest priority.

We postponed T2 as T1 is identical in the testing approach and testing technique, and would not contribute any further to demonstrating my testing skills.

Testing Documentation

This section states specific planning for test execution and implementation, and how the process of following up on the planning worked. Results are documented in document `LO3 3.3 EVI 1.`

Task ID	Approach	History	Comments / Further Materials
T1	Go through all UI elements a user could interact with on review platform: - manual e2e testing using selenium to simulate user interactions. Too complicated.	First attempt did not work due to complicated structure of test. Second attempt with cypress.js: Solid E2E	

		testing 1. Set up the application state. 2. Take an action. 3. Make an assertion about the resulting application state. Checking that access can't be gained through incomplete login or review prompts.	
T4	Postman - We request and check all API routes specified in the LO2 section - Collection runner with 10, 100, 1000, 10000 request for each API route - Adequacy: - Not simulating DoS attacks which simulate concurrent users - So probably **not perfectly adequate** to for load testing - If the priority of the requirement we're testing was higher, would use Artillery to simulate a concurrent DoS-attack. The current approach is still adequate for now.	Note on adequacy after implementing: - Not simulating DoS attacks which simulate concurrent users - So probably **not perfectly adequate** to for load testing - If the priority of the requirement we're testing was higher, would use Artillery to simulate a concurrent DoS-attack. The current approach is still adequate for now.	
T6	Creation of test protocol for flashcard component UI accessibility testing `$button` = Show Answer 1. Go to https://interactive-education.vercel.app/flashcard/cGbkE2GhernWmifSF1x9. 2. Open the console of your web browser. 3. Try to select the `$button` button using your tab key. 3.1. Fail: **Fail with status 'sa-button no-select'** 3.2. Success: 3.2.1. Try to click the button by pressing enter. Successful if the console prints feedback for this button click (this is enabled via the scaffolding produced for [...]). 3.2.1.1. Fail: **Fail with status 'sa-button select no-submit'** 3.2.1.2. Success: 3.2.1.2.1. Go back to (1) and repeat the same steps for the 'Forgotten' and then for the 'Answer'	Successful on first iteration. Approach might need to be adjusted if system changes to respective UI component.	/

	`$button` values. Respective fail status identifiers: **fo-button** and **re-button.**		
T10	Use a child process to compile, check status code of compilation.	Status code checking not trivial, instead checking for success message in console log.	In repository testing folder, see `LO3/t10.test.js`.
T12	Use manually implemented API requesting and coherence checking via Jest.	Initial approach too time-consuming. Instead, we automate UI interactions using cypress.	
T15	We use the relevant input values produced by scaffolding work carried out for T14 and T15. **Functional testing:** The specification defines a correct Leitner schedule to double the previous interval between two due dates, so we test for this case. **Structural testing:** - Coverage - Observe that branches are not executed. - Can use scaffolding work - Aware that structural testing unusual for this small scope (single unit) but demonstrates my understanding of how to realise and implement it.		In repository testing folder, see `LO3/t15.test.js`.
T18	Approach: we use a measurable attribute (performance, time) to test R8.		In repository testing folder, see LO3/t18.test.js.