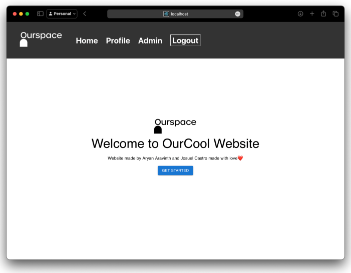
Ourspace

Inhaltsverzeichnis

Castro, Aravinth, 01.11.2023

Introduction

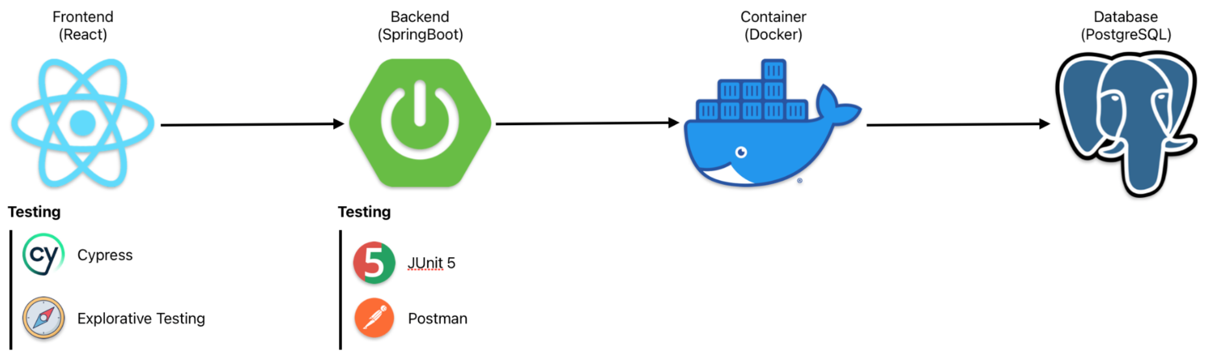
Welcome to the OurSpace Testing Documentation. This document outlines our systematic approach to testing a multi-user application. Here, we will detail our methodologies, protocols, and strategies ensuring robust application performance. Logo and Design handcrafted by Josuel Castro.



Welcome to OurCool Website

Website made by Aryan Aravinth and Josuel Castro made with love❤️

Project Structure



A cohesive technical framework is the backbone of any successful digital product. More often than not, it's the behind-the-scenes machinery that drives results. In this module, while the spotlight might be on the test scripts, let's not forget the bedrock on which our product stands.
Our blueprint laid out a user-centric frontend built on the React framework. React, with its component-based architecture, offered us the flexibility and efficiency we needed for dynamic user interactions. Transitioning to the backend, SpringBoot was our weapon of choice. Tailored for rapid application development, it played a crucial role in managing the complex operations and interfacing seamlessly with our PostgreSQL database. PostgreSQL, renowned for its reliability and scalability, was an obvious pick for our data storage requirements. The pivotal decision, however, was to encapsulate our application within Docker containers. This not only ensured uniformity across various deployment environments but also greatly simplified our scaling processes in cloud setups.
Though our initial ambition was grand, we swiftly recalibrated our approach, understanding the constraints posed by the testing requirements. Instead of a complex behemoth, we pivoted to a more manageable project that primarily revolved around CRUD operations. This was a strategic move, ensuring we could efficiently script tests that met the module's demands.
Despite our meticulous technical planning, the unforeseen time drain by the module tasks was a curveball. This led to slight adjustments in our timeline and a few task extensions. But, as always, challenges were met with resilience and a revised strategy.

Castro, Aravinth, 01.11.2023

Planning

Clear planning is always the key to a well led and managed project. Especially in massive ones! But in this module the product isn't the main focus, it’s the tests written for the project that will be graded and focused on.
We had a simple but clear plan setup, which was executed neatly. Our project Idea was quite bold at the beginning, but that changed after the realization that we can’t write tests for that project. So we decided on a simple project that has functionalities like CRUD, so that we can write tests that fit the obligations and requirements.

Our planning was well on point, the only flaw which we couldn’t foresee was the time invested into the module tasks. Which delayed a few tasks and had received a new deadline to be completed.

Date	Tasks
26.09.2023	Start Frontend development
26.09.2023	Start backend development
26.09.2023	Setup Project structure
03.10.2023	Setup Spring Boot Gradle Backend
03.10.2023	Navigation with Authorization
03.10.2023	Login with User
24.10.2023	Register/Create User
24.10.2023	Logout functionality
24.10.2023	Add User
24.10.2023	Edit User
24.10.2023	Complete remaining backend-Tasks
31.10.2023	Frontend UI
31.10.2023	Pages styling
31.10.2023	Complete Cypress Tests
31.10.2023	Complete Postman Tests
31.10.2023	Complete JUnit tests

Castro, Aravinth, 01.11.2023

Test Concept

Test Items

Our application comprises both frontend and backend components. A comprehensive architecture diagram will be attached to visually represent the components and their interconnections.
Features to be tested:

- User registration and authentication
- User data manipulation (edit profiles)
- Error handling mechanisms
- Frontend-backend API interactions
- Data validation processes
- Authorization and authentication mechanisms

Features not to be tested

Performance metrics and scalability aspects will be treated as non-functional requirements and won't undergo rigorous testing in this phase.

Approach

API Testing with Postman

- We'll use Postman to thoroughly test our Spring Boot backend's RESTful API endpoints. We'll create collections of API requests to cover a range of functionalities, including user registration, authentication, data manipulation (edit profiles), and error handling.
- Our API tests will focus on verifying the correctness of request and response payloads, status codes, headers, and security features.
- We'll also ensure that the API performs well under various conditions and remains secure against common vulnerabilities.
  Swagger Link(Project needs to be running to view Swagger)

End-to-End Testing with Cypress

- Cypress will be our tool of choice for end-to-end testing on the frontend. We'll write test scripts to simulate user interactions and validate user journeys.
- Our Cypress tests will cover essential user actions such as registration, login, and core application features (like editing profile or view users etc.). We'll verify that the frontend interacts seamlessly with our backend API.
- Additionally, we'll pay attention to data validation, authorization and authentication to ensure a robust user experience.

Unit Testing our Backend

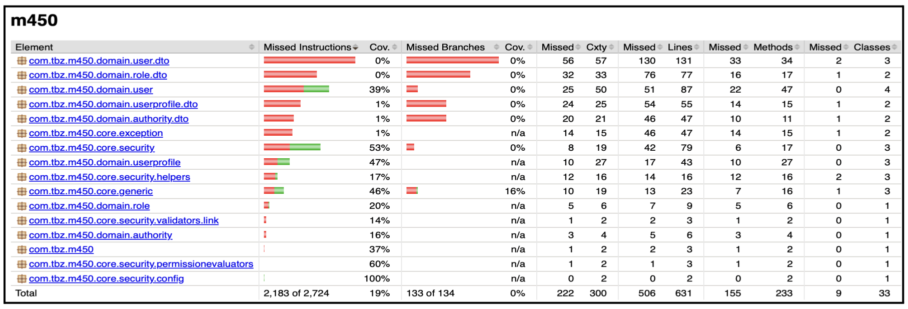
- For the backend, we'll leverage JUnit and Spring Boot Test to conduct comprehensive unit tests.
- Our focus will be on testing individual components in isolation, ensuring that each class, service, and controller behaves as expected. We'll write test cases to validate the business logic encapsulated in our services and utilities.
- By mocking dependencies using tools like Mockito, we ensure that our tests are not influenced by external factors and remain consistent.
- Our unit tests will cover scenarios such as data validation, exception handling, and edge cases. By integrating these tests into our CI/CD pipeline, we ensure that any changes or additions to the codebase are automatically validated for correctness, enhancing the reliability and maintainability of our application.

Item pass / fail criteria

- **Minor Errors:** The application runs but has specific flaws or cosmetic issues.
- **Moderate Errors:** The application presents clear mistakes but doesn't crash.
- **Severe Errors:** The application crashes or fails to perform a critical function.

Test Deliverables

- This Test Concept document.
- Test Scripts for Cypress.
- Collections of API requests in Postman.
- Unit test scripts.
- Report templates for test results. (with JaCoCo)



m450

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.tbz.m450.domain.user.dto		0%		0%	56	57	130	131	33	34	2	3
com.tbz.m450.domain.role.dto		0%		0%	32	33	76	77	16	17	1	2
com.tbz.m450.domain.user		39%		0%	25	50	51	87	22	47	0	4
com.tbz.m450.domain.userprofile.dto		1%		0%	24	25	54	55	14	15	1	2
com.tbz.m450.domain.authority.dto		1%		0%	20	21	46	47	10	11	1	2
com.tbz.m450.core.exception		1%		n/a	14	15	46	47	14	15	1	2
com.tbz.m450.core.security		53%		0%	8	19	42	79	6	17	0	3
com.tbz.m450.domain.userprofile		47%		n/a	10	27	17	43	10	27	0	3
com.tbz.m450.core.security.helpers		17%		n/a	12	16	14	16	12	16	2	3
com.tbz.m450.core.generic		46%		16%	10	19	13	23	7	16	1	3
com.tbz.m450.domain.role		20%		n/a	5	6	7	9	5	6	0	1
com.tbz.m450.core.security.validators.link		14%		n/a	1	2	2	3	1	2	0	1
com.tbz.m450.domain.authority		16%		n/a	3	4	5	6	3	4	0	1
com.tbz.m450		37%		n/a	1	2	2	3	1	2	0	1
com.tbz.m450.core.security.permissionevaluators		60%		n/a	1	2	1	3	1	2	0	1
com.tbz.m450.core.security.config		100%		n/a	0	2	0	2	0	2	0	1
Total	2,183 of 2,724	19%	133 of 134	0%	222	300	506	631	155	233	9	33

Environmental Needs

The testing environment will need both hardware and software configurations:

- **Hardware**: Standard development machines with sufficient memory and processing capabilities.
- **Software**: Spring Boot environment, Node.js for Cypress, Postman, and the respective databases and services the application relies upon.

Schedule

The detailed timeline will be attached, indicating phases like:

- Initial unit tests as per TDD.
- Integration tests once major components are developed.
- E2E tests nearing the completion of the project.

Reflexion

Josuel

I liked the subject of Testing. It was very interesting and intriguing. I really came into the lesson with practically no knowledge about testing, but now I can say that testing is much more complex and deeper than it might seem.

I was loving the individual tasks, even though they were quite time consuming, but in the end we completed all the tasks we needed to. The project was the tricky part. Without seeing my partner very often I always needed to have a meeting with him to see where we both stood with our progression. I didn't really like the project part, as it was just too much and when things don’t seem to be working, you start to worry.

During this course, I learned a lot about Test-Driven Development (TDD). It made me see coding differently, especially the idea of writing tests before starting the actual code. This method showed me the importance of planning ahead in software work.

Code Reviews were also an essential part. Having someone else look at the code helped catch mistakes and areas to improve. Working with my partner, we regularly checked each other's code to make sure it was good and didn't have any issues.

As for the topic of Testing, it was all new to me. I started the course knowing very little about it. But now, I realize there's a lot to it. The tasks were tough and took time, but finishing them felt great. The project part was a bit hard, especially because I didn't meet my partner often. So, we had to plan our meetings to check our progress. There were moments when the project felt too much, especially when things didn't go as planned. But by the end of the M450 course, I learned so much and started to value the importance of testing in coding.

Castro, Aravinth, 01.11.2023

Aryan

Reflecting upon this module, I would say it has had its ups and downs. The module structure was inconsistent and we never had a clear view of when we could work on our project and when we needed to complete the module task. But nonetheless we were able to manage and complete our tasks and projects.

The project itself went well, we didn't really feel congested in comparison to other modules. We worked well as a team and had a good flow and our chemistry meshed really well, since it wasn’t our first time working together. We didn’t have any merge conflicts and our communication went well, there wasn’t a basic error that stopped us.

In my opinion the only flaw we had was, estimating how long we would take for the module tasks. Which clearly took some time and decreased our time for the project.

I definitely learned a lot of new things from testing. Especially automated testing like cypress and so on. My knowledge on packages has increased by a massive margin. It is impressive that even if this module is new and it is a bit inconsistent, it delivers a lot of knowledge that is easy to digest and comprehensible