**Sprint 4 Testing Plan**

**PewPew Squad**

**Condo Management System Project:**

Condofy

Gevorg Alaverdyan #40202177

Ramy Attalla #40205642

Khashayar Azad #40211574

Sophie Hajjar #40210405

Yeprem Antranik #40204291

Mario El Shaer #40210124

Omar Shehata #40164193

Tarek Elalfi #40197527

Jay Patel #40203705

Khaled Saleh #40210125

24-02-2024

### 

### **1. Introduction**

This testing plan delineates the comprehensive approach, tools, metrics, and coverage strategies tailored for Sprint #4. The primary emphasis will be on meticulous manual testing, rigorous backend assessment utilizing Postman, and the establishment of foundational tests for system functionality and integration.

### 

### **2. Testing Tools and Approach**

The test-driven approach is beneficial for several reasons. Firstly, it encourages developers to clearly define the expected behavior of their code upfront through well-defined test cases. This helps in understanding the problem domain and desired outcomes more clearly. Secondly, it promotes an iterative and incremental development process, allowing developers to build and improve the codebase incrementally while ensuring compatibility with existing functionality. Additionally, by constantly running tests and ensuring that all code changes pass the defined test cases, developers can catch bugs and errors early in the development cycle, leading to more robust and maintainable code. Overall, adopting the test-driven approach helps enhance the quality, reliability, and maintainability of the software product while accelerating the development process.

The project makes use of important tools to for manual testing and to ensure high code coverage:

#### **Code Coverage:**

* **Utilization of Jest**: Leveraging Jest for code coverage metrics facilitates comprehensive evaluation of line, function, statement, and branch coverage.
* **Aim for High Coverage Percentage**: Striving for a high code coverage percentage ensures thorough testing and validation of the codebase, leading to enhanced robustness and reliability.

**Tools for Manual Testing:**

* **Postman**: Utilizing Postman facilitates the testing of endpoints and verifies seamless communication between different components of the backend system.
* **MongoDB**: MongoDB offers a user-friendly GUI that allows for copying important information for testing requests, as well as checking that operations were successful.

Now for the types of testing:

#### **Manual Testing:**

* **Depth Testing**: Each developer assumes responsibility for meticulously testing their respective codebase to ensure comprehensive coverage. Tools like Postman and MongoDB’s GUI are used extensively.
* **Test-driven Approach**: Adhering to the test-driven development (TDD) methodology involves crafting tests before writing actual code. This ensures alignment with specified requirements and facilitates long-term maintainability.
* **Periodic Code Reviews**: Conducting regular code reviews provides opportunities to identify and address any emerging issues, fostering a culture of continuous improvement.

#### **Developer-level Testing:**

* **Endpoint Testing**: Developers conduct rigorous testing of endpoints to validate their functionality.
* **MongoDB GUI**: Utilizing MongoDB GUI allows for convenient verification of object creation or modification, depending on the operation.
* **CRUD Operation Testing**: Comprehensive testing of CRUD (Create, Read, Update, Delete) operations ensures the robustness and reliability of the system under various scenarios.

#### **Unit Testing:**

* Unit testing has been in progress from the first sprint. During the second sprint we investigated easier ways to create unit tests and decided to use NestJs’ built in testing, along with mockingoose. Mockingoose is a library that allows developers to mock database actions, to avoid real read and write operations to the database. The new approach has been employed and will be used for the remainder of this project.
* To ensure that high statement coverage is met, GitHub actions has been configured to give a report of test coverage in every PR. This will ensure that for all features there is sufficient testing.

#### **Integration Testing:**

* Integration testing is an important part of testing, as it ensures that different components work together. For the purposes of the backend testing that we are focusing on, this means testing an entire module from end to end.
* Integration testing will be implemented to simulate the expected interaction with the production system. Automated integration testing will ensure no unintended breaking changes are introduced, because it will be harder to manually test the end-to-end behavior of each API endpoint.
* Developers will use NestJs’ end-end testing to write end-to-end tests that testing the full behavior of API endpoints. The tests will mock a REST API request coming from the front end, to ensure that no breaking changes are introduced.
* Integration testing is of the upmost important as we approach release and aim to stabilize production.

### 

### **3. Metrics and Coverage**

#### **Code Metrics:**

In the previous sprint we discussed the possibility of employing SonarQube for measuring important metrics. However, SonarQube and many other similar tools only offer a free trial, full access to the product is expensive.

Moreover, after more through investigations we have decided to use Codacy. Codacy is a quality and security analysis tool, that has been configured to give important measures for both the web application and mobile application. The tool runs automatically on every PR, as well as the main branches.

A screenshot of a phone

Description automatically generated

Taking a closer look at the CondoManagement repo, which contains the frontend for the web application, as well as the main backend, we can see that the tool gives a quality evolution chart, and a breakdown of issues.

The code evolution chart displays the percentage of issues (code style, error prone, best practice, security related issues), complexity (percentage of complex files), and code duplication.

A graph with a line

Description automatically generated

Moreover, Codacy gives a breakdown of issues and categorizes them so that they can handled according to their priority.

A screenshot of a computer

Description automatically generated

Moreover, upon clicking on one of the issues’ categories it gives a list of issues with their urgency.

A screenshot of a computer

Description automatically generated

The tool also gives an overall grade for each file and breakdown of each file and commit.

A screenshot of a computer

Description automatically generated

A white paper with blue lines

Description automatically generated

Moreover, the tool also analysis code patterns.

A screenshot of a computer

Description automatically generated

In future sprints, Codacy will be used more closely to ensure that high priority issues are fixed, and code is refactored to meet high quality exceptions and follow patterns. This will allow for less fixes to be required as the project grows, since the height quality code will allow for easier maintainability. To summarize sprint 4 will make use of the following metrics:

* Overall code duplication.
* Code style issues.
* Error prone issues.
* Best practice-related issues.
* Security issues.
* File grades, complexity, and duplication (which effects the overall percentages.)

### 

### **4. Acceptance Test Scenario**

Scenario: Manager View Employees on Mobile

1. Manager visits Condofy Mobile.
2. Manager navigates to “Manage Employees” page.
3. Expected: A table with all the employees of the company with ability to delete any of them.

Scenario: Manager Deletes Employee on Mobile

1. Manager visits Condofy.
2. Manager navigates to “Manage Employees” page.
3. Manager chooses an employee and presses on the right-aligned dropdown in their row and presses delete.
4. Expected: A pop up asks them to confirm their actions.
5. Manger
   1. Confirms:
      1. Expected: The user is deleted.
   2. Cancels
      1. Expected: Returned to “Mange Employees” page.

Scenario: Manager Creates Employee

1. Manager visits Condofy.
2. Manager navigates to “Manage Employees” page.
3. Manager clicks “Add Employee.”
4. Manager enters the required information.
5. Expected: Employee is created, and a success toast is dispatched.

Scenario: Owner Views Units on Mobile

1. Owner visits Condofy.
2. Owner navigates to “Properties”
3. Owner can view all units and their parkings, and storages

Scenario: Owner Claims Units on Mobile

1. Owner visits Condofy.
2. Owner navigates to “Properties”
3. Owner can click “Claim New Unit” and enter a registration key

Scenario: Owner Submits Requests

1. Owner visits Condofy.
2. Owner navigates to “Create Request”
3. Owner chooses category
4. Owner inputs request info
5. Owner can submit successfully and receives feedback

Scenario: Employee Handles Requests

1. Employee visits condofy
2. Employee navigates to requests
3. Employee can view all requests
4. Employee can resolve a request
5. Employee can respond to a request

Scenario: Manager Creates Facility

1. Manager visits conofy
2. Manager navigates to properties
3. Manager toggles to view properties facilities page
4. Manager clicks “Add New”
5. Manager enters facility info.
6. Manager submits successfully and receives feedback.

Scenario: Manager Views Facilities

1. Manager visits conofy
2. Manager navigates to properties
3. Manager toggles to view properties facilities page
4. Manager can view all facilities

Scenario: Manager Deletes Facility

1. Manager visits conofy
2. Manager navigates to properties
3. Manager toggles to view properties facilities page
4. Manager can view all facilities
5. Manager presses delete icon
6. Manager confirms decision
7. Manager submits successfully and receives feedback.

Scenario: Manager Modifies Facility

1. Manager visits conofy
2. Manager navigates to properties
3. Manager toggles to view properties facilities page
4. Manager can view all facilities
5. Manager presses modify icon
6. Manager enters info information
7. Manager confirms decision
8. Manager submits successfully and receives feedback.

Scenerio: Owner Makes Reservation

1. Owner visits condofy
2. Owner navigates to reserve
3. Owner sees all facilities in a selected building
4. Owner sees all availabilities
5. Owner selects an availability
6. Owner submits successfully and receives feedback.

Scenerio: User views Notifications

1. User visits condofy
2. User navigates to “Notifications” page
3. User views all notifications on the account

### **5. Test Execution**

During the development phase, developers will execute manual tests as an integral part of their workflow. Additionally, backend testing utilizing Postman will focus on thoroughly assessing endpoints and ensuring seamless communication between components. Furthermore, Jest will be leveraged for implementing a test-driven development approach and monitoring code coverage.

### **6. Test Environment**

To ensure consistency and isolation during testing, Docker will be utilized to set up a standardized test environment. This approach enables the creation of reproducible testing environments across different development setups, minimizing potential discrepancies and ensuring uniformity in test results. By employing Docker, we can streamline the process of environment setup, deployment, and configuration, thereby enhancing the efficiency and reliability of our testing procedures.

### **7. Conclusion**

In conclusion, this testing plan serves as a roadmap to guarantee the quality and dependability of the code produced during Sprint #4. Through the implementation of a comprehensive testing strategy, encompassing test-driven development, Postman, and Jest, along with diligent monitoring of code metrics and coverage, our objective is to deliver a product characterized by resilience and excellence. Moreover, we will emphasize working on integration testing which we have already set up and started in sprint #3. And by implementing integration tests we decrease the time required to fix breaking changes and decrease the time required to test API endpoints. By prioritizing acceptance test scenarios, which emphasize crucial user interactions, we remain committed to ensuring a seamless and satisfactory user experience.