ROOMIT

Testing Document

Eylon Sade
Ofir Povimonsky
Noa Magrisso

ROOMIT
Your ideal roommate awaits

1. Testing functional requirements

index	Description	Input	Expected Result	Actual Result	Pass/Fail
1	Registration				
1.1	Register with valid profile information	username, email, password, personal information	Successful registration	Successful registration	Pass
1.2	Register with existing username	username, email, password	Error: Username already exists	Error: Username already exists	Pass
1.3	Register with invalid username	invalid username, email, password	Error: Invalid username	Error: Invalid username	Pass
1.4	Register with same email as existing user	username, used email, password	Error: Registration failed – email in use	Error: Registration failed - email in use	Pass

1.5	Register with common password	username, email, common password	Error: Common password used	Error: Common password used	Pass
1.6	Register with short password	username, email, short password	Error: Password too short	Error: Password too short	Pass
1.7	Register with entirely numeric password	username, email, numeric password	Error: Numeric password used	Error: Numeric password used	Pass
1.8	Register with password similar to username	username, email, password	Error: Password similar to username	Error: Password similar to username	Pass
1.9	Register with non-matching passwords	username, email, password	Error: Passwords don't match	Error: Passwords don't match	Pass
2	Login				

2.1	Login with correct credentials	username, password	Successful login	Successful login	Pass
2.2	Login with wrong password	username, wrong password	Error: Login failed	Error: Login failed	Pass
2.3	Login with wrong username	wrong username, password	Error: Login failed	Error: Login failed	Pass
2.4	Login with non-existent username	non-existent username, password	Error: Login failed	Error: Login failed	Pass
2.5	Login when already logged in	username, password	Error: Already logged in	Error: Already logged in	Pass
3	Logout				
3.1	Logout when logged in	-	Successful logout	Successful logout	Pass

3.2	Logout when not logged in	-	User is already logged out	User is already logged out	Pass
4	Like/Unlike user				
4.1	Like + unlike user from profile page	username to like	Successful like/unlike	Successful like/unlike	Pass
4.2	Like + unlike user from profile page	Username to unlike	Successful unlike	Successful unlike	Pass
4.3	Check who liked me (not logged in)	-	Error: Not logged in	Error: Not logged in	Pass
4.4	Check who I liked (not logged in)	-	Error: Not logged in	Error: Not logged in	Pass
5	Requirements, Profile information & Property offers				
5.1	See property requirements display – no user connected	username	Redirected to login page	Redirected to login page	Pass

5.2	Check roommate requirements display – no user connected	username	Redirected to login page	Redirected to login page	Pass
5.3	Check property offer display	username	Redirected to login page	Redirected to login page	Pass
5.4	Check profile information display (offline)	username	Redirected to login page	Redirected to login page	Pass
5.5	Change requirements success	username, password	Successful login, update property and roommate requirements	Successful login, update property and roommate requirements	Pass
6	Recommendation System				
6.1	Recommendation system	usernames, password (2 similar users)	Successful interaction between users for recommendation – one sees the other's liked users	Successful interaction between users for recommendation – one sees the other's liked users	Pass

6.2	Recommendation system	usernames, password (2 similar users)	Successful interaction between users for recommendation – one doesn't sees the other's unliked users	Successful interaction between users for recommendation – one doesn't sees the other's unliked users	Pass
6.3	Recommendation system	usernames, password (2 unsimilar users)	One doesn't sees the other's liked users	One doesn't sees the other's liked users	Pass

2. Testing non-functional requirements –

   Security tests –

Test Method Name	Title	Description
test_unauthorized_access	Unauthorized Access	Verifies that when no user is logged in and attempts to access the admin page, they are redirected to the login page.

test_invalid_credentials	Invalid Credentials	Tests the scenario where the admin tries to log in with the wrong password and checks if the correct error message is displayed.
test_password_reset	Password Reset	Tests the functionality of password reset by sending a request and verifying the appropriate redirection.
test_role_permissions	Role Permissions	Tests the role-based permissions. Checks if a regular user is redirected to the login page when accessing the admin page and if the admin has access.
test_input_sanitization	Input Sanitization	Verifies that user input is sanitized to prevent cross-site scripting (XSS) attacks.

Performance tests –

The **test_performance** module is designed for performance testing using Locust, where multiple users simulate various tasks on a web application. This module focuses on simulating user registration, filling out user information, and performing tasks related to viewing the homepage and user profiles.

The performance test consists of three tasks:

- registration: Simulates user registration to the system and filling out their personal information and requirements.
- view_homepage: Simulates users accessing the homepage of the web application by sending a GET request to the root URL.
- view_profile: Simulates users viewing user profiles by sending a GET request to a randomly selected user's profile page. The username for the profile is chosen from the usernames list, which contains the usernames of all registered users.

The selected tasks for performance testing were intentionally chosen as they represent the heaviest operations in terms of database interactions and computational complexity. By focusing on these

tasks, we aim to evaluate the system's performance under demanding scenarios that involve substantial database operations and computational calculations.

Request Count	Failure Count	Median Response Time	Average Response Time	Min Response Time	Max Response Time	Average Content Size	Requests/s	Failures/s
264	0	440	1515.028477	20.5008	7884.7544	7696.128788	8.30455396	0

By running this performance test with Locust and scaling up the number of users, it becomes possible to measure the web application's performance under load. This module can help identify potential performance bottlenecks, such as slow response times or resource limitations, allowing developers to optimize and improve the application's scalability.

Test result (aggregated) –

3. Test-Driven Development (TDD) –

   In our group project, we did not strictly follow the Test-Driven Development (TDD) approach. However, we collectively emphasized writing tests for the core functionalities and continuously added and updated tests throughout the development process. While TDD can provide several benefits, we opted for a more iterative and exploratory approach to accommodate the evolving requirements and frequent changes.

4. Random & automatically-generated tests –

   We focused on testing the boundaries of inputs and explored various combinations of values. By using Selenium, we automated the testing process and generated a significant number of test cases with different input scenarios. This allowed us to uncover potential issues related to boundary conditions and handle them effectively.

5. Testing the User Interface –

   To test the user interface (UI) of our Django project, we utilized Selenium, a widely-used tool for UI automation. By simulating user interactions and automating the input of various values, we covered a broad range of scenarios to ensure the UI behaved correctly. This approach enabled us to verify the functionality of forms, navigation flow, responsiveness, and error handling. While we did not provide on-line help specifically, we strived to create an intuitive and user-friendly interface that minimized the need for extensive help documentation.

6. Testing Build, Integration & Deployment –

For the build and deployment processes, we used the command "**python manage.py runserver**" to launch the development server and checked for any build-time issues, such as syntax errors or missing dependencies.
Regarding installation, as our project is a website deployed through PythonAnywhere (accessible via the URL https://roomit.pythonanywhere.com/), there is no specific installation or uninstallation process required. However, we manually verified that the necessary resources, such as database connections and file system access, were in place for the proper functioning of the deployed website. Additionally, we ensured that the project's prerequisites, such as specific Python, Django version and other used libraries, were documented in the **requirements.txt** file which is used to install them thus verifying that these were properly installed.