|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Matrix** | | | | | |
| **Test Case No** | **Test Case Name** | **Input** | **Expected Outcome** | **Actual Outcome** | **Result (Pass/Fail)** |
| 1 | ChatModelTest.test\_chat\_creation | 1)Createuser:username='testuser', email='test@test.test'  2) Create Chat(user=user) | Chat successfully created and linked to the user,  is\_active = True | Chat created with user=testuser,is\_active=True, test passes | Pass |
| 2 | MessageModelTest.test\_message\_creation | 1. Create user 2. Create chat with this user 3. Create Message(chat=..., sender=..., content=...) | Message is stored with correct values: chat, sender, content | Message created with correct chat, sender, and content. Test passed | Pass |
| 3 | UserChatViewTest.test\_user\_chat\_view\_get | 1. Login user 2. GET request to 'support:user\_chat' | GET request to 'support:user\_chat' returns page with status 200, uses template 'support/user\_chat.html', contains “Chat with Support” | Status 200 received, rendered template 'support/user\_chat.html',contains text 'Chat with Support' | Pass |
| 4 | UserChatViewTest.test\_user\_chat\_view\_post | 1. Login user 2. POST request with {'content': 'Test message'} to 'support:user\_chat' | POST request creates a new message with content='Test message' and saves it in the database, then redirects | Status 302 (redirect) received,Message with content='Test message' exists in DB | Pass |
| 5 | FindTransportViewTest.test\_find\_transport | GET request without parameters to 'find\_transport:find\_transport' | Status 200, displays 'E-Bike', 'E-Scooter', 'Bike'; does not contain 'Not Available' | Test passes, verification successful,as filtering is absent, all available vehicles (except unavailable ones) are shown | Pass |
| 6 | FindTransportViewTest.test\_find\_transport | GET request with {"type": "E-Bike"} | Displays only "E-Bike", does not contain "Bike", "E-Scooter" | Test fails because the HTML form contains <option> elements for "Bike" and "E-Scooter" | Fail |
| 7 | FindTransportViewTest.test\_find\_transport | GET request with {"min\_battery": "70"} | Displays "E-Bike" (80%), does not contain "E-Scooter" (50%) | Test fails because "E-Scooter" is still present in the HTML (select) | Fail |
| 8 | FindTransportViewTest.test\_find\_transport | GET request with {"type": "E-Scooter", "min\_battery": "70"} | Since "E-Scooter" has 50%, it should not be displayed | Test fails because "E-Scooter" is still present in the HTML (select) | Fail |
| 9 | FindTransportViewTest.test\_find\_transport | GET request with {"min\_battery": "abc"} | Status 400, JSON: {"error": "Invalid battery percentage"} | Test passes successfully (filter throws an error and returns 400) | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Report  
Oleksandr Podynihalzov

## Developing

In the development of this project, we used the Kanban methodology to distribute the workload effectively. This allowed us to fairly allocate tasks among team members and clearly visualize who was working on what at any given moment. Our team decided to divide responsibilities based on applications rather than project sections. As a result, each team member implemented multiple system functions, which helped clearly define individual contributions to the project.We chose parts of the project where we felt most confident in development. For example, I took on the development of the support function because I have experience implementing such features in various ways like through API and on different platforms.

Challenges with Filtering in Tests   
The filtering tests are failed because of presence of terms such as "Bike" and "E-Scooter" within the HTML, even when the filtering logic results in an empty vehicle list. These terms are consistently included in the <select> filter options, causing the assertNotContains function to detect them in the source code, which ultimately leads to test failures. We can fix it in two ways: 1. Modify the tests to concentrate on a specific subset of transport listings (for example, employing BeautifulSoup to parse the <div class="vehicle-list"> and identify pertinent entries). 2. Examine more specific elements (such as verifying <h3>Bike</h3> instead of simply searching for the term "Bike").

## Migration Challenges

We faced migrations conflicts such as migrations collisions in case of pulling changes with migrations.To mitigate this issue, we have included migration files in .gitignore to prevent them from being tracked in the repository. Consequently, each developer will run migrations immediately after pulling the latest code from the branch, thereby ensuring that conflicts are avoided prior to making additional changes.