|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Matrix** | | | | | |
| **Test Case No** | **Test Case Name** | **Input** | **Expected Outcome** | **Actual Outcome** | **Result (Pass/Fail)** |
| 1 | test\_voucher\_creation | Voucher with code TEST100, discount 20, active status True, user assigned | The voucher should be created successfully with the given attributes. | The voucher was created, assigned an ID, and all attributes matched expectations. | Pass |
| 2 | test\_voucher\_string\_representation | Voucher with code TEST100 | str(voucher) should return 'TEST100' | str(voucher) correctly returned 'TEST100' | Pass |
| 3 | test\_vehicle\_voucher\_application | Vehicle rental price: 10.0  Voucher VEHICLE20 (20% discount) | The discounted price should be 8.0, and discount value should be 20 | Response returned price 8.0 and discount 20 | Pass |
| 4 | test\_subscription\_voucher\_application | Subscription price: 50.0  Voucher SUB10 (10% discount) | The discounted price should be 45.0, and discount value should be 10 | Response returned price 45.0 and discount 10 | Pass |
| 5 | test\_invalid\_voucher\_code | Voucher code: INVALID\_CODE | Response should return a 400 error with the message "Invalid voucher code" | Response returned status 400 and the expected error message | Pass |
| 6 | test\_expired\_voucher | Expired voucher (valid\_to in the past) | Response should return 404 (voucher not found) | Response returned 404 | Pass |
| 7 | test\_inactive\_voucher | Inactive voucher (active=False) | Response should return 400 with "Invalid voucher code" | Response returned 400 and the expected error message | Pass |
| 8 | test\_voucher\_usage\_limit | Voucher with max\_use=1, used=1 | Response should return 400 with usage limit reached" | Response returned 400 and the expected error message | Pass |
| 9 | test\_voucher\_apply\_form\_valid | Form submitted with code 'TESTCODE' | Form should be valid | Form was valid | Pass |
| 10 | test\_voucher\_apply\_form\_invalid | Form submitted with an empty code | Form should be invalid | Form was invalid | Pass |
| 11 | test\_route\_creation | **Input Data:** Create a Route object with user testuser, name Test Route, and points [{lat: 53.349805, lon: -6.26031}, {lat: 53.350805, lon: -6.27031}] | **Expected Result:** The route should be successfully created, have a valid ID, retain the correct name, be linked to the user, and have a creation timestamp | Route was created successfully with all expected attributes | Pass |
| 12 | test\_route\_string\_representation | Create a Route object with name Test Route | Calling str(route) should return Test Route | The string representation matched Test Route | Pass |
| 13 | test\_get\_places\_success | Mock API request for places with category restaurant, latitude 53.349805, longitude -6.26031 | API should return a 200 response with JSON containing a features key | Response returned 200 with features key in JSON | Pass |
| 14 | test\_get\_places\_api\_failure | Simulate an API failure returning a 404 Not Found | The view should return a 404 response | Response returned 404 | Pass |
| 15 | test\_get\_route\_success | Mock API request with waypoints [[53.349805, -6.26031], [53.350805, -6.27031]] | API should return a 200 response with a features key containing route data | Response returned 200 with features key in JSON | Pass |
| 16 | test\_get\_route\_invalid\_waypoints | Test route retrieval with empty waypoints and single waypoint | API should return a 400 Bad Request | Response returned 400 for both cases | Pass |
| 17 | test\_get\_route\_method\_not\_allowed | Send a GET request to an endpoint that only supports POST | API should return 405 Method Not Allowed | Response returned 405 | Pass |
| 18 | test\_map\_view\_renders\_correct\_template | Request the map view page | The response should be 200 OK and render the get\_direction.html template | The expected template was rendered with status 200 | Pass |
| 19 | test\_bike\_creation | Create a Vehicle object with type "Bike", latitude 53.349805, longitude -6.26031, status True, and price\_per\_hour 2.5 | The bike should be created successfully with a null battery percentage and the correct type "Bike" | The bike was created successfully with the expected attributes | Pass |
| 20 | test\_e\_bike\_creation | Create a Vehicle object with type "E-Bike", latitude 53.349805, longitude -6.26031, status True, battery\_percentage 80, price\_per\_hour 5.0, and associated with an EVStation | The e-bike should be created successfully with a battery percentage of 80 and linked to the EVStation | The e-bike was created successfully with the expected attributes | Pass |
| 21 | test\_e\_bike\_invalid\_battery\_percentage | Attempt to create a Vehicle object with type "E-Bike", latitude 53.349805, longitude -6.26031, status True, battery\_percentage 101 (invalid), and price\_per\_hour 5.0 | Attempt to create a Vehicle object with type "E-Bike", latitude 53.349805, longitude -6.26031, status True, battery\_percentage 101 (invalid), and price\_per\_hour 5.0 | A ValidationError was raised as expected | Pass |
| 22 | test\_station\_capacity | Create five E-Bike objects associated with an EVStation with a max capacity of 5, then attempt to add one more | The first five e-bikes should be created successfully, but adding the sixth should raise a ValidationError | The first five e-bikes should be created successfully, but adding the sixth should raise a ValidationError | Pass |
| 23 | test\_ev\_station\_creation | Create an EVStation object with latitude 53.349805, longitude -6.26031, and max\_spaces 10. | The EV station should be created successfully with the correct attributes. | The EV station was created successfully with the expected attributes. | Pass |
| 24 | test\_get\_vehicles\_all | Retrieve all available vehicles via the API endpoint | Retrieve all available vehicles via the API endpoint | The API should return a 200 response with a list of available vehicles (excluding unavailable ones) | Pass |
| 25 | test\_get\_vehicles\_by\_type | Retrieve vehicles of type "E-Bike" via the API | The API should return a 200 response with only E-Bikes in the results. | The response returned 200 with one E-Bike in the results | Pass |
| 26 | test\_get\_vehicles\_invalid\_battery | Retrieve vehicles with an invalid battery percentage value (e.g., "abc"). | The API should return a 400 Bad Request with an appropriate error message. | The response returned 400 with the expected error message. | Pass |
| 27 | test\_get\_stations | Retrieve all EV stations via the API endpoint. | The API should return a 200 response with a list of stations. | The response returned 200 with at least one station in the results. | Pass |
| 28 | test\_get\_station\_invalid\_id | Attempt to retrieve a station with an invalid ID parameter | The API should return a 404 response with an appropriate error message. | ValueError | Fail |

Report  
Oleksandr Podynihalzov

In this iteration, I prioritized improving security, performance, and automation.  
Find\_transport App:I incorporated an interactive map where the user location is the center. Stations are loaded dynamically using an API call to the server, which obtains the addresses from a third-party API. A path can be created from the user's location by clicking on a station. Routing is performed server-side with an external API utilized to calculate routes, hence reducing the burden on the client.  
To improve performance, I added caching for station information. I also created data generators for automatic JSON generation, supporting dynamic display of transport units. Filtering by type or battery percentage is now instant without page reloading, and transport is sorted based on its allocated station. More than one filter can be applied at a time, and pagination dynamically adjusts depending on the vehicles.  
Support App: The support chat now uses WebSockets for real-time messaging, e.g., audio alert on new messages. Admins can view all ongoing user chats, enter any conversation, and send messages. Messages sent via WebSocket are stored in the database, and I created an asynchronous Daphne server to provide this feature.  
Voucher App:I implemented vouchers that are redeemable for different payment types (e.g., subscriptions, rentals) with validation to correctly use them.  
  
Get\_direction App: Users can enter detailed routes with turn-by-turn directions. Users can place pins on the map and create multi-stop routes automatically. Surrounding points of interest (e.g., museums, cafes) can even be found and routed to.  
Data is passed in API endpoints, and only processed results after they have been processed are returned to the client. All page data is dynamically generated through scripts for automatic updates to categories, APIs, and maps.  
I installed an SMTP server for notification emails, i.e., confirmation of rentals and subscriptions. I also set up Django's own logging for the monitoring of warnings and errors. Throughout the project, I used reusability principles so that few code adjustments were done in adding new functionalities.

Oleksandr Kurilets

In this iteration, i had focus to work on subscriptions, statistics, booking, and some design parts. Here’s what I did.

For subscriptions, I built a system where users can buy plans like monthly or weekly. It checks if a user has an active plan and activates a new one if not. I handled dates and conditions to make sure it runs smooth.

Next, statistics. I created a model to track user activity – hours they ride and money spent. It updates after every booking. This helps see how users use the app and can be good for future improvements.

Booking was the biggest part. I made it so users can rent vehicles, pick hours, and pay with Stripe or a subscription. I added vouchers for discounts too. If a vehicle isn’t available, it redirects back. Took time to get all the logic right.

For design, I used Bootstrap and some CSS. It’s simple – clean forms, working buttons, easy to use. Not super fancy, but it does the job.

Overall, the features work well. Subscriptions activate, stats update, booking flows nice, and design looks okay. Could add more tests or style later, but it’s solid now.