# Structured Test Cases

## Test Case ID: TC-CustomerLogin-01

**Test Case Title:** Verify user login with valid credentials

**Test Objective**: Ensure that the user can log in successfully with valid credentials.

**Preconditions:** User must be registered in the system.

**Test Data:** Username: testuser, Password: correctpassword

**Test Steps:**

* - Send a POST request to /api/auth/custumerlogin with valid username and password.
* - Observe the response.

**Expected Result:** User should receive a success response and a valid JWT token.

**Actual Result:** User logged in successfully, JWT token received.

**Postconditions:** User is authenticated and can access protected routes.

**Status:** Pass

Remarks:

## Test Case ID: TC-RenterLogin-01

**Test Case Title:** Verify user login with valid credentials

**Test Objective**: Ensure that the user can log in successfully with valid credentials.

**Preconditions:** User must be registered in the system.

**Test Data:** Username: testuser, Password: correctpassword

**Test Steps:**

* - Send a POST request to /api/auth/renterlogin with valid username and password.
* - Observe the response.

**Expected Result:** User should receive a success response and a valid JWT token.

**Actual Result:** User logged in successfully, JWT token received.

**Postconditions:** User is authenticated and can access protected routes.

**Status:** Pass

Remarks:

## Test Case ID: TC-AdminLogin-01

**Test Case Title:** Verify admin login with valid credentials

**Test Objective**: Ensure that the admin can log in successfully with valid credentials.

**Preconditions:** admin must be registered in the system.

**Test Data:** Username: admintestuser, Password: correctpassword

**Test Steps:**

* - Send a POST request to /api/auth/adminlogin with valid username and password.
* - Observe the response.

**Expected Result:** admin should receive a success response and a valid JWT token.

**Actual Result:** admin logged in successfully, JWT token received.

**Postconditions:** admin is authenticated and can access protected routes.

**Status:** Pass

Remarks:

**Test Case ID: TC-Login-02**

**Test Case Title:** Verify user login with invalid credentials

**Test Objective**: Ensure that login fails with incorrect credentials.

**Preconditions:** User must be registered in the system.

**Test Data:** Username: testuser, Password: wrongpassword

**Test Steps:**

* - Send a POST request to /api/auth/login with invalid password.
* - Observe the response.

**Expected Result:** User should receive an error response indicating invalid credentials.

**Actual Result:** Error message received: Invalid credentials.

**Postconditions:** User is not authenticated.

**Status:** Pass

Remarks:

**Test Case ID: TC-CustumerRegister-01**

**Test Case Title:** Verify user registration with valid data

**Test Objective**: Ensure that a new user can register successfully.

**Preconditions:** User email must not already exist in the system.

**Test Data:** Username: newuser, Email: newuser@example.com, Password: StrongPass123

**Test Steps:**

* - Send a POST request to /api/auth/custumerregister with valid user data.
* - Observe the response.

**Expected Result:** User should receive a success response and be registered in the database.

**Actual Result:** User registered successfully.

**Postconditions:** User is created and can log in.

**Status:** Pass

Remarks:

## Test Case ID: TC-RenterRegister-01

**Test Case Title:** Verify user registration with valid data

**Test Objective**: Ensure that a new user can register successfully.

**Preconditions:** User email must not already exist in the system.

**Test Data:** Username: newuser, Email: newuser1@example.com, Password: StrongPass123

**Test Steps:**

* - Send a POST request to /api/authrenter/register with valid user data.
* - Observe the response.

**Expected Result:** User should receive a success response and be registered in the database.

**Actual Result:** User registered successfully.

**Postconditions:** User is created and can log in.

**Status:** Pass

Remarks:

## Test Case ID: TC-AdminRegister-01

**Test Case Title:** Verify admin registration with valid data

**Test Objective**: Ensure that a new user can register successfully.

**Preconditions:** User email must not already exist in the system.

**Test Data:** Username: newuser, Email: admin@example.com, Password: StrongPass123

**Test Steps:**

* - Send a POST request to /api/auth/register with valid user data.
* - Observe the response.

**Expected Result:** Admin should receive a success response and be registered in the database.

**Actual Result:** Admin registered successfully.

**Postconditions:** Admin is created and can log in.

**Status:** Pass

Remarks:

**Test Case ID: TC-User-CRUD-01**

**Test Case Title:** Retrieve user profile by ID

**Test Objective**: Ensure that a user profile can be retrieved using a valid user ID.

**Preconditions:** User must exist in the database.

**Test Data:** UserID: 507f1f77bcf86cd799439011

**Test Steps:**

* - Send a GET request to /api/users/507f1f77bcf86cd799439011.
* - Observe the response.

**Expected Result:** User profile data is returned.

**Actual Result:** User profile retrieved successfully.

**Postconditions:** User data is displayed.

**Status:** Pass

**Test Case ID: TC-Updatevehycle-CRUD-02**

**Test Case Title:** Update vehycle

**Test Objective**: Ensure that a vehycle can be updated.

**Preconditions:** vehycle must exist in the database.

**Test Data:** UserID: 507f1f77bcf86cd799439011, New Name: Updated Name

**Test Steps:**

* - Send a PUT request to /api/users/vehycle/507f1f77bcf86cd799439011 with updated data.
* - Observe the response.

**Expected Result:** vehycle is updated and new data is returned.

**Actual Result:** vehycle updated successfully.

**Postconditions:** vehycle data is updated in the database.

**Status:** Pass

Remarks:

**Test Case ID: TC-AddVehycle-01**

**Test Case Title:** Add a Vehycle

**Test Objective**: Ensure that a valid data can be uploaded successfully.

**Preconditions:** User must be authenticated.

**Test Data:** File: test-document.pdf (1MB, application/pdf)

**Test Steps:**

* - Send a POST request to /api/vehycle /addvehycle with a valid file and authentication token.
* - Observe the response.

**Expected Result:** Vehycle is added and a success message is returned.

**Actual Result:** Vehycle is added successfully.

**Postconditions:** Data is stored on the server.

**Status:** Pass

Remarks:

## Test Case ID: TC-DeleteVehycle-01

**Test Case Title:** Delete a Vehycle

**Test Objective**: Ensure that a selected vehycle can be delete successfully.

**Preconditions:** vehycle data must be authenticated.

**Test Steps:**

* - Send a POST request to /api/vehycle /deletevehycle
* - Observe the response.

**Expected Result:** Vehycle is Deleted and a success message is returned.

**Actual Result:** Vehycle is Deleted successfully.

**Postconditions:** Data is deleted on the server.

**Status:** Pass

Remarks:

## Test Case ID: TC-FileUpload-01

**Test Case Title:** Upload a valid file

**Test Objective**: Ensure that a valid file can be uploaded successfully.

**Preconditions:** User must be authenticated.

**Test Data:** File: test-document.pdf (1MB, application/pdf)

**Test Steps:**

* - Send a POST request to /api/files/upload with a valid file and authentication token.
* - Observe the response.

**Expected Result:** File is uploaded and a success message is returned.

**Actual Result:** File uploaded successfully.

**Postconditions:** File is stored on the server.

**Status:** Pass

Remarks:

## Test Case ID: TC-FileUpload-02

**Test Case Title:** Reject invalid file type

**Test Objective**: Ensure that uploading an invalid file type is rejected.

**Preconditions:** User must be authenticated.

**Test Data:** File: test.exe (10MB, application/x-msdownload)

**Test Steps:**

* - Send a POST request to /api/files/upload with an invalid file type.
* - Observe the response.

**Expected Result:** Error message indicating invalid file type is returned.

**Actual Result:** Upload rejected with error: Invalid file type.

**Postconditions:** File is not stored.

**Status:** Pass

Remarks:

## Test Case ID: TC-DB-01

**Test Case Title:** Database connection success

**Test Objective**: Ensure that the backend connects to the database successfully.

**Preconditions:** Database server is running.

**Test Data:** Database URI: mongodb://localhost:27017/test-db

**Test Steps:**

* - Start the backend server.
* - Observe the server logs for database connection status.

**Expected Result:** Server logs indicate successful connection to the database.

**Actual Result:** Database connected successfully.

**Postconditions:** Backend is ready to handle requests.

**Status:** Pass

Remarks:

**Test Case ID: TC-Middleware-01**

**Test Case Title:** Verify JWT authentication middleware

**Test Objective**: Ensure that protected routes require a valid JWT token.

**Preconditions:** User must be registered and have a valid token.

**Test Data:** JWT Token: valid-jwt-token

**Test Steps:**

* - Send a request to a protected route with a valid JWT token.
* - Observe the response.

**Expected Result:** Access is granted to the protected resource.

**Actual Result:** Access granted with valid token.

**Postconditions:** User accesses the resource.

**Status:** Pass

Remarks:

## Test Case ID: TC-API-01

**Test Case Title:** Get all resources

**Test Objective**: Ensure that all resources can be retrieved via the API.

**Preconditions:** Resources exist in the database.

**Test Data:** None

**Test Steps:**

* - Send a GET request to /api/resources.
* - Observe the response.

**Expected Result:** List of resources is returned.

**Actual Result:** All resources retrieved successfully.

**Postconditions:** Data is displayed to the user.

**Status:** Pass

Remarks:

# TC-CustomerReport-01: Verify customer report generation

**Test Objective**: Ensure that customer report is generated correctly for a valid customer.

**Preconditions:** Customer data exists in the system.

**Test Data:** CustomerID: 12345

**Test Steps:**

1. 1. Navigate to the customer report section.
2. 2. Enter valid CustomerID.
3. 3. Click on 'Generate Report' button.

**Expected Result:** A detailed report of the customer is displayed.

**Actual Result:** Report displayed successfully.

**Postconditions:** Customer report is available for view or download.

**Status:** Pass

# TC-Booking-01: Verify vehicle booking

**Test Objective**: Ensure that a vehicle can be booked successfully.

**Preconditions:** Vehicle and user must be available.

**Test Data:** VehicleID: V123, UserID: U456

**Test Steps:**

1. 1. Login as user.
2. 2. Navigate to available vehicles.
3. 3. Click 'Book Now' on a vehicle.
4. 4. Fill booking details and confirm.

**Expected Result:** Booking confirmation message is displayed.

**Actual Result:** Booking successful.

**Postconditions:** Vehicle is marked as booked.

**Status:** Pass

# TC-Estimate-01: Estimate vehicle cost

**Test Objective**: Ensure cost estimation is correct for selected vehicle.

**Preconditions:** Vehicle exists in database.

**Test Data:** VehicleID: V123, Duration: 5 days

**Test Steps:**

1. 1. Select a vehicle.
2. 2. Enter duration of use.
3. 3. Click 'Estimate Cost'.

**Expected Result:** Correct cost estimate is displayed.

**Actual Result:** Estimate shown successfully.

**Postconditions:** User is informed of expected charges.

**Status:** Pass

# TC-CityCarData-01: Retrieve city and car data

**Test Objective**: Ensure city and car data loads correctly.

**Preconditions:** Database contains city and car data.

**Test Data:** None

**Test Steps:**

1. 1. Navigate to 'Browse Vehicles'.
2. 2. Wait for city and car dropdowns to load.

**Expected Result:** City and car options are populated.

**Actual Result:** Data loaded successfully.

**Postconditions:** User can select options from dropdown.

**Status:** Pass

# TC-SearchFilter-01: Search or filter vehicles

**Test Objective**: Ensure that vehicle filtering works properly.

**Preconditions:** Vehicles available in the system.

**Test Data:** City: Lahore, Type: SUV

**Test Steps:**

1. 1. Go to vehicle search page.
2. 2. Apply filters: City = Lahore, Type = SUV.
3. 3. Click 'Search'.

**Expected Result:** Filtered list of vehicles is shown.

**Actual Result:** Filter applied correctly.

**Postconditions:** Only matching vehicles are displayed.

**Status:** Pass

# TC-Chat-01: Verify group chat feature

**Test Objective**: Ensure users can send and receive messages in group chat.

**Preconditions:** User is logged in and added to group.

**Test Data:** Message: 'Hello group!'

**Test Steps:**

1. 1. Open group chat.
2. 2. Type and send message.
3. 3. Check for new incoming messages.

**Expected Result:** Message sent and received in group chat.

**Actual Result:** Group chat is working as expected.

**Postconditions:** Communication maintained.

**Status:** Pass

# TC-Reserve-01: Reserve vehicle

**Test Objective**: Ensure vehicle can be reserved without immediate payment.

**Preconditions:** Vehicle is available.

**Test Data:** VehicleID: V123, UserID: U456

**Test Steps:**

1. 1. Login as user.
2. 2. Navigate to vehicle details.
3. 3. Click on 'Reserve'.
4. 4. Confirm reservation.

**Expected Result:** Vehicle reservation confirmed.

**Actual Result:** Vehicle reserved successfully.

**Postconditions:** Vehicle status updated to reserved.

**Status:** Pass

# TC-RatingAdd-01: Add vehicle rating

**Test Objective**: Ensure users can rate vehicles.

**Preconditions:** User has completed a ride.

**Test Data:** VehicleID: V123, Rating: 5 stars

**Test Steps:**

1. 1. Navigate to 'My Rides'.
2. 2. Click on a completed ride.
3. 3. Submit a rating.

**Expected Result:** Rating is saved and confirmation message shown.

**Actual Result:** Rating added successfully.

**Postconditions:** Rating is stored in system.

**Status:** Pass

# TC-RatingShow-01: Show vehicle ratings

**Test Objective**: Ensure ratings are visible to users.

**Preconditions:** Ratings exist for vehicle.

**Test Data:** VehicleID: V123

**Test Steps:**

1. 1. Open vehicle details page.
2. 2. Scroll to ratings section.

**Expected Result:** List of ratings and average shown.

**Actual Result:** Ratings displayed correctly.

**Postconditions:** Users can view feedback.

**Status:** Pass

## Test Case ID: TC-RenterReport-01

**Test Case Title:** Verify renter report generation

**Test Objective**: Ensure that renter report is generated successfully.

**Preconditions:** Renter has completed at least one booking.

**Test Data:** Renter ID: renter123

**Test Steps:**

1. 1. Login as renter.
2. 2. Navigate to the reports section.
3. 3. Request the report generation.

**Expected Result:** System generates and displays renter report successfully.

**Actual Result:** System generates and displays renter report successfully.

**Postconditions:** Report is available for view or download.

**Status:** Pass

Remarks:

## Test Case ID: TC-ReceiveBooking-01

**Test Case Title:** Verify booking reception by renter

**Test Objective**: Ensure that booking request is received by renter.

**Preconditions:** Booking has been made by customer.

**Test Data:** Booking ID: BKG12345

**Test Steps:**

1. 1. Login as renter.
2. 2. Check notification panel or booking requests.
3. 3. Open received booking request.

**Expected Result:** Renter receives and views booking request successfully.

**Actual Result:** Renter receives and views booking request successfully.

**Postconditions:** Renter has access to booking details.

**Status:** Pass

Remarks:

## Test Case ID: TC-ShowBooking-01

**Test Case Title:** Verify display of booking details

**Test Objective**: Ensure that booking details are displayed correctly.

**Preconditions:** Booking must exist in system.

**Test Data:** Booking ID: BKG12345

**Test Steps:**

1. 1. Login as customer or renter.
2. 2. Navigate to bookings section.
3. 3. Select a specific booking.

**Expected Result:** System displays complete booking details.

**Actual Result:** System displays complete booking details.

**Postconditions:** User reviews booking details.

**Status:** Pass

## Test Case ID: TC-ConfirmBooking-01

**Test Case Title:** Verify booking confirmation by renter

**Test Objective**: Ensure that renter can confirm booking successfully.

**Preconditions:** Booking request has been received.

**Test Data:** Booking ID: BKG12345

**Test Steps:**

1. 1. Login as renter.
2. 2. Navigate to booking requests.
3. 3. Click on confirm booking button.

**Expected Result:** Booking is confirmed successfully.

**Actual Result:** Booking is confirmed successfully.

**Postconditions:** Booking status is updated to confirmed.

**Status:** Pass

Remarks:

## Test Case ID: TC-Error-01

**Test Case Title:** Handle 404 error for invalid route

**Test Objective**: Ensure that accessing an invalid route returns a 404 error.

**Preconditions:** None

**Test Data:** Route: /api/invalidroute

**Test Steps:**

* - Send a GET request to /api/invalidroute.
* - Observe the response.

**Expected Result:** 404 error message is returned.

**Actual Result:** Received 404 error for invalid route.

**Postconditions:** User is informed of invalid route.

**Status:** Pass

**Test Case ID: TC-Security-01**

**Test Case Title:** Prevent SQL injection

**Test Objective**: Ensure that the backend sanitizes input to prevent SQL injection.

**Preconditions:** User input is accepted by the API.

**Test Data:** Input: "; DROP TABLE users; --

**Test Steps:**

* - Send a request with SQL injection payload in input fields.
* - Observe the response and database state.

**Expected Result:** Input is sanitized and no SQL injection occurs.

**Actual Result:** Input rejected or sanitized. Database intact.

**Postconditions:** Database remains intact.

**Status:** Pass

Remarks:

## Test Case ID: TC-Performance-01

**Test Case Title:** Handle concurrent requests

**Test Objective**: Ensure that the backend can handle multiple concurrent requests.

**Preconditions:** Backend server is running.

**Test Data:** 100 concurrent requests

**Test Steps:**

* - Send 100 concurrent requests to the API.
* - Measure response times and success rates.

**Expected Result:** All requests are handled within acceptable response time.

**Actual Result:** All requests processed within acceptable time. Server stable.

**Postconditions:** Backend remains stable.

**Status:** Pass

Remarks: