**Title: NeuroFleetX – AI Driven Urban Mobility Optimization**

**Vehicle & Trip Management**

**Module - 2**

**Vehicle & Trip Management**

**Validation & Testing:**

**1. Introduction:**

Module 2 manages vehicles and trips with CRUD operations. To ensure reliability, validation rules were applied, and functional testing was conducted on both backend APIs (Spring Boot + MySQL) and frontend (React dashboard).

**2. Validation Rules:**

**Vehicle Management:**

* Vehicle Registration Number must be unique & not empty.
* Vehicle Model and Type are mandatory.
* Status must only be: Idle, Enroute, Maintenance.
* Location (Latitude/Longitude) must be within valid geographic ranges.

**Trip Management:**

* A trip must link to an existing vehicle.
* A trip can only start if vehicle status = Idle.
* End time > Start time always.
* Distance traveled > 0.
* Trip status must be either Active or Completed.

**3. Test Cases:**

**Vehicle Management – Passed Cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Scenario** | **Input** | **Expected Result** | **Status** |
| VM-01 | Add vehicle with valid details | RegNo=TN09AB1234, Model=EV, Type=Car | Vehicle created successfully | Pass |
| VM-02 | Update vehicle status | Idle → Maintenance | Vehicle status updated | Pass |
| VM-03 | Delete vehicle | Vehicle ID=5 | Vehicle removed | Pass |

**Vehicle Management – Failed Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Scenario** | **Input** | **Expected Result** | **Actual Result** | **Status** |
| VM-F1 | Add vehicle with duplicate RegNo | TN09AB1234 | Error: “Vehicle already exists” | Error displayed | Fail handled |
| VM-F2 | Add vehicle without RegNo | Blank field | Error: “Registration number required” | Error displayed | Fail handled |
| VM-F3 | Update vehicle with invalid status | Status=Flying | Error: “Invalid status” | Error displayed | Fail handled |

**Trip Management – Passed Cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Scenario** | **Input** | **Expected Result** | **Status** |
| TM-01 | Start trip for idle vehicle | Vehicle=IDLE | Trip started successfully | Pass |
| TM-02 | End trip with valid details | End > Start | Trip completed & distance saved | Pass |
| TM-03 | View trip history | Vehicle=2 | List of trips displayed | Pass |

**Trip Management – Failed Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Scenario** | **Input** | **Expected Result** | **Actual Result** | **Status** |
| TM-F1 | Start trip for enroute vehicle | Vehicle=ENROUTE | Error: “Vehicle already in trip” | Error displayed | Fail handled |
| TM-F2 | End trip with invalid time | End < Start | Error: “Invalid end time” | Error displayed | Fail handled |
| TM-F3 | Start trip for non-existent vehicle | Vehicle ID=999 | Error: “Vehicle not found” | Error displayed | Fail handled |
| TM-F4 | Record trip with negative distance | Distance=-10 km | Error: “Invalid distance” | Error displayed | Fail handled |

**4. Testing Approach:**

* **Unit Testing** → Each API (/api/vehicles, /api/trips) tested individually.
* **Integration Testing** → Frontend React components tested with backend API responses.
* **Validation Testing** → Ensured invalid inputs are rejected.
* **UI Testing** → Verified dashboard updates when vehicles/trips are added/updated.
* **Database Testing** → Checked MySQL consistency (unique RegNo, valid trip logs).

**5. Results & Observations:**

* All valid cases passed successfully.
* Invalid inputs triggered proper error messages.
* CRUD operations performed reliably.
* Trip and vehicle data stayed consistent in database.
* System ready for scaling & AI enhancements.