**Test Plan for EV Motors Application**

**Project Name:** EV Motors  
**Tester:** Chinu Tyagi  
**Application Type:** C# Desktop with Database  
**Tools Used:**

* **Manual Testing:** Excel
* **Database:** SQL Server Management Studio
* **Integration Tests:** NUnit/xUnit, Postman, Moq
* **Functional Testing:** NUnit/xUnit, Postman, Moq

**1. Test Plan Overview**

This document outlines the testing strategy for the EV Motors application. Our goal is to verify the functionality, integration, and overall performance of a C# desktop application that interacts with a SQL Server database. The following sections describe the test scope, types, data requirements, entry/exit criteria, risk management, and deliverables for the project.

**2. Test Scope**

**In Scope:**

* **Manual Testing of UI:** Functional & usability tests covering all critical interactions.
* **Database Verification:** CRUD operations, data integrity, and constraints.
* **Integration Testing:** Testing the communication between the UI, business logic layer (BLL), and the database.
* **Functional Testing:** Automated and manual verification of core functionality to ensure the business requirements are met.

**Out of Scope:**

* Third-party tool testing (unless it directly affects application functionality)
* Hardware-specific testing (unless specified)

**3. Test Types**

**A. Manual UI Testing**

**Scenarios & Test Cases:**

* **Login Page:** Verify login with valid and invalid credentials.
* **Form Validation:** Ensure that required fields display appropriate validation messages.
* **Button Actions:** Verify that clicking the "Save" button stores data correctly.
* **Navigation:** Ensure menus, links, and back buttons function as expected.
* **Error Handling:** Confirm proper error messages are shown if the server or DB fails.

**Tools:**

* Manual testing, with test cases documented in Excel.

**Checkpoints:**

* Layout and alignment, control responsiveness, correct labeling, and accessibility (e.g., tab order, keyboard navigation).

**B. Database Testing**

**Scenarios:**

* **Data Insertion:** Validate that data entered via the UI appears correctly in the database.
* **Data Retrieval:** Ensure SELECT queries return accurate data.
* **Data Update/Delete:** Verify update and delete operations work as intended, including logging where applicable.
* **Stored Procedures/Triggers:** Ensure stored procedures and triggers behave correctly.
* **Foreign Key Constraints:** Test system behavior when invalid references are inserted.
* **Transaction Rollbacks:** Confirm atomicity and consistency of transactions on failures.

**Tools:**

* SQL Server Management Studio (SSMS)

**Checkpoints:**

* Schema correctness, default values, indexing, performance, and referential integrity.

**C. Integration Testing**

**Components & Test Cases:**

* **UI to BLL:** Validate that the UI’s input is correctly processed by the business logic layer.
* **BLL to DB:** Verify that the business logic layer communicates appropriately with the database (insert, read, update, delete).
* **External APIs:** Confirm external service calls return expected data.
* **Exception Handling:** Ensure integration layers handle exceptions gracefully.

**Tools:**

* NUnit/xUnit for test automation, Postman for API testing, and Moq for mocking dependencies.

**Checkpoints:**

* End-to-end data flow, proper logic enforcement, and robust error handling when individual components fail.

**D. Functional Testing**

**Objective:**  
Ensure the application’s core functionalities meet the defined business requirements, both through automated tests where applicable and through targeted manual tests.

**Scenarios & Test Cases:**

* **Business Process Flows:**
  + End-to-end workflow testing from data entry in the UI to data processing in the business logic layer and storage in the database.
  + Validation that business rules (e.g., calculation logic, eligibility criteria) are correctly applied.
* **Automated Functional Verification:**
  + Write automated tests using frameworks like NUnit/xUnit that simulate real-world scenarios (e.g., complete user journeys) to verify the correctness of core operations.
  + Examples include verifying that a completed order or transaction produces the expected results across all layers.
* **Error and Edge Case Handling:**
  + Confirm that the application displays proper error messages and recovers gracefully when presented with invalid data or unexpected events.
  + Test scenarios covering boundary conditions such as empty inputs, extremely long data entries, and special characters.

**Tools:**

* Automated test frameworks (NUnit/xUnit) for repeatable functional tests.
* Manual test case verification as documented in Excel for exploratory and boundary cases.

**Checkpoints:**

* Alignment with business requirements.
* Consistent responses between automated test outcomes and manual test observations.
* Correct processing of both expected and unexpected inputs.

**4. Test Data Requirements**

* **Authentication:** Valid and invalid credentials for login testing.
* **CRUD Operations:** Sample data for insert, update, and delete scenarios.
* **Edge Cases:** Data for null values, maximum lengths, special characters, and boundary conditions.
* **External API Responses:** Simulated and corrupt responses (when using mocks for external API calls).

**5. Entry and Exit Criteria**

**Entry Criteria:**

* All application components are built and deployed to the test environment.
* The test database is populated with seed data.
* The UI is accessible and stable.
* The test environment is properly configured.

**Exit Criteria:**

* Execution of all planned test cases.
* Resolution of all critical and major defects.
* A comprehensive Test Summary Report that has been reviewed and approved by stakeholders.

**6. Risk and Mitigation**

**Risk:** Data loss during testing  
**Mitigation:** Perform database backup/restore operations before and after test runs.

**Risk:** Unstable build environment  
**Mitigation:** Validate environment setup and connectivity prior to testing.

**Risk:** UI delays or flakiness  
**Mitigation:** Use verified and stable application versions for testing.

**7. Deliverables**

* **Test Plan Document:** This document.
* **Test Cases:** Detailed in an Excel file.
* **Test Summary Report:** Overall testing results and coverage.
* **Defect Report:** All issues logged and tracked via Jira.