1. **Introduction – Application overview**

The application is a web-based system developed using the MERN stack, comprising a React.js frontend and a Node.js backend API. It provides core functionalities including user registration, authentication, and CRUD (Create, Read, Update, Delete) operations on data entities. The backend RESTful API is built with Express.js and utilizes MongoDB as the database, interfaced through Mongoose ODM.

1. **Functionalities and Testing Methodologies:**

**2.1 Functionalities Covered:** The following core features of the application are included in the scope of automated testing:

**User Authentication:**

* Login with valid and invalid credentials.
* Logout functionality to terminate authenticated sessions.

**Product Management (CRUD Operations):**

* Add Product: Validate the creation of new product entries.
* Edit Product: Test the ability to update existing product data.
* Get Product(s): Verify product retrieval via individual and list endpoints.
* Delete Product: Confirm successful removal of product entries.

**2.2 Methodologies Covered:** To ensure comprehensive and repeatable test coverage, the following testing methodologies and tools are used:

**Functional UI Automation Testing:**

* **Tool:** Playwright JavaScript
* **Framework Integration:** BDD-style testing using Cucumber for human-readable scenarios.
* **Reporting:** Test results are generated in cucumber.html format for easy review.

**API Test Automation:**

* **Tool:** Supertest (for HTTP assertions) with Chai (for assertions)
* **Framework Integration:** BDD using @cucumber/cucumber for step definitions.
* **Reporting:** Automated HTML reports are generated for visibility and traceability of test outcomes.

1. **How To Run Tests:**
   * API Testing:- **Start backend** (cd backend, npm start) ; **Set test tags** - @api; **For execution** – (cd backend, npm run test:full)
   * UI Automation Testing – **Start backend**(cd backend, npm start) ; **Start frontend** ( cd frontend, $env:NODE\_OPTIONS="--openssl-legacy-provider"; npm run build ,

serve -s build) , **Set test tags** - @functional, **For execution** – (cd testremkarthika, npm run test:full) – UI have some existing defects, hence best to start with 1 or 2 data.

1. **Assumptions or Limitations:**

In certain test scenarios, specific data conditions must be met to validate the expected system behavior accurately. The following cases require deliberate **test data setup or teardown** to simulate real-world edge situations.

* + 1. Delete Product When No Products Exist
    2. Get All Products When No Data is Available

1. **Environment and Version Information:**

All other dependency versions (e.g., React, Express, Mongoose, Cucumber, Supertest, etc.) are explicitly defined in the package.json file. Please ensure you use the **exact versions** specified to avoid compatibility issues or unexpected test failures.

* + **Node.js:** v22.15.1
  + **npm:** v10.9.2

1. **Test Execution Report:**
   * **API Testing:**
     1. **Total Test Cases:** 16
     2. **Passed:** 15
     3. **Failed:** 1
   * **UI Testing:**
     1. **Total Test Cases:** 17
     2. **Passed:** 15
     3. **Failed:** 2

Execution Report for this data available in /archieveReports folder