Contents

Task 1:

1. **Prioritization of charters**: As application is mainly focusing on adding expenses and their tracking. Thus, I have started exploratory testing by adding income and expenses, follows by tracking if pie chart is displaying it correctly via percentage etc. and then checking the corresponding balance sheet
   Afterwards I started exploring other areas of application e.g. searching via note, validating that non-premium user can't use paid features, filtering expenses, adding expense in past date etc.
2. **Time Taken in each charter:** It took around ~10 minutes to explore each charter
3. **Risks:** Mentioned in exploratory file i.e. ***Exploratory testing.pdf***

Task 2:

1. **Added Tests:**
   - Added New income and validated same in balance sheet
     Added New expense and validated same in balance sheet
     Validated if Balance is showing correct result i.e. (Income – expense)
   - Added note to expense and validate same via search feature
   - Added test to validate monthly filter
   - Other possible tests (not automated yet)
     - Validate Pie chart
     - Validate other filters
     - Validate paid features are not accessible to non-premium user
     - Validate adding income and expense in past date
2. **Technology stack:**
   - **Appium (Automation tool)**
   - **Maven (Project building tool)**
   - **TestNG (Unit testing framework)**
   - **Java (Programming language)**

Reason for above choice:
As Appium is free and open-source which can automate both Android as well as iOS applications, enabling testing of native & hybrid mobile applications
Also, it provides support for multiple programming languages

TestNG easily lets us create suite files and can be invoked via terminal
Building tool can be either maven or gradle, I have used maven in this framework

Framework is created using Page Object Model and other design patterns
Execution flow starts from test itself

3. **Framework:**
    - **mobile-auto**

Task 3:

1. **Added Tests:**
    - Added test to validate system's health
    - Added test to create, update and delete product & also validate same by fetching its detail
    - Added negative tests like creating product via insufficient details, fetching and deleting invalid product
    - Other possible tests (not automated yet)
        - Validate above scenarios for other items like store, category etc.

2. **Technology stack:**
    a. **REST Assured (Java Library to test API endpoints)**
    b. **Maven (Project building tool)**
    c. **JUnit (Unit testing framework)**
    d. **Cucumber (BDD)**
    e. **Java (Programming language)**

Reason for above choice:
REST Assured removes a lot of boilerplate code which is required to set up HTTP connection
Cucumber is used as it is helpful for business stakeholders who do not have sufficient knowledge or do not want to get into code details

Execution flows start from feature files, then goes to step definitions and finally calls the Manager classes of respective feature e.g. Product, HealthCheck etc.

3. **Framework:**
   - **api-auto**