Test Plan

Team Members

Matthew Kahl | Gurinder Kaur | Andrew Lea-Wilson | Nicholas Lee

Sarah Lozier | Sean Mafnas | Bea Samantha Magno

Table of Contents

Scope

The **Cash Control** project is a Java-based application aimed to help users manage personal finances through a simple and effective platform. It will be a locally hosted solution for expense tracking and financial oversight, emphasizing privacy, and long-term utility. The goal of this testing effort is to verify that the application meets the user requirements and functions correctly in its local hosting station.

Objectives

The primary objective for this testing effort is to ensure the project system maintains its scope of offline accessibility, data privacy, and user-friendly interaction.

Specific Objectives Include;

- Ensuring user interface for ease of use in login
- Supporting basic financial functionality of overviewing transactions
- Identifying defects and report bugs that affect the performance or functionality of the system
- Error handling
- Verifying quality of code in the backend and frontend development

Strategy

Primary tools used to test will be with **JUnit**, **Jest**, and the **React Testing Library**. These will help gauge the status of the system through performance, usability, integration, and functionality.

- Unit Testing for key transaction endpoints potentially using JUnit
- API Testing with Jest and React for Backend development
- Backend database checks
- User Interface Login form
- User Experience in the Transactions and Summaries pages
- Verify user interface functionality in the Frontend
- Verify database Functionality

Boundaries and Limitations

To ensure the project's features are functioning correctly and optimized there will be items out of the project's scope.

Local hosting: There is no deployment to any external servers or cloud platforms.

No user authentication: There will not be any implementation for login security mechanisms such as passwords or session management. Users will be identified by name and email only.

Desktop compatibility: The program will be optimized for desktop environments only on most browsers and no mobile design or testing will be produced.

Procedures

The process for testing will cover;

Preparing Test data:

- Create mock data for user accounts' login information
- Create mock data for respective transaction data

Executing the Test Cases:

- Run Unit tests for the backend components using JUnit
- Runt Unit tests for the frontend components using Jest and React Testing Library
- Run API testing for backend development
- Run Frontend user testing for quality assurance

Tracking Defects:

- Log defects identified and report in shared document
- Resolve any bugs or issues and assign priority of resolution
- Retest resolutions and report any additional errors

Test Cases

Frontend User Interface / User Experience Testing

UI/UX testing focuses on each component of the frontend functioning smoothly, properly, and as expected for each user interaction. Tests will be written using **Jest** with the implementation of the **React Testing Library**.

Test Case: NavBar

What is being tested:	**Expected result:**
The function of the navigation bar.	Displays the front page of images and menu options smoothly showcasing the structure of the login and logout pages.

Test Case: LoginForm

What is being tested:	**Expected result:**
Input of the login menu for the user.	Handles the user login page with the name and email of the user, transitioning to the dashboard upon logging in.

Test Case: GetStartedForm

What is being tested:	**Expected result:**
Input for the new user registration sign-up.	Allows the user to make an account with their name and email, transitioning them to the dashboard upon signing up with empty results to be filled in.

Test Case: UserPortal

What is being tested:	**Expected result:**
Display of the main dashboard upon logging in.	Showcases the main dashboard of the summaries chart for the user and

	displays the proper buttons for logout and the transactions form and history.

Test Case: TransactionsForm

What is being tested:	**Expected result:**
Function for the user to input transaction information.	Allows the user to input the description of the transaction, date, description, and value in addition to labeling the transaction as an income or expense.

Test Case: TransactionsList

What is being tested:	**Expected result:**
Display of the past transaction history and inputs.	Accurately showcases the past inputs of transactions in chronological order with abilities to filter or delete any transactions.

Test Case: SummariesChart

What is being tested:	**Expected result:**
Display of transactions total based on time.	Accurately calculates and displays transaction values in a summary for the Month, Year, past 30 days, and past 365 days.

Backend Unit/API Testing

UNIT/API testing ensures the accuracy and reliability of the REST API Endpoints for stable integration with the frontend.

Test Case: JUnit: UserController

What is being tested:	**Expected result:**

User Registration, Login and Retrieval of User data	New users will be successfully created and stored in the database with valid login credentials. Retrieving a user by ID returns correct user details.

Test Case: JUnit: TransactionController

What is being tested:	**Expected result:**
Add, retrieve, and delete transaction data	Transactions are created and associated with the correct user. Requests for a user's transaction history return accurate data in TransactionsList and SummariesChart on the frontend. Transactions can be successfully deleted, and deleted entries no longer appear in future queries.

Schedule

Task	**Description**	**Est Time**
Implement Backend Unit Testing	Written with JUnit	2 days
Backend API Testing	Key endpoints - creation, deletion JUnit tests for Functions	3-5 days
Frontend User Interface/QA	Finances / filters/summaries login/home button functionality	2 days
Bug Logs	Log bug and set priorities assigning frontend and backend resolutions	1 week
Retesting	Retest resolved bugs for QA	2-3 days

Resources

Resources for testing will be software dependent based on the testing team.

Software Tools and Technologies

- **IDE/Code Editors:** VSCode, IntelliJ IDEA, Eclipse
  per developer preference)
- **Frontend Framework:** React.js, Axios, Jest, React Testing Library
- **Backend Framework:** Java with Spring Boot, H2 Database, Junit
- **Version Control:** Git and GitHub
  (GitHub organization: CMSC495-Group2)
- **Collaboration & Communication:** Google Chat, Google Docs, GitHub
- **Testing:** Jest for frontend; JUnit for backend, Word and Google Docs for bug logging

Defect Management

Identifying, logging, prioritizing, resolution, and retesting will be done to mitigate any issues within the system.

- Identify defects in the system upon any test case
- Log any bugs or issues manually in the shared living document
- Prioritize defects to resolve and push through with respective Frontend and Backend development team
- Fix and resolve any issues in a timely manner
- Retest and report any additional fixes or improvisations

Managing the defects through simple phases will ensure smooth integration and API handling of the Cash Control financial tracker programt.