Testing Document and Specification

Test Plan

Commerce Bank Group 3
CS 451R

Reed Heinrich
Landon Volkmann
Samuel Feye
Connor Larson
Tayler Singleton

Introduction

This document outlines the test plan for the Commerce Bank App, created by Group 3. In the Requirements Document, we outlined the needs of this system – this system will be a web application designed for users with an online account with Commerce Bank. It will allow users to see their transactions, sort their transactions by date, export their transactions to a spreadsheet, add/edit/delete notification rules to display which transactions fit under a set criterion inputted by the user, pull/compare notification rules, see the number of times a notification rule has been triggered over the past month and year.

The testing routine will test the user's ability to perform all necessary tasks such as logging in, signing up and creating a new account, viewing and searching through their transactions, adding transactions, viewing their notifications, editing their notification settings, and exporting their transactions to a .csv file.

Terminology

User – people who would use the app on release to view their bank account and only interact with the front end.

Front-end – the application/ what the user sees when they launch the website. Makes calls to the backend to display information to the user in a pleasant, organized fashion.

Back-end – the database that contains all relevant user and account information, as well as routes and code that securely stores information that is accessible through calls from the front-end.

Unit test – a hard-coded test that can be ran to look for and reveal bugs to the developers to eliminate as many flaws as they can. Can be automated or generalized to cover more areas of code where necessary.

Manual test – the only code here is the actual project itself. A manual test is where the developer/tester runs through the application making sure there are no bugs present and that everything loads correctly.

React framework – React is a JavaScript library for building user interfaces. We used React to create our front-end design.

Redux framework – Redux is a JavaScript library for managing the application state. This pairs well with the React library we used.

Items Tested

Items that will be tested during the testing phase will be but are not limited to the following (some Test Cases cover more than one feature):

Unit Testing (found in "test" folder in project):

- Unit Test 1: Makes sure that the correct data was loaded into the database using the .sql files. Specifically, this unit test makes sure all Account info is loaded in correctly
    - Test function: public void test_select_acc_info()
- Unit Test 2: Makes sure that the setTransaction call works in the backend and that all necessary data is loaded correctly.
    - Test function: public void test_set_transaction()
- Unit Test 3: Assures that notification triggers' type and description are inserted correctly.
    - Test function: public void test_notification_trigger_descr()
- Unit Test 4: Assures that notification triggers are actually created and all necessary values are in place.
    - Test function: public void test_create_notification_trigger()

Login Page

- Ability for a user to login via email and password
    - Test Case: 1.1
- Ability for a user to sign up and create their own account through the login page
    - Test Case: 1.2
- Ability for a user to save their email+password combination for next time
    - Test Case: 1.1
- Ability for a user to hit forgot password and have a refresh link sent to their email
    - Test Case: 1.3
- On launch, the Login page is correctly displayed.
    - Test Case: 1.1
- Password is masked and not visible to users
    - Test Case: 1.1

Dashboard

- Upon login, user is taken to dashboard
    - Test Case: 2.1
- Dashboard displays both the "Notifications" section at the top and the "Transactions" section below it

    - Test Case: 2.1
- From the dashboard/home screen, a user has the ability to select the “Transaction” tab and the “Notification Settings” tab on the left side of the screen
    - Test Case: 2.1
- Functional “Logout” button is visible at the top right corner of the screen
    - Test Case: 2.1

Transactions

- When a user clicks on the “Transactions” tab, the transactions list is displayed
    - Test Case: 3.1
- User has the ability to sort transactions by date - either by newest or oldest
    - Test Case: 3.2
- User has the option to turn on/off Dense Padding, which scales down the list to fit more on the screen.
    - Test Case: 3.2
- Each header is displayed correctly above the right columns: ID, Date, Amount, Type of Charge, Balance, and Description
    - Test Case: 3.1
- Each row is selectable - user has ability to select one or all rows and delete the transaction
    - Test Case: 3.2
- User has the ability to search for a transaction in the search field, and they can select which column they are searching through
    - Test Case: 3.2
- User has the ability to add a transaction
    - Test Case: 3.2
- User has the ability to export the transactions to a .csv file
    - Test Case: 3.3
- User has the ability to select the amount of rows they want displayed in one page (default 5)
    - Test Case: 3.2

Notifications

- When a user clicks on the “Notification Settings” tab, they are taken to the Notification Settings screen
    - Test Case: 4.1
- The functionality is the same for all four sections (transaction, balance, description, recurring alert): when the user clicks the “add” button, they are prompted to enter a value (if an amount, it’s a number, if it’s a description, it’s a word/some words)
    - Test Case: 4.1
- If a user no longer wants to receive a type of notification, they have the ability to delete a notification by clicking the ‘X’
    - Test Case: 4.2

Items Not Tested

There are features that will not be included in this round of testing, as we are still polishing and finishing up the application on this final iteration.

- Notification emails
- Some aspects of notifications are still being touched up at the time of testing

Approach

The method to this testing procedure is using a mix of unit testing and manual system testing, as well as an even mix of testing the front-end and back-end to make sure everything is bug free. The unit testing can be found in the project under the “test” folder - the unit tests primarily test the back-end and database connection, making sure all data is loaded correctly, and that all back-end calls such as creating transactions and notification triggers do their jobs. The manual system testing being done primarily tests the front-end and what the user will be seeing, also making sure that the back-end calls that are made display the information correctly. It is worth noting that this testing will be done on April 26th, while our final iteration is due first week of May, so there may be some slight differences between our application now and our final product. System testing will continue, but will not be able to be documented.

Item Pass/Fail Criteria

In designing this project, we have been referring to the project requirements given to us by the Commerce Bank representatives and implementing the required features to the best of our ability. It is our main goal to meet all of these requirements, along with our stretch goals, and have the user be able to use these features effectively with little to no training.

Features that contain defects will fail the testing procedure, will be documented, and then referred to the developers for investigation and revision if necessary.

Test Deliverables

In addition to the Test Plan, other test deliverables include the Test Specification which outlines the specific test cases and expected results of each test, and Test reports which consists of Incidents, Defects and Changes.

Testing Tasks

The following list the testing deliverables and the activities required to produce the deliverable.

Deliverables	Activities
Test Plan	● Analyze Requirements for System Features ● Determine Testable/Non-Testable Features ● Develop Approach/Method for testing ● Determine Task and Estimate Efforts
Test Specifications	● Analyze Requirements ● Define Test Cases for Testable Features as Outlined by the Test Plan
Test reports	● Implement Test Cases as Outlined by the Test Specifications ● Document Incidents and Defects ● Determine Severity of Incidents and Defects ● Determine Changes that Need to be Made to System ● Document and Submit Change Request to Developer