System Verification and Validation Plan

ShareWize: Expense Sharing Web App

Group 24

Arun Rathaur

Ayaz Lockhat

Mubtasim Rahman

Veeral Patel

Feb 7, 2024

ShareWize

SHARING MADE SIMPLE

Our Team

Name	Email	Role
Arun Rathaur	rathaa1@mcmaster.ca	Developer (Data and Security)
Ayaz Lockhat	lockhata@mcmaster.ca	Developer (Front-End)
Mubtasim Rahman	rahmam91@mcmaster.ca	Developer (User Interface and Design)
Veeral Patel	patev18@mcmaster.ca	Developer (Back-End)

Revision History

Version	Date	Change	Revision
0	Feb 7, 2024	Added Key info	1
1.0	Feb 8, 2024	Updated info	2
1.1	Feb 9, 2024	Cleaned up grammar and info	3

Purpose of Project

1.1. Goal of the Project

The ShareWize Expense Sharing Web App aims to provide a user-centric solution for simplifying bill splitting and expense tracking within social circles. With a focus on ease of use and financial transparency, our goal is to empower users to efficiently manage shared expenses among friends, roommates, and other groups. Leveraging web-based technology, our solution prioritizes adaptability for seamless mobile usage. Previous documents, including the Software Requirements Specification (SRS) and Design, outline the project's detailed specifications and architectural considerations. Our primary objective, as a service goal, is to provide a highly efficient, user-friendly, and supportive platform that fosters harmonious financial

interactions among friends, roommates, and social circles. As we continue to evolve, we may also explore revenue goals to sustain and enhance the service for our users.

Components Test Plan

1.1. Authentication:

1.1.1. Unit Tests:

- We can test the creation of accounts by providing user data and verifying a successful creation. This could mean by identifying user data in the connected database or psychically logging in.
- Test login functionality by providing valid credentials and ensuring successful authentication.
- Test the password recovery feature by initiating a password recovery request and verifying the updated password in the database. Also by confirming the email communication was successful and the user can login with the new password and not the old.

1.1.2. Performance Tests and Metrics:

- Measure the time it takes to create an account, to login to an account, to reset a password, etc.
- Measure the latency of the login process under varying concurrent user loads to ensure responsiveness. Make sure the database is able to handle multiple login/creations at a time.
- Measure the accuracy of logging in.

1.2. Bill Splitting Logic:

1.2.1. Unit Tests:

- Test to make sure the group creation feature is functional. Can add valid data and verify successful creation with the database and UI.
- Test expense addition by adding expenses to the group. Must also ensure that the balances are accurate for each user and the calculations are correct. Needs to be able to handle the maths for large amounts of groups/money.
- Test settlement functionality by settling balances within groups and verifying correct adjustments.

1.2.2. Performance Tests and Metrics:

- Measure the time efficiency of calculating expenses, settlements, and bill splitting calculations between bigger groups.
- Measure how long it takes to create a group and accept invitations to the group.
- Measure the calculations to ensure correct values.
- Measure database additions and UI additions (for creating a group)

1.3. Data Visualization:

1.3.1. Unit Tests:

- Test if data is visually correct.
- Provide sample data to verify accurate creation of charts and values.
- Test if calculations on the charts are correct and align with the provided data from users.

1.3.2. Performance Tests and Metrics:

- Measure the time it takes for a chart to generate for different types of charts, different sizes, different amounts of information, etc.
- Measure the amount of memory the visualization data generates. Ensure efficient utilization.

1.4. Expense Tracking Module:

1.4.1. Unit Tests:

- Test expense recording by adding expenses with different values, different formats ($0.01, $100.10, $100, etc), and other attributes. Ensure they are accurate in storage.
- Test the editing and deleting of expenses to ensure they are being modified accurately and verifying the proper updates in calculations it relates to (data visualization, group expenses, image attachments).
- Image functionality, ensure the images attached to the expenses are stored and displayed correctly.

1.4.2. Performance Tests and Metrics:

- Measure the time it takes for expense recording, editing, and deletion operations under concurrent loads.
- Measure the time it takes to upload/download the image attachments.

1.5. Database:

1.5.1. Unit Tests:

- Test data storage functionality by storing various types of data (user profiles, authentication data, expenses, groups) and verifying successful retrieval.
- Test data retrieval functionality by querying stored data with different criteria and ensuring accurate results.

1.5.2. Performance Tests and Metrics:

- Test the time taken for the data storage to retrieve operations under heavier loads.
- Test the time taken for the database to respond to different queries to ensure optimal performance.

Other tests may include: Integration tests to measure functionality and interactions between other components, User Acceptance Testing which would involve users and stakeholders to verify real world scenarios, Load testing to assess the systems performance and stability, and Data Integrity testing to make sure data is processed and backup correctly. These tests will help evaluate the systems functionality.