Accountill: Invoicing Application

Software Testing Project Plan

EEC 623/CIS 636 Software Quality Assurance

Proposed by

Anubhuti Dayal - 2824826
Shruthi Gangineni - 2826027
Suman Kumar - 2861070

SOFTWARE TESTING PROJECT PLAN

1 Introduction

1.1 Project scope

As a part of this project we are planning to test accounting software "**Accountill: Invoicing Application**", it is an open-source web application. This is designed to help users manage their personal and business finances. The software provides features such as expense tracking, invoicing, and reporting, to help users monitor their cash flow.

The application is built using modern web technologies such as Node.js, React, and Redux, and is licensed under the MIT License, which allows users to use, modify, and distribute the software freely.

This is a Web-based application, and the source code can be found here: https://github.com/panshak/accountill

The major inputs of Accountill include user data such as total payment received, pending amount, and total amount, paid invoices, partially paid and unpaid invoices, and financial transactions. The software processes this data using algorithms and formulas to generate reports, graphs, and charts that provide insights into the user's financial status.

The software also provides tools for creating invoices, and managing accounts, which help users to manage their finances more efficiently.

1.2 Major software functions & Technologies [1]

Functional decomposition of the application can be done as below:

- User Management:

  1. Registering new users.
  2. Authenticating users using JWT token and Google OAuth.
  3. Managing user accounts and profiles.
  4. Resetting user passwords.

- Customers Management:

  1. Adding new customers once logged in.
  2. Performing CRUD over customer data.

- Invoice Management:

    1. Creating new invoices.
    2. Editing existing invoices.
    3. Adding items to invoices.
    4. Deleting items from invoices.
    5. Displaying invoice details.
    6. Adding and displaying notes to invoices.

- Email and PDF Invoice Generation:

    1. Generating PDF invoices, receipts, estimates, quotations, and bills.
    2. Sending invoices, receipts, estimates, quotations, and bills via email.
    3. Downloading invoices, receipts, estimates, quotations, and bills in PDF.

- Setting due dates for invoices:

    1. Automatically changing invoice status when payment is recorded.
    2. Recording payment history for each invoice.
    3. Allowing partial payment of invoices.
    4. Displaying payment history and invoice status

- Reporting and Analytics:

    1. Generating financial reports by generating charts.
    2. Providing insights into financial data in the admin dashboard like displaying all invoice statistics including total amount received, total pending, recent payments, total invoice paid, total unpaid and partially paid invoices.

2 Major software technologies [1]

This project was built with the MERN stack (MongoDB, Express, React and NodeJS) [1], please see details of the technologies used in this project.

Client

- React JS
- Redux (for managing and centralizing application state)
- React-router-dom (To handle routing)
- Axios (for making api calls)
- Material UI & CSS Module (for User Interface)
- React simple Snackbar (To display success/error notifications)
- Cloudinary (to allows users to upload their business logo)
- Apex Charts (to display payment history)

- React-google-login (To enable authentication using Google)

Server

- Express
- Mongoose
- JWT (For authentication)
- bcryptjs (for data encryption)
- Nodemailer (for sending invoice via email)
- html-pdf (for generating invoice PDFs)
- Database
- MongoDB (MongoDB Atlas)

2.1 Tasks

- **Installation and Setup:**

  The application is already hosted by the developer https://accountill.com we will be using the same deployed environment for our manual testing.

- **Set up a testing environment:**

  We will be creating one account at https://accountill.com and the same will be used to perform the manual testing on the application by all our team members.

- **Write test driver/stub:**

  As we are following the manual testing approach, we don't have to design such test driver/stubs but the test cases need to be designed which is discussed in the next section.

- **Design test cases:**

  By designing test cases, it would be easier to test the website comprehensively and ensure that all the important features are working as expected. The tasks for test cases could be divided in several ways such as:
  Functionality Testing focuses on its key features, including user registration, login, account creation, and transaction recording.

  User Interface Testing focuses on the website's usability, navigation, and responsiveness on different devices and browsers.

Security Testing assesses the website's security measures, such as encryption, firewalls, and user authentication.

Performance Testing evaluates the website's performance under various load conditions to identify any performance issues.

Compatibility Testing ensures the website functions correctly on different browsers, operating systems, and devices.

Integration Testing ensures testing the website's integration with external systems, such as payment gateways, email service providers, and social media platforms.

Regression Testing involves re-testing the website after any changes or fixes have been made, to ensure that existing functionality is still working as expected.

We will discuss and finalize our testing strategy along with that will design functional & non-functional requirements test cases based on features of the software.

Our group will divide the tasks so that we don't miss any features to test. We are also planning to review each other works to ensure we cover all the portions of the software.

- **Perform tests:**

  Finally based on our test cases, we will perform the testing for each functional & non functional requirements, it will be system testing.

- **Summarizing the Results :**

  All the results will be finally merged together by all the testers and will be attached in the Final report as Testing Results.

- **Final report :**

  Final report will be prepared by all team members and will contain outcomes of the project, test results summary along with project demonstration and presentation.

2.2 Schedule

This section presents a list of project tasks with their starting and ending dates

Sr. No.	Activity	Start Date	End Date	Status
1	Project finalization	6-Mar-23	13-Mar-23	Done
2	Test plan preparation	13-Mar-23	20-Mar-23	Done
3	Test plan submission	20-Mar-23	20-Mar-23	Done
4	Installation of the software	21-Mar-23	25-Mar-23	yet to start
5	Set up a testing environment	26-Mar-23	1-Apr-23	yet to start
6	Write test driver/stubs	1-Apr-23	8-Apr-23	yet to start
7	Design test cases	8-Apr-23	16-Apr-23	yet to start
8	Perform tests	17-Apr-23	25-Apr-23	yet to start
9	Summarizing results	25-Apr-23	28-Apr-23	yet to start
10	Final report preparation	29-Apr-23	1-May-23	yet to start
11	Final presentation preparation	2-May-23	5-May-23	yet to start
12	Final submission	6-May-23	12-May-23	yet to start

3 References

[1] https://github.com/panshak/accountill