Boston University
Electrical & Computer Engineering
EC464 Senior Design Project II

Final Prototype Testing Plan

AI-Enhanced Pharmacy Procurement

by

Team 6
PAPO Group

Team Members

Joel Akerman akermanj@bu.edu
Taha Ababou hababou@bu.edu
Manuel Segimon manuelsp@bu.edu
Bora Bulut bbulut@bu.edu
Zaiyan Muhammad monem@bu.edu

1. Introduction

The final prototype represents a seamless integration of database architecture, API functionality, and front-end design, offering an efficient and user-friendly platform. Our secure database ensures the safe storage and easy access of user information, while a suite of APIs facilitates essential user interactions like registration, login, and profile management, streamlining the user experience. At the heart of this prototype is a refined front-end interface, featuring a straightforward login page and a main dashboard. This dashboard employs advanced algorithms for package size analysis and cost evaluation, optimizing the drug selection process. These components work together to provide users with personalized, cost-effective drug replacements, enhancing decision-making and efficiency. This streamlined prototype underscores our commitment to enhancing healthcare outcomes through the strategic integration of technology, offering a robust tool for managing pharmaceutical data with precision and user-centric design.

1.1 Purpose

This document outlines the additions made since the second prototype testing. It highlights the specifications and tests done on the front-end, user interface, which holds a significant part of the overall PAPO Procurement Software, as well as the interaction of the UI with the database for storing user information and API implementation. All of these features will be used by all stakeholders, including developers and testers.

1.2 Scope

1.1.1 In Scope (features that will be tested in this prototype)

- **Online Database**
    - Validate the integrity of the database that stores user credentials by ensuring that it is up and running, bound to the API, and updates user information according to the changes in real time.
- **API Calls**
    - Certify that API calls create user, delete user, edit certain granted user information, log users in, log users out, and lock users after a certain number of incorrect login attempts.
- **Cost-Effective Algorithm**

    - Ensure that the addPriceAverage function calculates the weighted average of the prices of the replacements based on three price segments and outputs them in a sorted order.
- **Login Page**
    - Certify that the login page accepts user email and password, cross checks them with the credentials stored in the database, and grants access to the user by taking them to the main page.
- **Main Page**
    - Verify that the main page takes in the 8-digit code of a drug and displays the top five replacement drugs in a descending order of replacement suitability.
    - Certify that the checkboxes for each of the three price segments pass in the percentages as weights in the cost-effective algorithm and return the replacements accordingly.
    - Test the checkbox to allow multiple pills and ensure it outputs replacements whose package size is greater than that of the original drug in accordance with the package size algorithm.
    - Test the 'load all' button and verify it lists all the replacements beyond the top five items when pressed.
- **Logging**
    - Ensure that the email, password, and cookie value of users are logged in a file every time they login and logout.
- **Output Accuracy and Format**
    - Ensure that the final output is accurate, properly formatted, and matches the expected criteria based on the input data.

1.3) Roles and Responsibilities

Manuel	Algorithm Developer/Head of Continuous Improvement
Taha	Front-End Developer/QA Tester
Bora	Database Developer/QA Tester
Joel	Team Leader/Product Manager
Zaiyan	Technical Project Manager

2. Code Overview

Cost-Effective Algorithm

Functionality	Description
Setting Weights	Sets the weights (as percentages) for calculating the average price of drug replacements. w1, w2, and w3 represent the importance of the AWP Price, Acquisition Price, and WAC Price, respectively.
Calculate Average Price	Calculates the weighted average price of drug replacements using the formula (w1 * AWP Price + w2 * Acquisition Price + w3 * WAC Price) / (w1 + w2 + w3). This is directly applied to the replacements data structure to add a new column Average Price.
Sort Replacements	Sorts the replacements data structure in ascending order based on the Average Price. This is achieved using the sort_values(by='Average Price') method.
Export to CSV	Exports the sorted replacements data structure to a CSV file named replacements.csv, ensuring no index is included in the output file.
Add Price Average and Sort	This function encapsulates the process of calculating the weighted average price and sorting the replacements. It accepts weights w1, w2, w3, and the replacements data structure as arguments, calculates the average price, sorts the replacements by this average price, and returns the sorted replacements. This function is defined twice with identical functionality and parameters.

Front-End Implementation

Login Page

Feature	Description
useState for formData	Initializes state for form data to manage username and password inputs.
useNavigate Hook	Provides navigation functionality to redirect users after successful login.
handleSubmit Function	Handles form submission, including API call for user login, displaying success or error messages, setting cookies, and navigation.
Toast Notifications	Utilizes react-toastify for displaying success or error messages upon login attempts.
Cookies Management	Sets a session cookie with an expiration date upon successful login using js-cookie.
API Interaction	Calls login_user function from API services to authenticate user credentials.
Form Input Handling	Dynamically updates form data state upon input changes, ensuring real-time synchronization of user inputs.
Conditional Navigation	Redirects users to the home page upon successful login, enhancing user experience and flow.
Remember Me Checkbox	Includes a "Remember me" option for users, potentially for future implementation of session persistence.

Password and Username Inputs	Provides input fields for username and password with validation, ensuring user input is captured securely.
Link to Registration Page	Offers navigation to the registration page for new users who do not have an account yet.
Styling and Accessibility	Implements a responsive and accessible design for the login form, improving user interaction and inclusivity.

Main Page

Feature	Description
useNavigate Hook	Utilizes useNavigate for navigation control, redirecting users to the login page if they are not authenticated.
Authentication Check	Checks for the presence of a token cookie to verify user authentication; redirects unauthenticated users to the login page.
RecentOrders Component	Incorporates the RecentOrders component to display recent orders, illustrating modular design and component reuse.
No Data Constants	Uses NO_DATA constant to handle scenarios where there is no data available to display in components.
Dashboard Layout	Creates a dashboard layout with a flexible structure using flex display to organize components like RecentOrders.
DashboardStatsGrid Component	Presents a grid of statistics including total sales, expenses, customers, and orders,

	enhancing the dashboard's informative capacity.
Icons for Visualization	Employs icons from Io5 library (IoBagHandle, IoPieChart, IoPeople, IoCart) for intuitive visual representation of different statistics.
BoxWrapper Component	Utilizes a BoxWrapper function to standardize the appearance of statistic boxes, promoting a consistent and clean UI design.
Statistical Data Display	Showcases key business metrics (sales, expenses, customers, orders) in an easy-to-read format, underlining the dashboard's role as a decision-making tool.
Styling and Theming	Implements a cohesive styling and color theme across the dashboard, using tailwind classes for aesthetics and alignment.

Main Page

2.1 Visualization

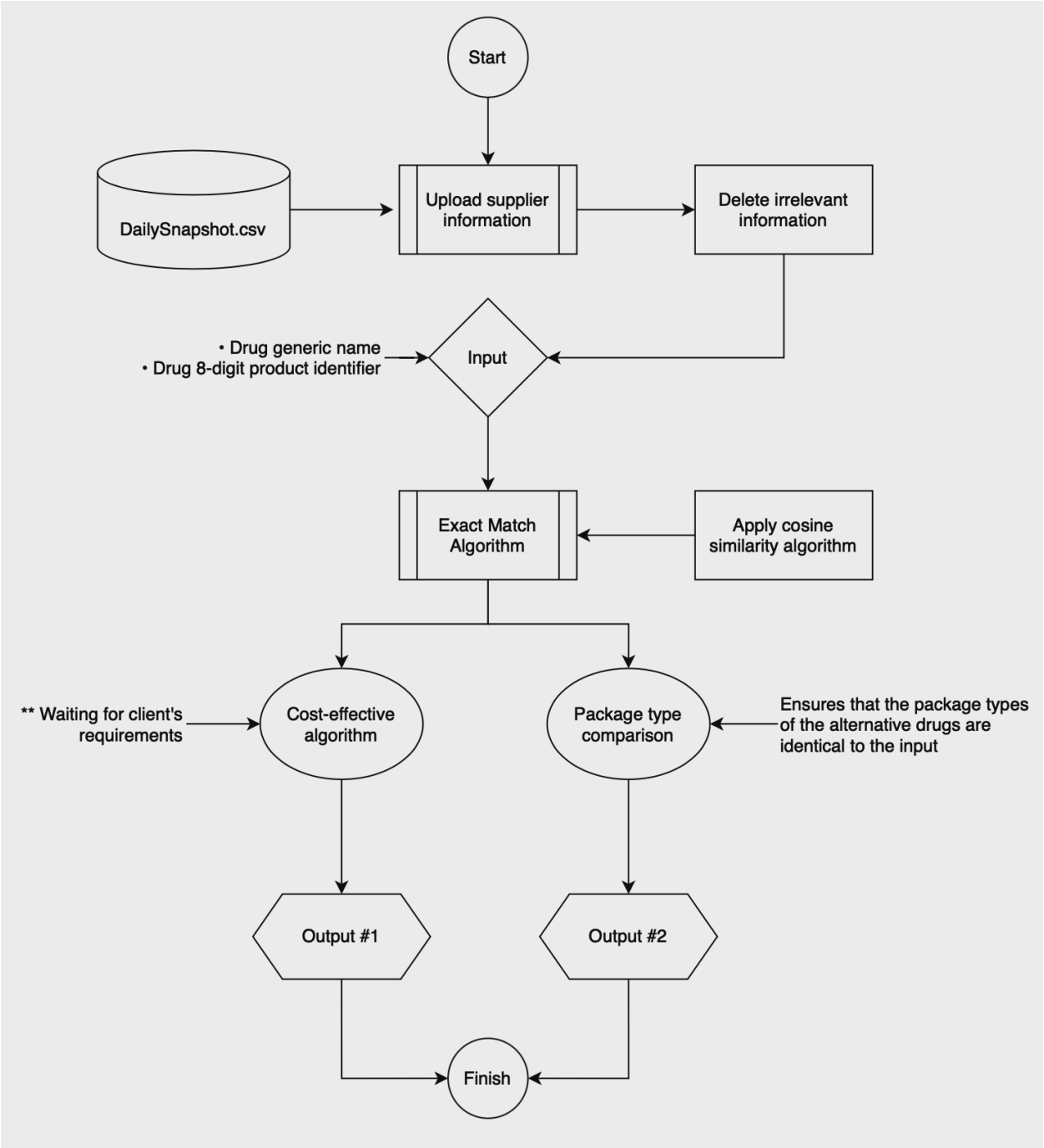

Figure A. Prototype Flowchart

3. Test Methodology

3.1 Pre-Testing Setup Procedure

1. Database Preparation:
    a. Login to AWS with admin credentials. Find the PostgreSQL database created to store user information. Ensure it is up and running, security protocols are set to accept incoming connections, and it is made 'publicly available.
2. Environment Setup:
    a. Ensure that the development and testing environments are properly configured, including any necessary software, databases, and network configurations.
    b. Verify that all dependencies and libraries used in the project are installed and up-to-date.
3. Data and Script Preparation:
    a. Ensure the client's data file, 'Daily Snapshot.csv', is stored in the API. This file contains the data set required for the testing.
4. Environment Verification:
    a. Ensure API is up and running and that it responds in a timely manner.
5. Browser Setup:
    a. Launch a reliable browser and navigate to the project website: https://papo-pearl.vercel.app/home

By following these enhanced setup procedures, the testing environment will be optimally prepared, ensuring a smooth and efficient testing process.

3.2 Testing Procedure

1) After navigating to the project website, observe the dynamic text on the left side that shows the capability of the software and the login box on the right side.
2) Under ‘Your email,’ enter the email of a registered user, and under ‘Password,’ enter the password of that user. Press the ‘Sign in’ button right below.
3) In the GitHub repository, find the update log file with the email, password, and cookie value of the user who just logged in.
4) After being taken to the main page, find the ‘Search drug’ box on the upper left corner. Enter the 8-digit identifier of the drug to find alternatives to.
5) Using the slides for each price segment, change the percent of the amount that could be ordered from that vendor. Make sure the sum of the percentages is equal to 100%.
6) Optionally, press the checkbox to allow multiple pills to acquire replacements whose package size is greater than that of the original drug in accordance with the package size algorithm.
7) Press enter to list the replacements. Observe the top five replacements based on match accuracy.
8) Press ‘load all’ to observe all the replacements identified by the algorithm.

Measurable Criteria

Criteria ID	Criteria Description	Target Value	Measurement Method
MC1	Accuracy of Cost Analysis Algorithm	≥ 98% accuracy	Compare algorithm output with known data
MC2	Response Time of Cost Analysis Calculations	≤ 2 seconds	Measure time from request to response
MC3	Login Page Authentication	100% accuracy	Manual and automated login attempts with valid/invalid credentials
MC4	Main Page Functionality	100% compliance	Manual verification of drug code input and replacement display
MC5	Price Segment Weight Application	100% accuracy	Automated tests with varied weights for price segments
MC6	Logging of User Credentials and Activities	100% accuracy	Audit logs for login/logout actions and data consistency
MC7	Output Accuracy of Replacement Drugs	100% accuracy	Manual and automated verification against expected outputs

Score sheet

The following active ingredient-drug code combinations will be tested:

Description	Drug Code
entecavir ORAL TABLET 0.5 MG	10000077
entecavir ORAL TABLET 1 MG	10000082
METHADONE HCL 10 MG/5 ML SOL 5	10083467
METHADONE HCL 5 MG/5 ML SOL 50	10083468

SAXAGLIPTIN 2.5 MG TAB 30	10281732
SAXAGLIPTIN 5 MG TAB 30	10281664
FLUPHENZINE HCL 2.5 MG TAB 100	10262230
FLUPHENZINE HCL 5 MG TAB 100 (	10262212
ISOSORBIDE DINITRATE 5 MG TAB	10208756
ISOSORBIDE DINITRATE 20 MG TAB	10208855

Score sheet	**Pass/Fail**	**Comments**
1. **Accuracy of Cost Analysis Algorithm:** Compare algorithm output with known data	☐	
2. **Response Time of Cost Analysis Calculations:** Measure time from request to response	☐	
3. **Login Page Authentication:** Manual login attempts with valid/invalid credentials	☐	
4. Main Page Functionality: Manual verification of drug code input and replacement display	☐	
5. **Price Segment Weight Application:** Automated tests with varied weights for price segments	☐	
6. Logging of User Credentials and Activities: Audit logs for login/logout actions and data consistency	☐	
7. Output Accuracy of Replacement Drugs: Manual and automated verification against expected outputs	☐	

4. Conclusion

Our prototype encapsulates the seamless integration of a reliable online database, robust API functionality, and advanced algorithms to elevate user experience and efficiency. The database secures user credentials and updates information in real time, ensuring data integrity. Our APIs facilitate essential user operations, including registration, authentication, and security measures against unauthorized access. At the core, our cost-effective algorithm meticulously calculates and sorts drug replacements by leveraging weighted averages of price segments, providing users with financially optimal options. The login page rigorously validates credentials, granting access to a main page where users can intuitively search for and customize drug replacements based on their preferences. Logging user activities enhances security, while the system's design ensures the accuracy and user-friendliness of outputs, affirming our prototype as a leading solution in pharmaceutical data management.

4.1 Future milestones

4.1.1 UI Tweaks

- Objective: Enhance user experience by simplifying the main page to display only essential information.
- Action Plan: Conduct user research to identify key data points of interest and redesign the main page UI to focus on these elements.

4.1.2 Customer Installation

- Objective: Ensure the product is easily installable by clients on their systems.
- Action Plan: Develop a comprehensive installation guide and an installer package that automates the setup process as much as possible.

4.1.3 Obtain Real Price Data from the Client

- Objective: Accurately calculate and display drug replacement options using real-world pricing.
- Action Plan: Work with the client to establish a secure, automated process for regularly updating drug price data in the system.

4.1.4 Settings Page Tweaks

- Objective: Streamline user management and data import processes within the application.
- Action Plan: Implement a settings section for admin users, featuring user creation tools and CSV upload capabilities for easy data management.

Appendix

- GitHub repository: https://github.com/BU-EC463