Software Implementation and Testing Document

For

Group <20>

Version 1.0

Authors:
Chelsea Mensah
Brady Henderson
Sebastian Metellus
Ryan Fontaine
Nevin Ramjitsingh

1. Programming Languages (5 points)

HTML, CSS, JavaScript.

2. Platforms, APIs, Databases, and other technologies used (5 points)

FullCalendar API (Sebastian Metellus)

- A JavaScript library that provides a customizable calendar with various renders allowing for user interaction and event planning, such as specifying various calendar renders. This was used in the Dashboard component with the 'Dashboard.js' file being used to import the scripts and render from the scripts in the 'index.html'. The 'Dashboard.js' also entails 'FullCalendar' and 'dayGrid' plugins to load the calendar into the dashboard.

Figma

- Figma is a collaborative web application for interface design. We have used it in our project to design the landing page, nav bar, and login page. We rendered the designs in HTML and CSS. We put the HTML code in React components and the CSS code in either the index.css file or separate CSS files specified for the specific component.

React

- React is a free and open-source front-end JavaScript library for building user interfaces based on components. React is how we have the components for our webpage and it is also how we have been able to set up the routing of the web app.

Node.js

- Node.js is a cross-platform, open-source server environment. Node.js is a back-end JavaScript runtime environment, that runs on the V8 JavaScript engine and executes JavaScript code outside a web browser. We are using it to test and make sure our website is building up to our standards.

Axios

- A JavaScript library that allows us to send HTTP requests from our code. This is used in the components 'Login.js' and 'RegistrationSection.js' to send user input data through POST requests to the server and route to corresponding authentication scripts hosted on the server side where the user data is compared with or added to our SQLite database.

SQLite

- A database engine is used to store a client's username and passwords into a 'users' table. The database is used by the AuthController.js and RegistrationController.js files to compare POST Request forms with the stored database information and add new data to the database respectively. The database is stored in the /pawplan/src/ folder as UsersDB.db

bcrypt

- A hashing function based on the blowfish cipher used to protect sensitive data. Here it is used to encrypt users' passwords in the RegistrationController.js file and compare input data with the hashed passwords stored by the database in the AuthController.js file. Hashed passwords are encrypted with a 'salt' to protect against rainbow table attacks and run with multiple salt rounds to help deter brute-force password attacks in case user data is stolen or leaked from the SQLite database.

FireBase

- Firebase is a Backend-as-a-Service (BaaS) platform developed by Google. It provides a variety of tools and services for mobile and web developers, including a real-time database, user authentication, cloud storage, and hosting services. Firebase's main goal is to help developers build, improve, and grow their apps. We have used Firebase as a replacement for SQLite to allow users to register and log in to our hosted website.

Netlify

- Netlify allows you to manage content without a server via Netlify CMS. It doesn't require a database, works directly with files in Git, and it's open source. We integrated netlify with

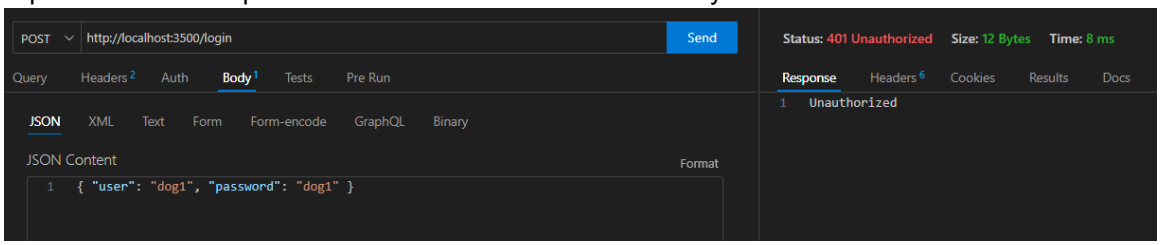
our firebase branch in github and we have allowed it to automatically build and deploy our site for us with every commit to the branch.

JWT

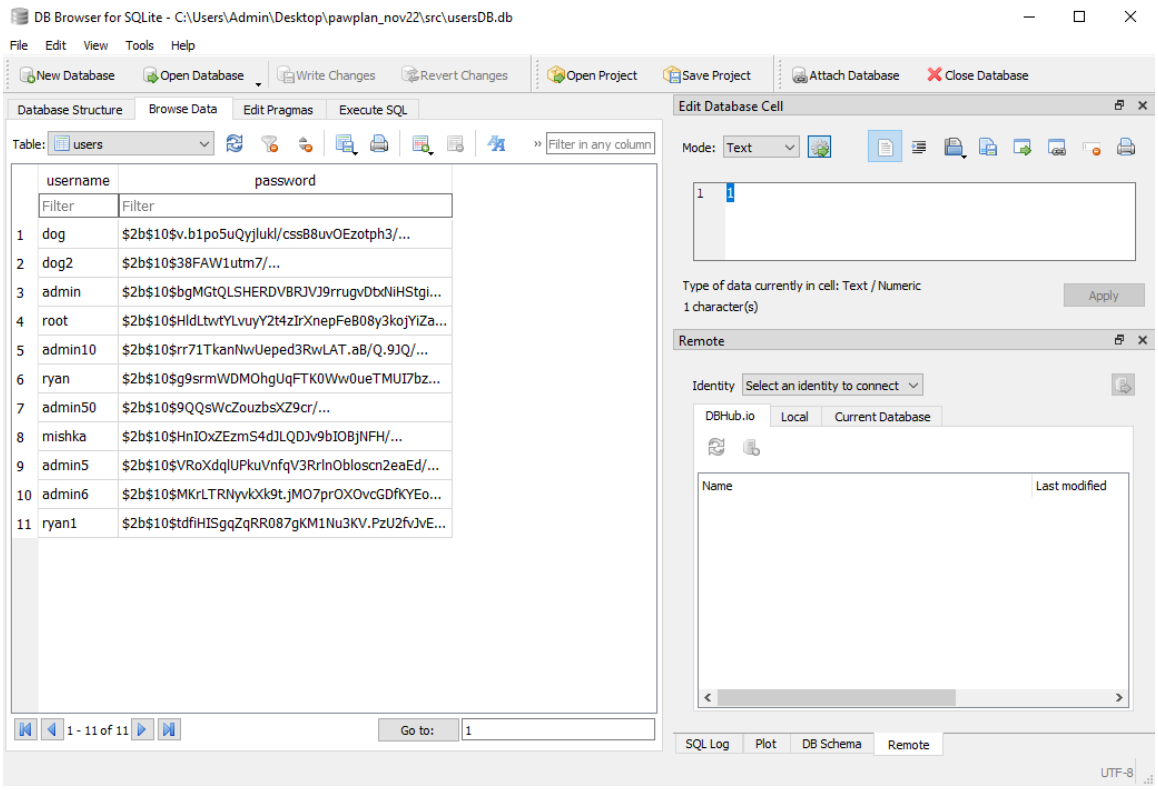
- JSON Web Tokens (JWTs) were originally used to create authenticated user sessions. This can be used to create an encrypted data point that can store relevant user data. It was used to store the users' username, password, and email in the browser's local storage before the project was migrated to Firebase. The authController signed a JWT with a passcode and then it was stored in the browser local storage if the user successfully logged in. Other pages such as the profile page could then access relevant data by decrypting the token, and also check to see if the user was authenticated by checking if the token was present in local storage. Signing out deleted the JWT from the browser.

3. Execution-based Functional Testing (10 points)

1. The buttons on the Nav Bar must route the user to the corresponding page.
    a. We have tested this by using the navbar and going from the home page to the About page and the login page by clicking on their respective buttons on the navbar.
2. The user must be able to use the login prompt to authenticate themselves on the Login page.
    a. A user can input a username and password on the /login page, when clicking the submit button, an HTTP POST request is sent to the server via the Axios API and routed to the AuthController.js script where the form's data is handled, and compared to the UsersDB SQLite database for authentication.
    b. Axios POST requests were debugged with the Thunderclient Plugin in VS Code to provide server response status codes and test functionality.



    c. **Firebase:** Firebase handles sign-in. Check for a matching email and password, and if it is matching then the user is signed in. Shows errors if the email or password doesn't match and renders it to the login screen. This was tested by inputting the correct email and password, as well as incorrect emails and passwords.
3. The user must be able to create a new account via the registration prompt on the Registration page.
    a. A user can input a username and password in the /registration page, when clicking the submit button, an HTTP POST request is sent to the server via the Axios API and routed to the RegistrationController.js script where the form's data is handled, and added to the UsersDB SQLite database with a hashed password.
    b. Axios POST requests were debugged with the Thunderclient Plugin in VS Code to provide server response status codes and test functionality via sending an HTTP request to /register on the server routing to the RegistrationController.js script.
    c. The UsersDB SQLite database was debugged via the DB Browser application to test and see if the RegistrationController was inserting users into the database correctly.



   d. **Firebase:** Firebase completely handles the user authentication, we tested the ability to create an account with multiple different emails. Firebase checks for emails that already exist in the system and if the passwords aren't "strong enough".
4. The user must be able to use the calendar on the dashboard to put in the scheduled activities for their respective dogs.
   a. We have tested this with the add and delete buttons on the dashboard.
   b. We have tested dog-specific events by changing the dogs we assign events to in our app and see how the calendar event is affected by the changes

4. Execution-based Non-Functional Testing (10 points)

*Describe how/if you performed non-functional testing for your project (i.e., tested for the **non-functional requirements** listed in your RD).*

1. Pages should render consistently across different screen sizes.\
   a. We have been working on testing the webpage on different screen sizes and tweaking the designs for what we need.
2. The database should not be able to be deleted by users.
   a. The user has no access to the database from inside the web application.
3. Users should not be able to view other users' information.
   a. Each user can only view the application from their perspective and can't see other users as there are no user interactions on the website
4. Users should only be able to see their events on the calendar
   a. We tested multiple accounts and confirmed that a user can also see the events that are currently stored in their unique event array attached to their user document in the user collection in our Firestore./
   .

5. Non-Execution-based Testing (10 points)

Our code was peer-reviewed as it was developed. We also had friends test our application after we hosted it on Netlify.