Ricard oosthuizen

Final Term 4 Task (Pen Pets)

Functional Requirements Checklist:

Completed Features:

- ☑ ~~**User Authentication**: Fully functional, secure account creation and login.~~
- ☑ ~~**Pet Upload and Display**: Users can upload pet details with images, which are stored in MongoDB and shown on the home page.~~
- ☑ ~~**Browsing and Filtering**: The browse page lists pets with options to filter by name (A-Z) and age.~~
- ☑ ~~**Navigation and Logout**: Smooth transitions between pages with a navbar that highlights the active page. Secure logout returns users to the login page.~~
- ☑ ~~**Database Integration**: Efficient MongoDB storage and retrieval for user and pet data.~~
- ☑ ~~**Admin Management**: Complete CRUD operations for the admin to manage pet data.~~
- ☑ ~~**Security and Verification**: Enhanced measures to ensure secure handling of user data.~~

- ☑ ~~**Clear Navigation**: Fully functional, intuitive navigation across pages, with a highlighted navbar for each active page.~~

- ☑ ~~**Deployment**: Progress is underway on deployment with AWS and Google Cloud, bringing the project closer to a live environment.~~

System Documentation:

(https://github.com/2Ricky3/PetAdoption.git)

1. **Technical Architecture:**
   - **Frontend:** React (JavaScript, CSS) for building a responsive UI.
   - **Backend:** Node.js and Express.js for handling API requests and MongoDB interactions.
   - **Database:** MongoDB Atlas for storing user and pet data. The connection is managed through Mongoose.

2. **Database Schema:**
   - **Users Collection:**
     - `username` (String)
     - `password` (String, hashed)
     - `email` (String, unique)
   - **Pets Collection:**
     - `petName` (String)
     - `age` (Number)
     - `imageURL` (String)
     - `description` (String)
     - `uploadedBy` (User Reference)

3. **API Endpoints:**
    - **POST** `/auth/signup` – Create a new user.
    - **POST** `/auth/login` – Authenticate an existing user.
    - **POST** `/pets/upload` – Upload pet details.
    - **GET** `/pets` – Fetch all available pets.
    - **GET** `/pets/filter` – Filter pets by name or age.

4. **UI Design:**
    - Built using React, the UI offers a user-friendly experience with clear navigation, interactive forms for pet uploads, and responsive design for desktop and mobile users.
    - CSS styling ensures consistency across pages with a simple and clean aesthetic.
    -

5. **Deployment Instructions:**

1. **Setting Up MongoDB Atlas:**
    - Ensure your MongoDB Atlas account is properly set up and connected to the application. Use the connection string provided to link your backend to the MongoDB database.
    - MongoDB will handle the data for users and pets.

2. **Frontend (React):**

    Navigate to the client directory in your project and install all dependencies by running:
    ```
    npm install
    ```

    To start the React development server, use:
    sql
    ```
    npm start
    ```

    - This will start the frontend application, which you can access locally.

3. **Backend (Node.js and Express):**

    Navigate to the server directory and install all necessary packages with:
    ```
    npm install
    ```

Start the server by running:

```
node server.js
```

- This will start the Node.js server to handle API requests.

4. **Google Cloud Deployment:**
   - **Step 1: Set Up a Google Cloud Project:**
     - Log in to your Google Cloud Console and create a new project.
     - Enable **Google Cloud App Engine** for deploying the application.
     - Ensure that **Google Cloud Storage** is enabled to handle media assets like pet images.
   - **Step 2: Set Up Google Cloud Environment:**
     - Use **Google Cloud Shell** to connect to your project.

   Navigate to the root directory of your application in the shell and initialise Google App Engine using:

   ```
   gcloud app create
   ```

     - This will set up the required environment for deploying the app.
   - **Step 3: Deployment:**
     - To deploy your application, first make sure that both the frontend and backend are set up in a single repository or properly connected.

   Deploy the app using:

   ```
   gcloud app deploy
   ```

     - Google Cloud will deploy your MERN stack application, and you will receive a URL where your live application is hosted.

5. **Post-Deployment Steps:**
   - After deployment, verify that the application is live and connected to the MongoDB database.
   - Set up any environment variables for security purposes (e.g., API keys, database credentials) using the Google Cloud Console.
   - Ensure you configure Google Cloud to use HTTPS for secure communication.
   - Monitor the app performance using Google Cloud's monitoring tools

Problem Statement:

The growing number of stray animals and overcrowded shelters present a significant societal challenge, leading to an overwhelming demand for an efficient and user-friendly system that facilitates the adoption process. Traditional adoption methods often lack accessibility, transparency, and ease of use, which can deter potential adopters from finding and adopting pets in need. In response to this issue, this MERN stack application is designed to simplify the pet adoption process by providing an accessible platform where users can easily browse available pets, filter by specific characteristics like age or breed, and adopt them from the comfort of their homes.

Furthermore, the platform offers an intuitive pet upload system for pet owners and adoption centres, allowing them to quickly publish pet details, including images, to increase visibility and adoption chances. The seamless login and signup functionality, integrated with MongoDB, ensures secure user authentication, while the CRUD system for managing pets makes it simple to create, read, update, and delete pet entries. By streamlining the entire process, this application reduces the time and effort required for both adopters and organisations, ultimately encouraging more successful pet adoptions and reducing the strain on animal shelters.