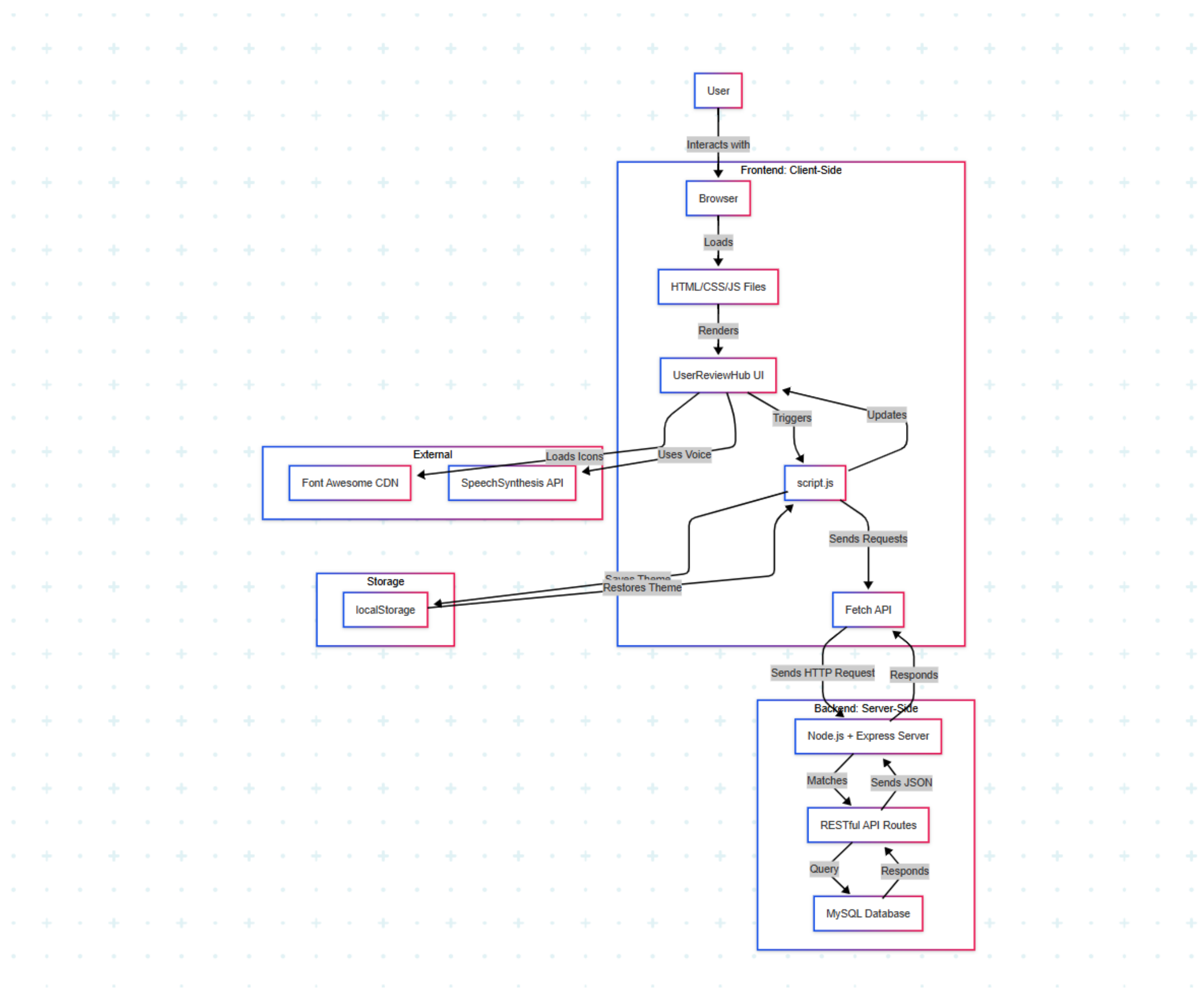
UserReviewHub Documentation

A Comprehensive Review Management Application

ARCHITECTURE DIAGRAM:



Contents

1 Project Overview

UserReviewHub is a web application designed to manage user reviews for various products. It provides an interactive user profile page where users can view, sort, filter, and manage their reviews. The application includes modern UI/UX features such as animations, theme switching, and accessibility options like text-to-speech for reviews. This project was developed iteratively, incorporating user feedback to enhance functionality and user experience.

1.1 Key Details

- **Project Name**: UserReviewHub
- **Development Period**: Iteratively developed as of June 16, 2025
- **Author**: Grok 3, created by xAI
- **Target Audience**: Users who wish to manage product reviews interactively

1.2 Languages and Technologies Used

- **Frontend**:
  - HTML: Structure of the web application
  - CSS (with Tailwind CSS): Styling, animations, and responsive design
  - JavaScript: Core logic, DOM manipulation, and API interactions
- **Backend** (assumed, as per API endpoints):
  - Node.js (assumed): For handling API requests
  - RESTful API: Endpoints for authentication, product fetching, and review management
- **Dependencies**:
  - Tailwind CSS (v3.4.1): For utility-first CSS styling

  - PostCSS (v8.4.31): For processing Tailwind CSS
  - Autoprefixer (v10.4.15): For CSS vendor prefixing
  - Font Awesome: For star icons in ratings

2 Setup Instructions

This section outlines the steps to set up the UserReviewHub application on your local machine.

2.1 Prerequisites

- Node.js (v16 or higher) and npm installed
- A code editor (e.g., VS Code)
- Git (optional, for cloning the project)
- A modern web browser (e.g., Chrome, Firefox)

2.2 Step-by-Step Setup

1. **Clone or Download the Project**
   - If using Git, clone the repository:
     `git clone <repository-url>`
   - Alternatively, download the project files as a ZIP and extract them.
2. **Install Dependencies**
   - Navigate to the project directory: `cd UserReviewHub`
   - Install the required dependencies: `npm install tailwindcss@3.4.1 postcss@8.4.31 autoprefixer@10.4.15`
3. **Configure Tailwind CSS**
   - Create a `tailwind.config.js` file: `npx tailwindcss init`
   - Update `tailwind.config.js` to include project files:

```
/**   @type   {import('tailwindcss').Config}   */
module.exports   =   {   content:   ['./*.html',
'./src/**/*.{html,js}'], theme: { extend: {},
    },
    plugins: [],
};


```

   - Create a `postcss.config.js` file:

```
module.exports = {
    plugins: {
    tailwindcss: {},
    autoprefixer: {},
    },
};
```

4. **Set Up the Backend (Assumed)**
   - The application interacts with a backend API at `http://localhost:3000/api`.
   - Ensure the backend server is running, providing endpoints like:
     - `/api/auth/me`: Fetch user details
     - `/api/auth/signin`, `/api/auth/signup`, `/api/auth/signout`: Authentication
     - `/api/products`: Fetch products and reviews
     - `/api/products/{id}/review`: Manage reviews (POST, PUT, DELETE)
   - If the backend is not set up, you may need to create a Node.js server or mock these endpoints.

5. **Prepare the Frontend Files**

   • Ensure the project has an `index.html` file with the following structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,␣initialscale=1.0">
    <title>UserReviewHub</title>
    <link
       href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/5.15.4/css/all
       .css" rel="stylesheet">
    <link href="dist/styles.css" rel="stylesheet">
</head>
<body>
    <header>
        <h1>UserReviewHub</h1>
        <div id="auth-section"></div>
    </header>
    <div id="search-section">
        <input type="text" id="search-bar" placeholder="Search␣ products..."
           onkeyup="searchProducts()">
    </div>
    <div id="products"></div>
    <div id="user-profile" class="hidden">
        <h2>User Profile <span id="close-profile"
           class="cursorpointer">[Close]</span></h2>
        <div id="profile-details"></div>
        <div id="user-reviews"></div>
    </div>
    <div id="auth-modal" class="hidden">
        <span id="close-modal" class="close">&times;</span>
        <h2 id="modal-title"></h2>
        <form id="auth-form">
            <div id="username-field">
                <label for="username">Username:</label>
                <input type="text" id="username" required>
            </div>
            <label for="email">Email:</label>
            <input type="email" id="email" required>
            <label for="password">Password:</label>
            <input type="password" id="password" required>
            <button type="submit" id="auth-submit"></button> </form>
    </div>
    <script src="src/script.js"></script>
</body>
</html>
```

- Place `script.js` and `styles.css` in the appropriate directories (`src/` and `src/`, respectively).

3 Running the Application

Follow these steps to run UserReviewHub on your local machine.

3.1 Steps to Run

1. **Compile the CSS**
   - Run the following command to process `styles.css` with PostCSS: `npm run build:css`
   - If not already set, add this script to `package.json`:

```
"scripts": {
    "build:css": "postcss src/styles.css -o dist/styles.css"
}
```

2. **Start the Backend Server**
   - Ensure the backend server is running on `http://localhost:3000`.
   - If using a Node.js backend, start it with: `node server.js` (or the appropriate command for your backend setup).
3. **Serve the Frontend**
   - Use a local server to serve the frontend files. For example, using `live-server`: `npx live-server`
   - Alternatively, use any static file server or open `index.html` directly in a browser (note: some features may require a server due to CORS).
4. **Access the Application**
   - Open your browser and navigate to `http://localhost:8080` (or the port provided by your server).
   - You should see the UserReviewHub homepage with a list of products.

4 Testing the Application

This section provides instructions to test the core functionalities of UserReviewHub.

4.1 Test Scenarios

1. **Authentication**
   - *Test Case 1: Sign Up*
     - Click ”Sign Up”, enter a username, email (e.g., `hlokeshwari14@gmail.com`), and password.
     - Submit the form and verify a success message. • *Test Case 2: Sign In*
     - Click ”Sign In”, enter the email and password.
     - Verify the welcome message (e.g., ”Welcome, Hlokeshwari”) and profile button appear.
   - *Test Case 3: Sign Out*
     - After signing in, click ”Sign Out”.
     - Verify the sign-in/signup buttons reappear.

2. **User Profile and Reviews**
   - *Test Case 4: View Profile*
     - Sign in as ”Hlokeshwari” and click ”View Profile”.
     - Verify the profile displays with username, email, and reviews.
   - *Test Case 5: Sort Reviews*
     - In the profile, use the ”Sort by” dropdown to sort reviews by ”Highest Rating”. – Verify Product 2 (4 stars) appears first.
   - *Test Case 6: Filter Reviews*
     - Use the ”Filter” dropdown to select ”4 Stars”. – Verify only Product 2 is shown.
   - *Test Case 7: Edit/Delete Reviews*
     - Click ”Edit” on a review, update the rating and text, and submit.
     - Click ”Delete” on a review and confirm.
     - Verify the changes are reflected after page reload.
3. **Advanced Features**
   - *Test Case 8: Sentiment Indicators*
     - In the profile, verify reviews show sentiment indicators (e.g., ” Positive” for 4 stars).
   - *Test Case 9: Review Animations*
     - Open the profile and observe the fade-in-up animation for reviews.
   - *Test Case 10: Review Statistics*
     - Verify the ”Review Statistics” section shows total reviews, average rating, and highest-rated product.
   - *Test Case 11: Rating Distribution Chart*
     - Check the chart in ”Review Statistics” for correct distribution (e.g., 1: 1, 2: 1, 4: 1).
   - *Test Case 12: Highlight Positive Reviews*
     - Click ”Highlight Positive” and verify Product 2 gets a glowing green border.
   - *Test Case 13: Theme Switching*
     - Click the theme toggle (/) and verify the UI switches between light and dark modes.
   - *Test Case 14: Voice Feedback*
     - Click ”Listen” on a review and verify the review text is read aloud.

4.2 Debugging Tips

- Open the browser’s Developer Tools (F12) to check Console logs for errors.
- Inspect the Network tab to verify API requests to `http://localhost:3000/api`.
- If Tailwind CSS styles fail, ensure `dist/styles.css` is generated and linked correctly.

5 Functionalities

UserReviewHub includes the following features, implemented iteratively:

1. **User Authentication**
    - Sign up, sign in, and sign out via API endpoints.
    - Displays a welcome message and profile button upon login.
2. **User Profile Display**
    - Shows username, email, and user reviews.
    - Fetches reviews by matching `user.id` with `review.user_id`.
3. **Review Sorting**
    - Sort reviews by newest, oldest, highest rating, or lowest rating.
4. **Review Editing/Deleting**
    - Edit or delete reviews with API calls.
5. **Formatted Timestamps**
    - Displays review timestamps in a readable format (e.g., ”June 16, 2025, 11:07 PM”).
6. **Sentiment Indicators**
    - Shows an emoji and label for each review (e.g., ” Positive” for 4-5 stars).
7. **Animated Review Cards**
    - Reviews fade in with a bounce effect when the profile loads.
8. **Review Statistics**
    - Displays total reviews, average rating, and highest-rated product.
9. **Review Filtering**
    - Filter reviews by rating (1-5 stars or all).
10. **Rating Distribution Chart**
    - A canvas-based bar chart showing the distribution of ratings.
11. **Highlight Positive Reviews**
    - Toggle to highlight reviews with 4-5 stars with a glowing effect.
12. **Dynamic Theme Switching**
    - Toggle between light and dark modes, with preferences saved in `localStorage`.
13. **Voice Feedback**
    - ”Listen” button to read review text aloud using text-to-speech.

6 Project Structure

The project is organized as follows:

- `index.html`: Main HTML file with the application structure.
- `src/script.js`: JavaScript file containing all logic.
- `src/styles.css`: CSS file with Tailwind CSS and custom styles.
- `dist/styles.css`: Compiled CSS output.
- `tailwind.config.js`: Tailwind CSS configuration.
- `postcss.config.js`: PostCSS configuration.
- `package.json`: Project metadata and scripts.

7 Known Issues and Limitations

- **Backend Dependency**: Theapplicationrequiresarunningbackendserverat`http://localhost:3000`. Without it, API calls will fail.
- **Browser Compatibility**: Voice feedback (text-to-speech) may not work in all browsers (e.g., older versions of Safari).
- **Linting Errors**: Tailwind CSS directives may cause linting errors in some editors. Configure the editor to ignore unknown at-rules or use PostCSS language mode.
- **Performance**: Loading many reviews may slow down the profile page; consider implementing pagination in future updates.

8 Future Enhancements

- Add profile avatar upload functionality.
- Implement review tags for better categorization.
- Add a shareable profile link feature.
- Introduce pagination for reviews to improve performance.
- Enhance accessibility with ARIA labels and keyboard navigation.

9 Conclusion

UserReviewHub provides a robust and interactive platform for managing product reviews. With features like sorting, filtering, animations, and accessibility options, it offers a modern user experience. This documentation serves as a guide for setting up, running, and testing the application, along with a detailed overview of its functionalities.