SOFTWARE REQIUREMENT SPECIFICATION

Table of Contents:

1.Abstract:

This HTML webpage provides a user-friendly interface for exploring GitHub user profiles. Utilizing modern web technologies including HTML, CSS, and JavaScript, the webpage allows users to input a GitHub username, retrieves user data from the GitHub API using Axios, and dynamically displays the user's information including their name, bio, followers, following, and repositories. The design is clean and intuitive, featuring responsive elements for optimal viewing on various devices. Error handling is implemented to gracefully handle cases such as user not found or unexpected API errors. The interface promotes ease of use, making it simple for users to navigate and discover GitHub profiles effortlessly. Whether for personal curiosity or professional research, this GitHub Profiles webpage offers an accessible gateway to explore the diverse GitHub community.

2.Functional Requirements:

2.1.User Input:

- The webpage shall provide an input field for users to enter a GitHub username.
- The input field shall allow users to type alphanumeric characters, symbols, and special characters typically found in GitHub usernames.
- Users shall be able to submit their input either by pressing the "Enter" key or clicking a submit button.

2.2.Fetching User Information:

- Upon submission of a GitHub username, the system shall make an HTTP request to the GitHub API to fetch the corresponding user's information.
- The system shall handle successful API responses by extracting relevant user data, including name, bio, followers count, following count, and public repositories count.
- The system shall display the fetched user information on the webpage in a visually appealing and informative manner.

2.3.Displaying User Information:

- The system shall present the fetched user information in a user-friendly format, using appropriate typography, layout, and styling.
- The user's profile picture shall be displayed alongside their name and other details.
- Each piece of user information (name, bio, followers count, following count, and public repositories count) shall be clearly labeled and formatted for easy comprehension.
- The user's name shall serve as a clickable link to their GitHub profile, opening in a new browser tab.

2.4.Error Handling:

- The system shall handle scenarios where the entered GitHub username does not correspond to an existing user by displaying an appropriate error message.

- In case of unexpected errors during the API request (e.g., network issues, server errors), the system shall display a generic error message indicating the failure to fetch user data.
- Error messages shall be prominently displayed on the webpage and formatted to ensure visibility and readability.

2.6.User Feedback:

- Upon successful retrieval and display of user information, the system shall provide visual feedback to the user, indicating that the operation was successful.
- Feedback mechanisms such as loading spinners, success banners, or animations shall be utilized to enhance user experience and convey system status.

2.7.Accessibility:

- The webpage shall be designed and implemented following web accessibility standards to ensure usability for individuals with disabilities.
- All interactive elements, including input fields, buttons, and links, shall be accessible via keyboard navigation and assistive technologies.
- Textual content, including user information and error messages, shall be legible and appropriately contrasted for users with visual impairments.

3.Non Functional Requirements:

3.1.Performance:

- Response Time: The webpage shall load within 3 seconds under normal network conditions.
- API Request Time: The system shall make API requests to fetch user data with a response time of less than 1 second on average.
- Scalability: The webpage shall be capable of handling concurrent user requests without significant degradation in performance, even during peak usage periods.

3.2.Security:

- Data Encryption: All communication between the webpage and the GitHub API shall be encrypted using HTTPS to ensure data confidentiality.
- Data Handling: User data fetched from the GitHub API shall be handled securely and stored temporarily in memory, without persisting sensitive information on the client-side or server-side.
- Authentication: The webpage shall not require authentication from users to access GitHub profiles, but it shall follow GitHub API authentication guidelines if rate limits become a concern.

3.3.Usability:

- User Interface Consistency: The webpage shall maintain consistency in design elements, layout, and navigation across different browsers and devices.
- Accessibility: The webpage shall comply with web accessibility standards (e.g., WCAG) to ensure usability for users with disabilities, including support for screen readers and keyboard navigation.
- Error Handling: Error messages shall be clear, concise, and displayed prominently on the webpage to assist users in understanding and resolving issues encountered during usage.

3.4.Reliability:

- Availability: The webpage shall be available for use 24/7, with minimal downtime for maintenance or updates.
- Error Recovery: The system shall gracefully handle errors encountered during API requests or data retrieval, providing informative error messages and allowing users to retry operations if necessary.
- Data Integrity: User information displayed on the webpage shall accurately reflect the data retrieved from the GitHub API, ensuring consistency and reliability in the displayed content.
- **3.5.Compatibility:**
- Browser Compatibility: The webpage shall be compatible with major web browsers, including Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.
- Device Compatibility: The webpage shall be responsive and render properly on various devices, including desktops, laptops, tablets, and smartphones, with different screen sizes and resolutions.

3.5.Documentation:

- User Documentation: A user manual or help section shall be provided to guide users on how to use the webpage, including instructions for entering GitHub usernames and interpreting displayed information.
- Developer Documentation: Technical documentation shall be available for developers, including API usage instructions, system architecture overview, and code documentation for maintainability and future enhancements.

4.Design:

4.1.High Level Design:

4.1.1.Frontend Architecture:

- HTML Structure: The webpage shall be structured using HTML5, with semantic markup for content elements such as forms, headings, and sections.

- Styling with CSS: CSS (Cascading Style Sheets) shall be used to define the visual presentation of the webpage, including layout, typography, colors, and responsiveness for different screen sizes.
- Interactivity with JavaScript: JavaScript shall be employed to add dynamic behavior to the webpage, such as handling user input, making API requests, and updating the DOM (Document Object Model) with fetched user data.

4.1.2.User Interface Components:

- Input Field: An input field shall be provided for users to enter a GitHub username.
- Display Area: A designated area on the webpage shall display the fetched user information, including profile picture, name, bio, followers count, following count, and public repositories count.
- Error Message Area: An area shall be reserved for displaying error messages in case of failed API requests or invalid user input.

4.1.3.API Integration:

- The frontend shall utilize the Axios library to make HTTP requests to the GitHub API for fetching user data based on the entered username.
- Error handling mechanisms shall be implemented to handle various scenarios, such as user not found or unexpected API errors, providing appropriate feedback to the user.

4.1.4.Responsive Design:

- The webpage shall be designed to be responsive, ensuring optimal viewing and usability across devices with different screen sizes and resolutions.
- Media queries and CSS flexbox/grid layouts shall be employed to adapt the layout and styling based on the device's viewport dimensions.

4.2.Low-Level Design:

4.2.1.HTML Structure:

- The HTML structure shall consist of semantic elements such as <header>, <main>, <section>, and <footer> to organize the content.
- Input fields, buttons, and error message containers shall be created using <input>, <button>, and elements with appropriate IDs and classes for styling and JavaScript interaction.

4.2.2.CSS Styling:

- CSS rules shall be defined to style various elements on the webpage, specifying properties such as font family, colors, margins, padding, and box shadows.
- Class and ID selectors shall be used to target specific elements and apply styling rules consistently across the webpage.

4.2.3.JavaScript Interactivity:

- Event listeners shall be added to the input field and form submission to trigger JavaScript functions for handling user input and API requests.
- JavaScript functions shall be written to make asynchronous HTTP requests to the GitHub API using Axios, parse the API response, and update the DOM with fetched user data or error messages.

4.2.4.Error Handling:

- Error handling functions shall be implemented to handle different types of errors, including network errors, server errors, and API response errors.
- Error messages shall be dynamically generated and inserted into the error message container on the webpage, providing feedback to the user about the encountered issue.

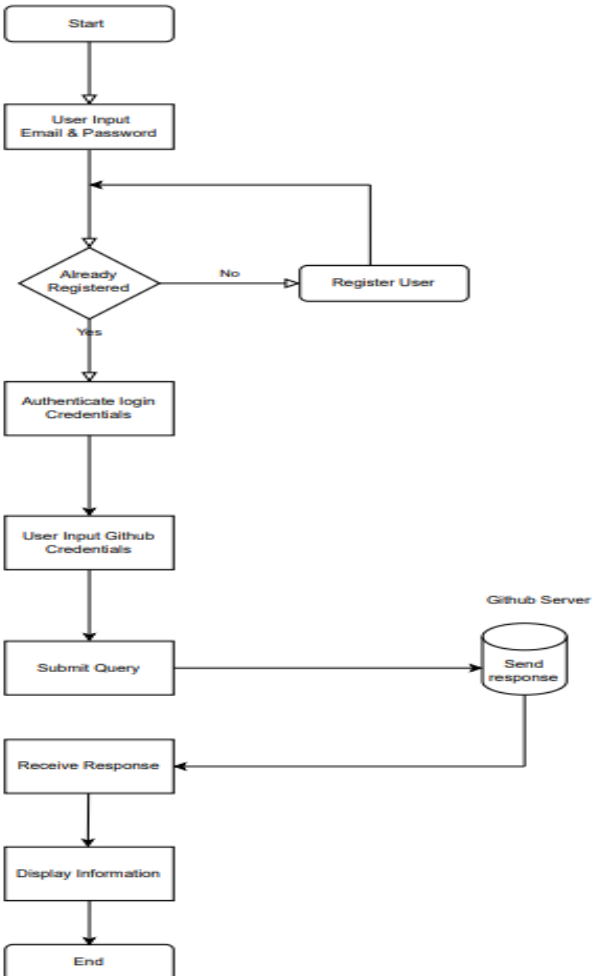
4.2.5.API Integration:

- The Axios library shall be imported into the JavaScript code using a <script> tag or module import statement.
- Axios configuration options, such as the API base URL and request headers, shall be set to interact with the GitHub API securely and efficiently.

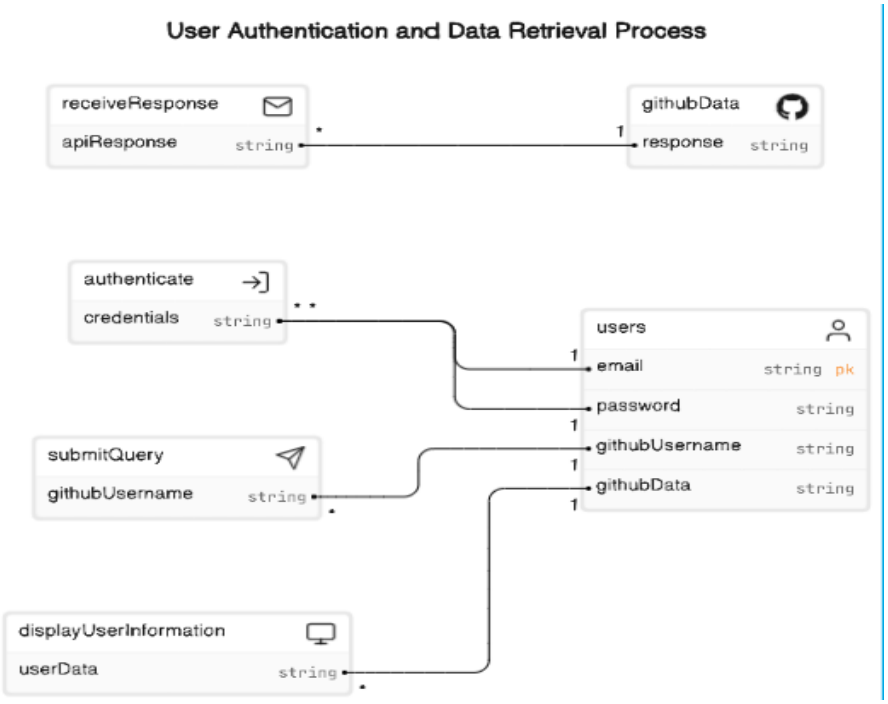
4.2.6.Responsive Design Implementation:

- Media queries shall be defined in the CSS stylesheet to adjust the layout, font sizes, and other styling properties based on the viewport width breakpoints.
- CSS flexbox and grid layouts shall be utilized to create responsive and flexible designs that adapt to different screen sizes and orientations

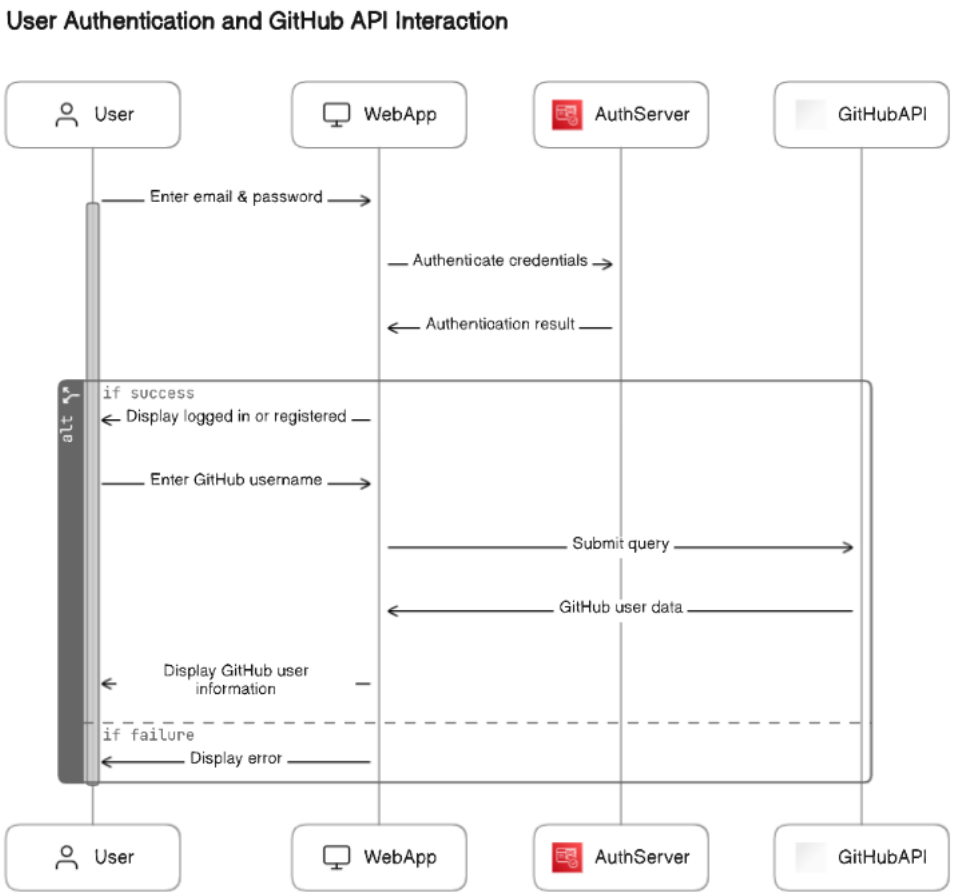
5.Flowchart:



6.ER Diagram:



7.Sequence Diagram:


User Authentication and GitHub API Interaction
User
WebApp
AuthServer
GitHubAPI
Enter email & password
Authenticate credentials
Authentication result
alt
if success
Display logged in or registered
Enter GitHub username
Submit query
GitHub user data
Display GitHub user information
if failure
Display error
User
WebApp
AuthServer
GitHubAPI

8.Test Case: