UI/UX Documentation

for Stargazers Viewer App

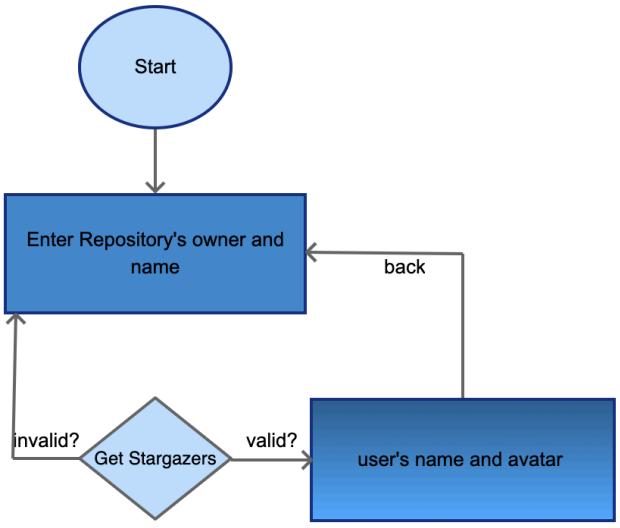
Overview

The Stargazers Viewer App is a simple mobile application built with Expo and React Native. It allows users to view the list of stargazers for a given GitHub repository. Users can input the repository owner's name and the repository name to fetch and display the list of users who have starred the repository, along with their avatars and usernames.

Expo was chosen for its testing process with Expo Go, which allows real-time testing on physical devices by simply scanning a QR code.

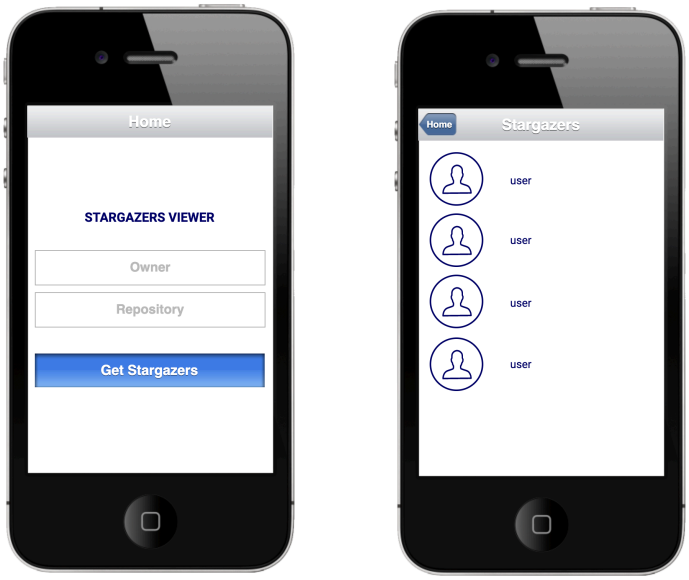
Potential Users include: Software Developers, General GitHub Users.

User Flow



1. **Start**: The user launches the Stargazers Viewer application.
2. **Enter Repository's Owner and Name**: The user inputs the repository owner's name and the repository name.
3. **Press "Get Stargazers" Button**:
    - **If Valid**: If the repository exists, the app displays a list of users who have starred the repository, showing their avatars and usernames.
    - **If Invalid**: If the repository does not exist or cannot be found, the user receives an informative message and can corrected the data.

Wireframe



1. **Home Screen**:
    - The user enters the GitHub repository owner and repository name in the respective input fields.
    - The user taps the "Get Stargazers" button to fetch the list of stargazers.
    - If the input is invalid or empty, the button is disabled.
2. **Stargazers Screen**:
    - The list of stargazers is displayed with their avatars and usernames.
    - The user can scroll through the list to view all stargazers.

Features

- **Input Fields**: Two text input fields for entering the GitHub repository owner and repository name.
- **Fetch Stargazers**: Fetch the list of stargazers from the GitHub API.
- **Display Stargazers**: Display the stargazers with their avatars and usernames.
- **Loading State**: Show a loading indicator while fetching data.
- **Error Handling**: Display an error message if the fetch operation fails.
- **Responsive Design**: Adjust layout and styles for different platforms.
- **Multilingual Support**: Included i18n for multilingual support.

Component Structure

App Component

- **UI Elements**:
    - **NavigationContainer**: Manages the navigation tree and the app's navigation state.
    - **Stack.Navigator**: Defines a stack-based navigation pattern, allowing for transitions between different screens.
    - **Stack.Screen**: Specifies individual screens within the stack navigator, including HomeScreen and StargazersScreen.

HomeScreen Component

- **State Management**
    - **owner**: Stores the repository owner's name.
    - **repo**: Stores the repository name.
    - **loading**: Indicates whether the app is currently fetching data.
    - **error**: Stores any error message from the fetch operation.

- **UI Elements**:
    - **View**: it wraps the components of this screen.
    - **InputText** component encapsulates the TextInput functionality in a reusable and manageable way. Utilized here for input of owner and repository name.
    - **ImageBackground**: for displaying background images.
    - **Pressable** button to trigger the fetch operation.
    - **ActivityIndicator** to show loading state.
    - **Text:** It is used for showing labels, error messages, and button text.
    - **KeyboardAvoidingView**: Adjusts the UI layout when the keyboard is shown.

- **Styles**: Custom styles for layout and appearance.

InputText Component

- **Props**:
    - **placeholder**: Placeholder text displayed when the input is empty.
    - **value**: Current value of the input field.
    - **onChangeText**: Callback function invoked when the text input changes.
    - **onSubmitEditing**: Callback function invoked when the submit action is triggered.
    - **returnKeyType**: Specifies the type of action triggered by the return key on the keyboard.

- **UI Element**:
    - **TextInput**: React Native component for text input, providing essential text input functionalities.

StargazersScreen Component

- **Props**:
  - **stargazers**: List of stargazers passed from the HomeScreen component.
- **UI Elements**:
  - **View**: it wraps the components of this screen.
  - **FlatList** to render the list of stargazers.
  - **Image** to display user avatars.
  - **Text** to display usernames or info message.
- **Styles**: Custom styles for layout and appearance. For example, StatusBar and Dimensions are used to adjust the layout.

API Integration

fetchStargazers Function:

- **Dependencies:**
  - **axios**: A popular library for making HTTP requests utilized here to send a GET request to the GitHub API.
- **Parameters:**
  - **owner**: The GitHub repository owner's username or organization name
  - **repo**: The name of the GitHub repository
  - **t**: Translation function provided by useTranslation for error message localization.
- **Description**
  - Sends a GET request to the GitHub API to retrieve the list of stargazers for the specified repository.
  - Handles loading, success, and error states.
  - Returns the data or an error message based on the response. If data is null return an info message.

Internationalization (i18n)

- The app supports multiple languages using the i18next library.
- useTranslation:A hook from the react-i18next library. It provides translation functions and the current language.
- Text elements and error messages are translated based on the user's language preference.

Design Decisions

- **Loading Indicator**: An ActivityIndicator is used to inform the user that data is being fetched, improving the user experience.
- **Error Handling**: Error messages are displayed to the user to indicate issues with the fetch operation, enhancing transparency.
- **Responsive Design**: The layout is adjusted using Platform.select and conditional styles to ensure a good user experience on both web and mobile platforms.
- **Visual Appeal**: An ImageBackground is used on the HomeScreen for visual appeal, with an overlay to ensure text readability.
- **Input Handling:** The input fields support auto-focus transition on Enter key press, enhancing the user experience by making the form navigation more intuitive.
- **Keyboard Avoidance**: Utilizes KeyboardAvoidingView to adjust layout when the keyboard is shown on iOS devices, ensuring that input fields remain visible.

Testing

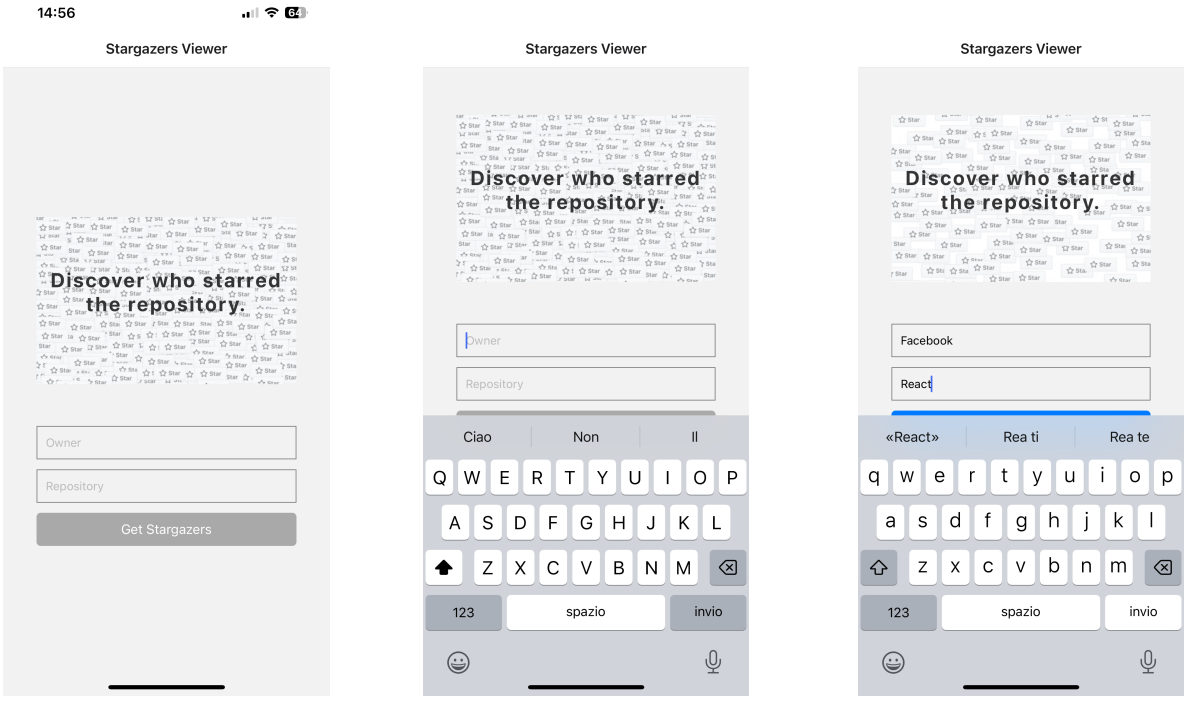
The project includes a dedicated test folder (__tests__) containing the following test files:

- **HomeScreen.test.js :** Verify that the Home screen renders correctly with essential UI elements visible.
- **StargazersScreen.test.js:** Verify that the Stargazers screen renders correctly with the provided stargazers data

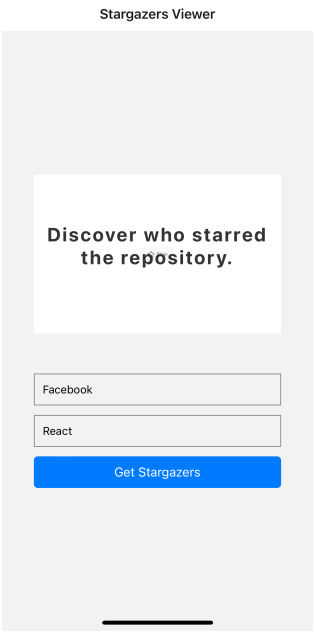
UI Screens

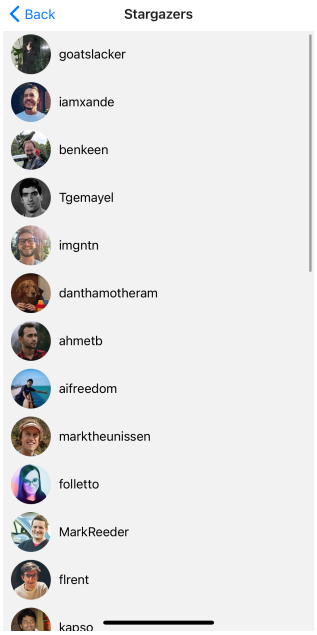
IOS - Standard Flow

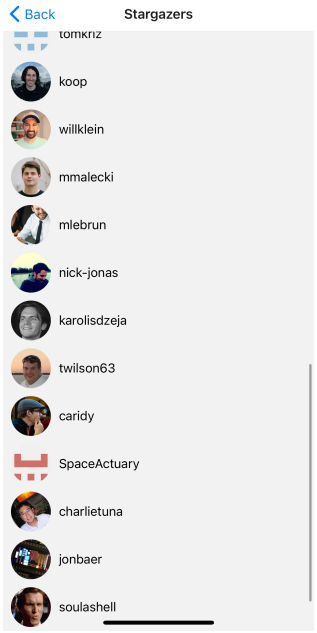
Below are shown the initial screen with button disabled, the initial screen with the keyboard displayed and a screen with the inputs completed.



Below we find a screen with button ‘Get Stargazers’ enabled and the screens, that appears after pressing the button, with the list of stargazers, scrolled using the scroll bar.



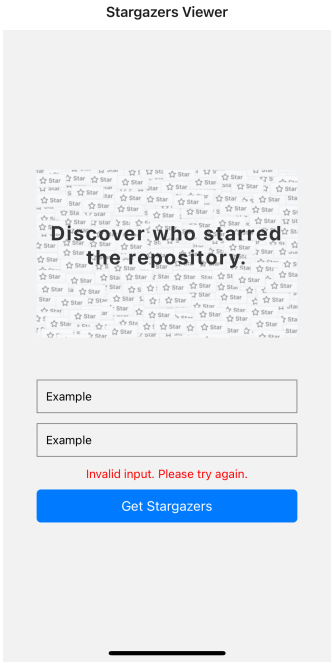




This flow is tested on IOS platform.

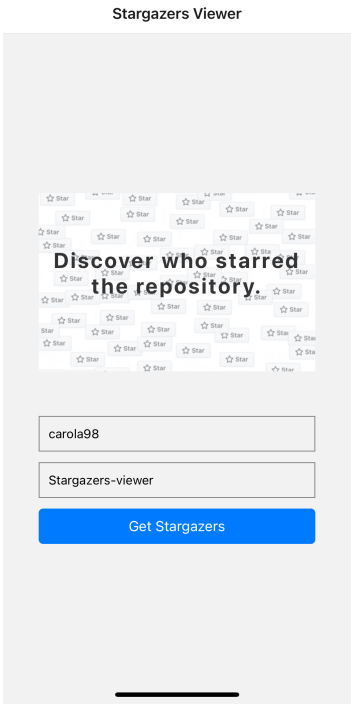
IOS - Error Flow

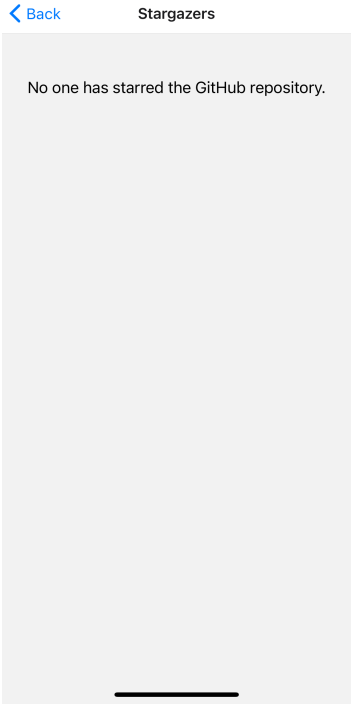
Next to the scenario where the inputs are incorrect on IOS.



IOS - No Data Flow

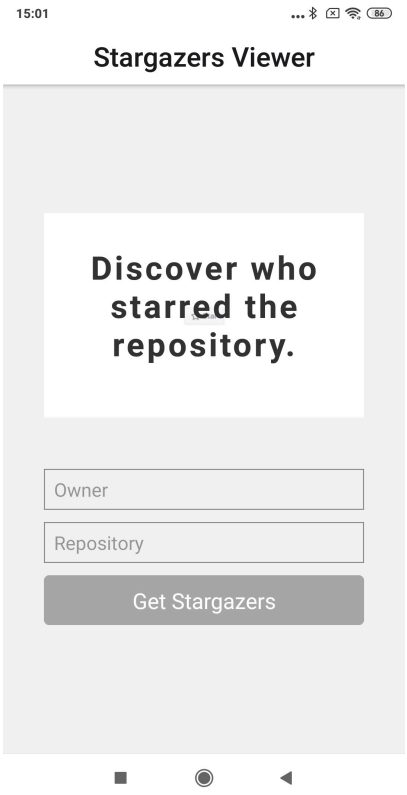
Beside the case where no one has starred the GitHub repository on IOS.

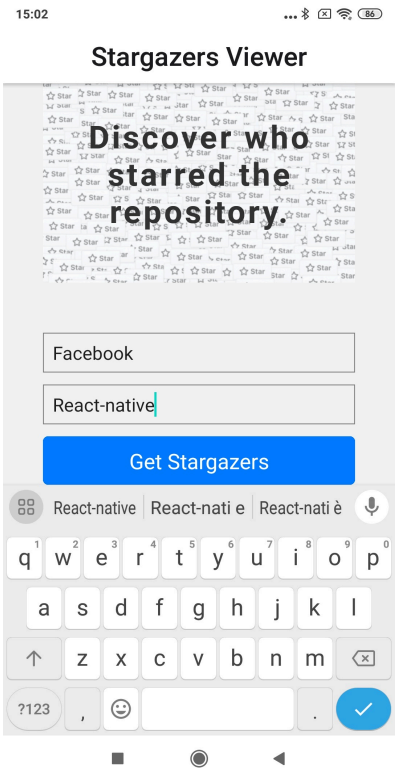




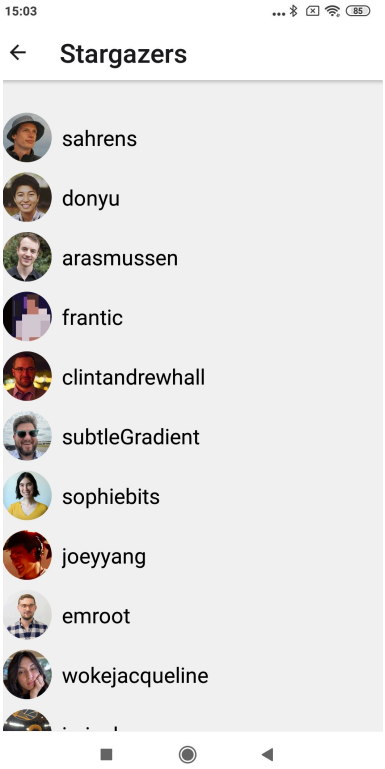
Android - Standard Flow

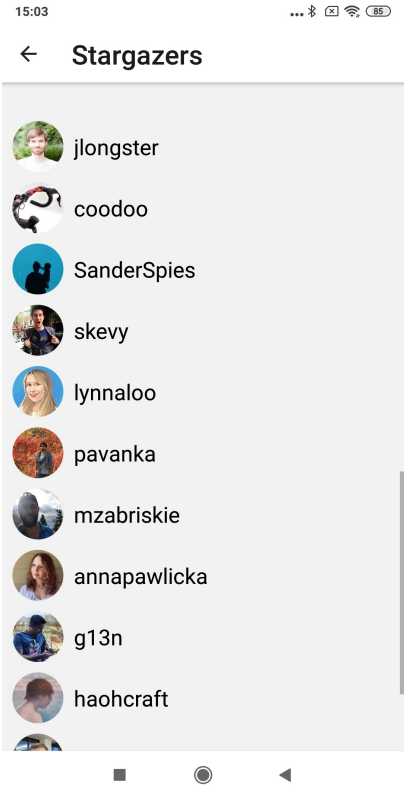
Below are shown the initial screen with button disabled, the initial screen with the keyboard displayed and the inputs completed.





Here we see the data obtained, scrollable with a scrollbar.

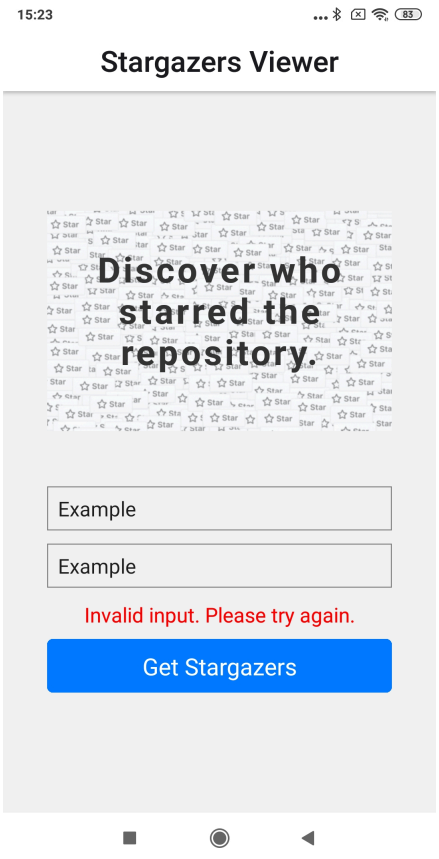




This flow is tested on android platform.

Android - Error Flow

Next to the scenario where the inputs are incorrect on Android.



Android - No data Flow

Below the case where no one has starred the GitHub repository on Android.