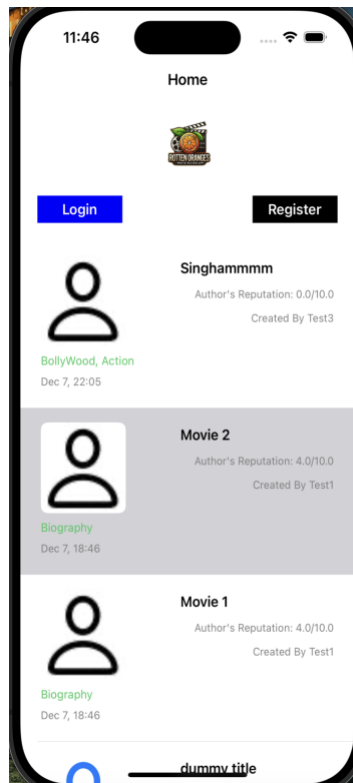


Github Repo: <https://github.com/DarshanBhokare/RottenOrangesMobileApp> (API Key is not pushed to the repo)

Initial App Screen:



Register Flow:

➤ User Registration Screen:

The user is presented with a registration screen where they need to provide the following details:

- Name
- Photo (using camera or gallery)
- Type (User or Critic) via a picker menu
- Email
- Password and Confirm Password

➤ Role Type Selection:

- The user selects a role type from a picker menu (User or Critic).
- Based on the selection, if the role is **Critic**, additional options for expertise tags are displayed.

➤ Expertise Tag Input:

- If the user is a Critic, they can enter their expertise area (e.g., "Action", "Horror") using a text field, which suggests matching categories as they type.
- The user can add expertise tags to their profile by tapping a button.

➤ Profile Photo Selection:

- The user can select a profile photo by tapping a button, either using the camera or picking from the gallery.

➤ Validation:

- The entered email is validated for correctness.
- All fields (name, email, password, and confirm password) are checked to ensure they are filled out.
- The password and confirm password fields must match.

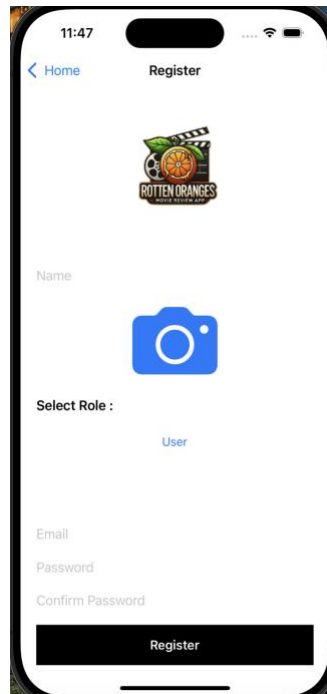
➤ **User Creation:**

- Upon successful validation, the user is registered using Firebase Authentication with email and password.
- A profile photo (if selected) is uploaded as part of user creation in Firebase Firestore.

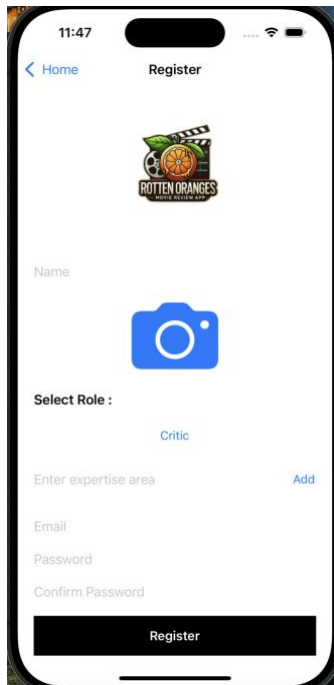
➤ **Completion and Redirection:**

- If user registration is successful, the app logs the user out and redirects them to the login screen.

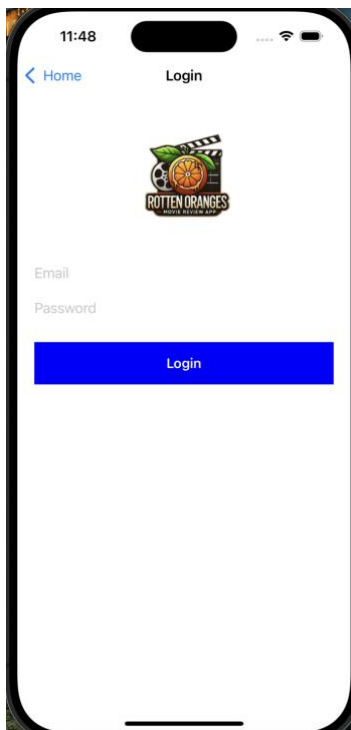
Screen for User:



Screen for Critic:



Login Flow:



➤ Login Screen Layout:

- The user sees a login screen with input fields for **email** and **password**, and a **login button**.

➤ Email and Password Input:

- The user enters their **email** (which is converted to lowercase) and **password** into the respective text fields.

➤ **Keyboard Dismissal:**

- A tap gesture recognizer is added to dismiss the keyboard when the user taps outside of the text fields.

➤ **Login Button Action:**

- When the user taps the **login button**, the following steps occur:
 - The email is validated to check if it follows the correct email format using a regex.
 - If the email format is invalid, an alert message is shown to the user.
 - If either the email or password is empty, an alert message is shown prompting the user to enter both fields.

➤ **Authentication:**

- The app attempts to log the user in using **Firebase Authentication** with the provided email and password.
- If the login fails (e.g., due to invalid credentials), an alert message is shown, and the error is logged.
- If the login is successful, the user is redirected to the **TabsViewController**.

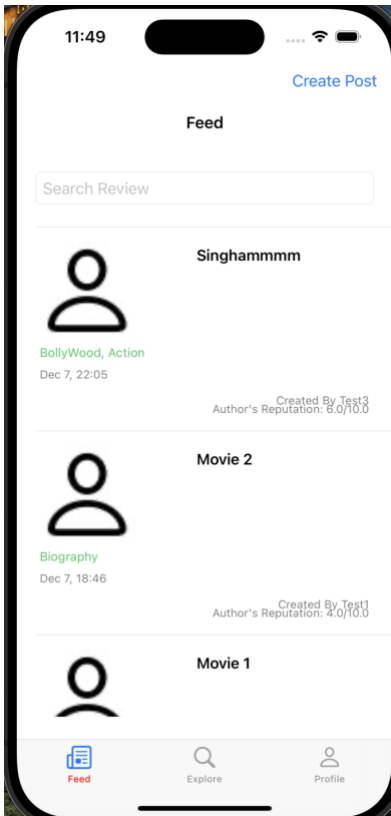
➤ **Navigation:**

- If the user is already logged in (i.e., there's an authenticated user in Firebase), they are automatically redirected to the **TabsViewController**.
- A **register button action** is implemented (though not used in this code) to navigate to the **RegisterViewController** for new users to sign up.

➤ **Alerts:**

- Error messages are displayed via **UIAlertController** for invalid input, failed login, or missing credentials.

Home Screen:



➤ UINavigationController:

- The primary container for managing the tab bar interface with multiple view controllers.

➤ Profile Image (UIImageView):

- A placeholder for the user's profile image, though it is not fully utilized in the code.

➤ Navigation Controllers:

- **Feed Navigation Controller:** Manages the navigation stack for the **FeedViewController**.
- **Explore Navigation Controller:** Manages the navigation stack for the **ExploreViewController**.
- **Profile Navigation Controller:** Manages the navigation stack for the **ShowProfileViewController** (dynamically created based on user data).

➤ Tab Bar Items:

- Each tab is represented by a **UITabBarItem**, which includes:
 - **Feed Tab:** With a newspaper icon.
 - **Explore Tab:** With a magnifying glass icon.
 - **Profile Tab:** With a person icon.

➤ User Role and Profile Image:

- The user's role and profile image are fetched from Firebase and used to personalize the experience (e.g., showing "Create Post" if the user is a **Critic**).

➤ "Create Post" Button: (Only for Critics)

- Appears in the navigation bar if the user has the "Critic" role. It navigates to the **NewPostViewController** when tapped.

➤ Notification Observer:

- Listens for the **logoutCompleted** notification to trigger user logout actions and dismiss the current view controller.

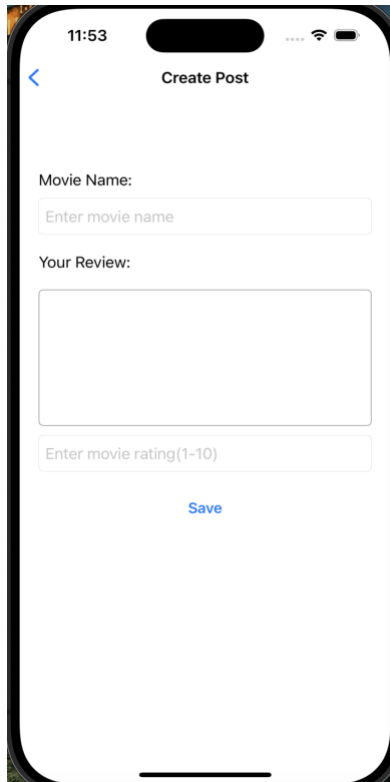
➤ **Tab Bar Customization:**

- Modifies the appearance of the tab bar items, ensuring the images are always rendered in their original color and the selected item has red text.

➤ **Delegate Methods:**

- **UITabBarControllerDelegate:** Ensures proper handling when switching tabs, including reloading data in **FeedViewController** and **ExploreViewController** when their respective tabs are selected.

Critic can create post to post review about a movie.

A screenshot of a mobile application interface for creating a post. The status bar at the top shows the time 11:53, signal strength, and battery level. The app's navigation bar has a back arrow on the left and the title "Create Post" in the center. The form contains three main sections: "Movie Name:" with a text input field containing the placeholder "Enter movie name"; "Your Review:" with a large text area; and a rating input field with the placeholder "Enter movie rating(1-10)". At the bottom of the form is a blue "Save" button.

Users can rate other reviewers which will calculate the reputation points for reviewer(Average rating)



In the explore tab you can follow a particular reviewer which will then populate in the feed section

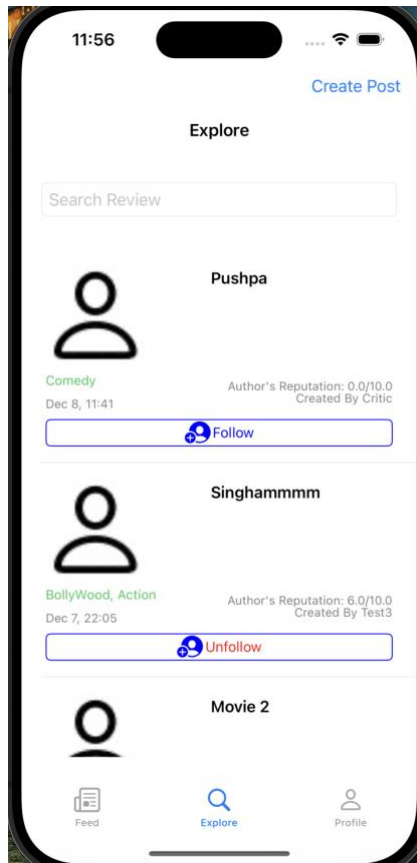
➤ Main Components:

- **Search Bar:** For searching posts based on title or tags.
- **Table View:** Displays a list of posts fetched from Firebase, including details like the title, content, and author.

➤ User Interactions:

- **Search:** As the user types in the search bar, posts are filtered dynamically based on the entered text.
- **Follow/Unfollow:** Users can follow or unfollow authors of posts directly from the list, with the follow button changing based on the action.
- **Post Selection:** Users can tap on a post to view its detailed review.

Data Source: Posts are fetched from Firebase Fire store, sorted by timestamp, and displayed in the table view.



Feed section:

➤ **Main Components:**

- **Search Bar:** Allows users to filter posts by title or tags.
- **Table View:** Displays a list of posts fetched from Firebase, including details like title, content, author, and image.

➤ **User Interactions:**

- **Search:** As users type in the search bar, posts are filtered dynamically based on title or tags.
- **Post Selection:** Users can tap on a post to view its detailed review.

➤ **Data Source:**

- Posts are fetched from Firebase Fire store, specifically from authors that the current user follows.

➤ **Fetching Data:**

- User details are retrieved to identify the authors they follow.
- Posts from followed authors are fetched and displayed, sorted by timestamp.

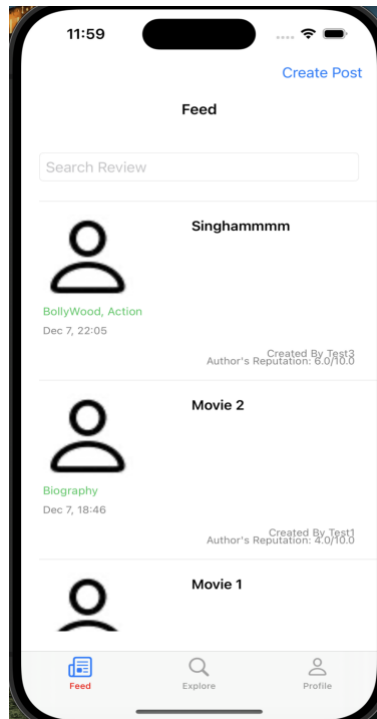
➤ **Dynamic Updates:**

- The table view reloads when new posts are fetched or when search results change based on user input.

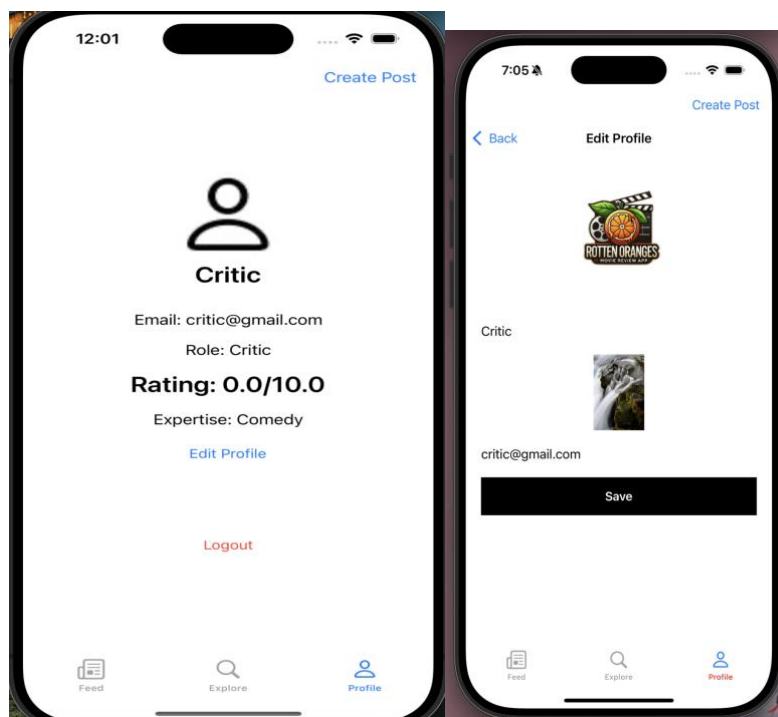
➤ **Table View:**

- Configured to display posts with dynamic heights.

- Cells are populated with post data, including author details fetched asynchronously.
- **Filtered Posts:** When text is entered into the search bar, posts are filtered by matching tags or title. The table view shows the filtered posts or all posts if the search bar is empty.



User profile section and Edit Profile Section:



➤ **View Initialization:**

- Pre-populates fields with user data (name, email, profile image) when the view loads.

➤ **Image Selection:**

- User taps the profile image button to select an image:-
 - **Camera:** Take a new photo.
 - **Gallery:** Select an existing photo.
- The selected image is displayed on the button.

➤ **Form Submission:**

- User enters name and email, optionally selects a profile image.
- Tap "Save" button:
 - **Validation:** Checks email format, ensures name/email are not empty.
 - **Image Upload:** Uploads selected image to Firebase Storage, retrieves the image URL.
 - **Firestore Update:** Updates Firestore with name, email, and image URL.

➤ **Error Handling:**

- Alerts shown for validation errors (e.g., invalid email, failed image upload).

➤ **Success Feedback:**

- Success message shown after profile update.
- Navigation back to the previous view.