

## **Flutter Final Assignment**

### **"JobFinder: A Smart Job Search & Recruitment App"**

#### **Step : 1**

##### **Create a New Flutter Project**

Run the following command in your terminal to create a new Flutter project:

**flutter create jobfinder**

Navigate into the project folder:

**cd jobfinder**

#### **Step: 2**

##### **Install Firebase tools**

Run the following command in your terminal to download the necessary packages:

**npm install -g firebase-tools**

Run the following command in your terminal to retrieve the flutter path:

**dart pub global activate flutterfire\_cli**

Run the following command in your terminal to login into firebase:

**firebase login**

Run the following command in your terminal to select the created database we created in firebase:

**firebase configure**

**Install the necessary dependencies.**

#### **Step: 3**

Run the following command in your terminal to Open project in vscode.

Code .

#### **Step: 4**

**Install Android studio and open**

**Navigate to more actions → virtual device manager → start the emulator**

#### **Step: 5**

**In vscode open terminal and type “flutter run”**

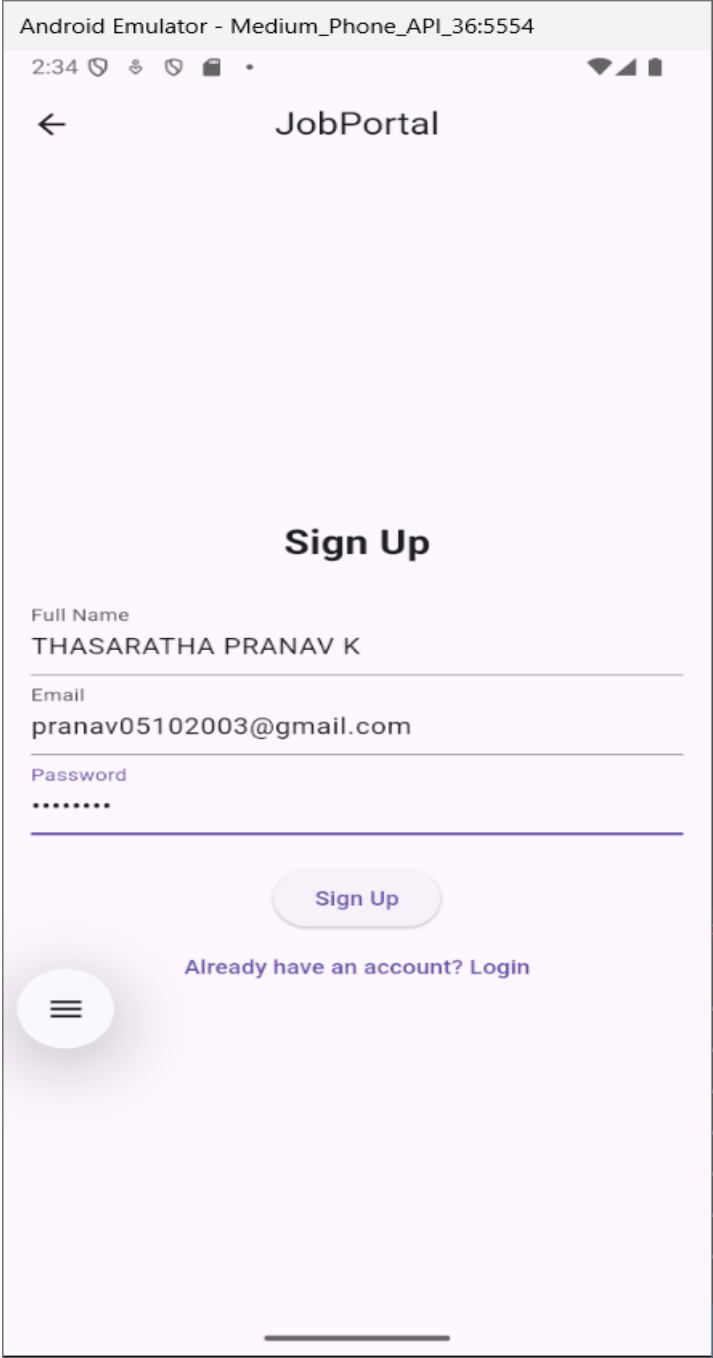
**Flutter app will be build and displayed in the emulator.**

**Step 6 :****Creating signup page :**

The Signup screen allows new users to create an account by entering their email and password. Key features:

- A text field for name input.
- A text field for email input.
- A password field with hidden input for security.
- A signup button that triggers Firebase Authentication to create a new account.

If successful, the user is redirected to the home screen.



The screenshot shows an Android emulator window titled "Android Emulator - Medium\_Phone\_API\_36:5554". The time is 2:34. The app is "JobPortal". The screen displays a "Sign Up" form with the following fields:

- Full Name:** THASARATHA PRANAV K
- Email:** pranav05102003@gmail.com
- Password:** Hidden with dots (.....)

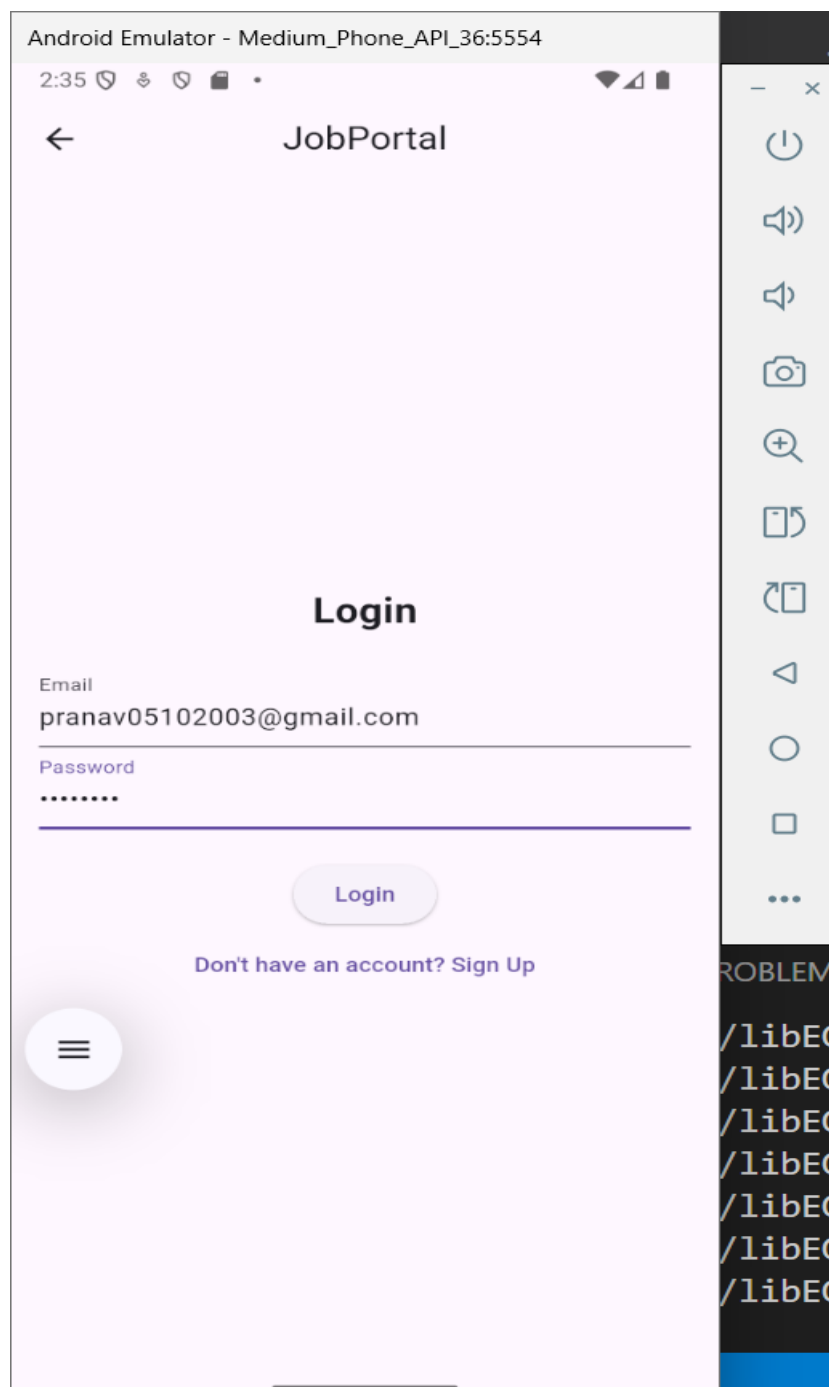
Below the fields is a "Sign Up" button. At the bottom, there is a link: "Already have an account? Login". A hamburger menu icon is visible in the bottom left corner.

**Step 7 :****Creating login page :**

The Login screen allows existing users to access their accounts. It includes:

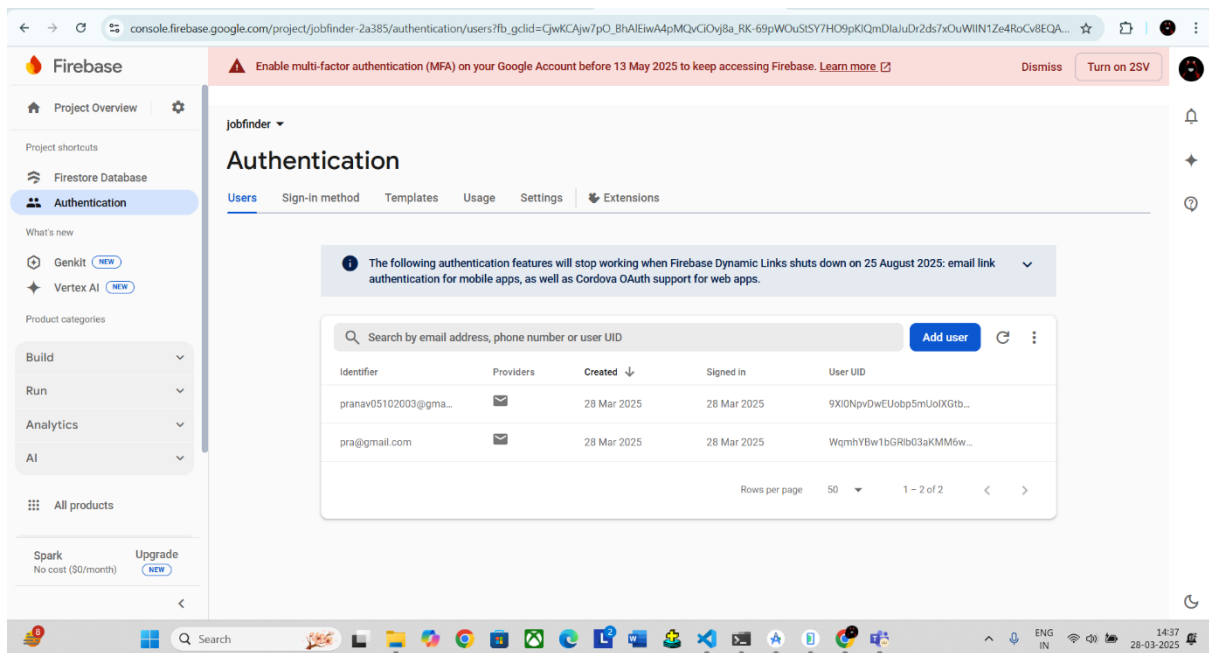
- An email input field
- A password field
- A login button that verifies credentials with Firebase
- Don't have an account sign up will hover to sign up page.

If authentication is successful, the user is logged in and redirected to the home page.



## Storing user information:

- I have created a Firestore database named jobFinder.
- In that I have enabled the signup method.
- If the user sign up in the app the data will be stored here.
- If user login then it will authenticate the e-mail and password if it is correct the home screen page opens.



The screenshot shows the Firebase Authentication console for a project named 'jobfinder'. The 'Users' tab is selected, displaying a table of users. A notification at the top states: 'The following authentication features will stop working when Firebase Dynamic Links shuts down on 25 August 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.'

Identifier	Providers	Created	Signed in	User UID
pranav05102003@gma...	📧	28 Mar 2025	28 Mar 2025	9XlONpyDwEUobp5mUotXGtb...
pra@gmail.com	📧	28 Mar 2025	28 Mar 2025	WgmhYBw1bGRbG3aKMM6w...

At the bottom of the table, it shows 'Rows per page: 50' and '1 - 2 of 2'.

## Step 8:

### Creating Home page :

- ✓ Displays a list of job openings fetched from an API.
- ✓ Each job listing includes:
  - Job Title
  - Company Name
  - Location
  - Salary Details
- ✓ Users can scroll through jobs

### Purpose:

- Provides an intuitive job browsing experience.
- Fetches job listings dynamically from the API.
- Allows users to navigate to **Job Details Page** by clicking a job.

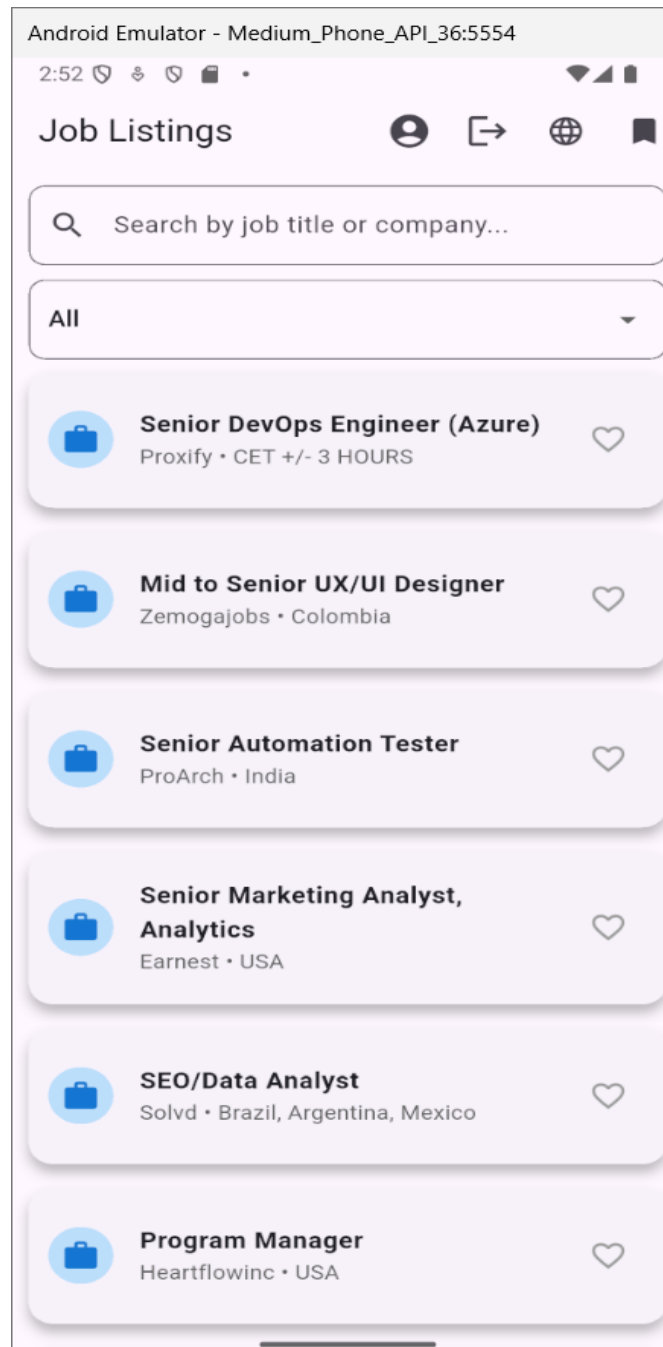
### Job details Screen:

Displays detailed job information, including:

- Job Title & Description
- Company Name & Logo
- Salary & Benefits
- Required Skills & Experience
- "Save Job" button to mark jobs as favorites.

### Purpose:

- Helps users understand job requirements before applying.
- Improves engagement with a user-friendly UI.
- Allows users to save jobs for later.

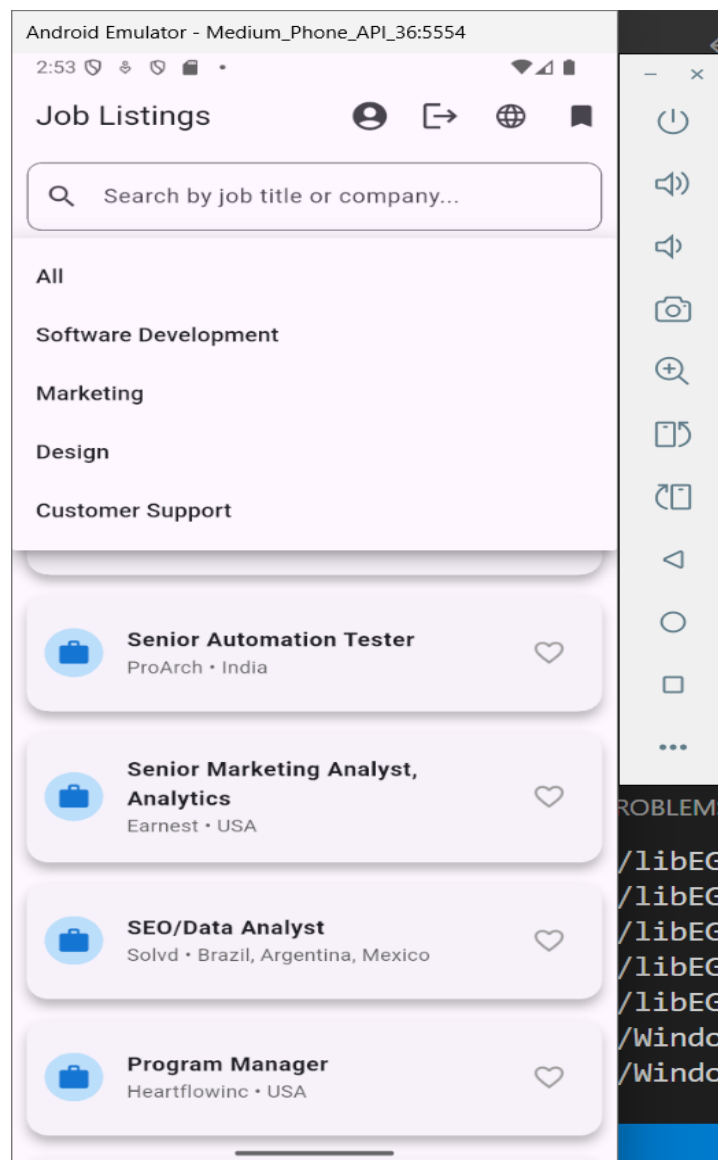


### Job Search & Filtering Screen

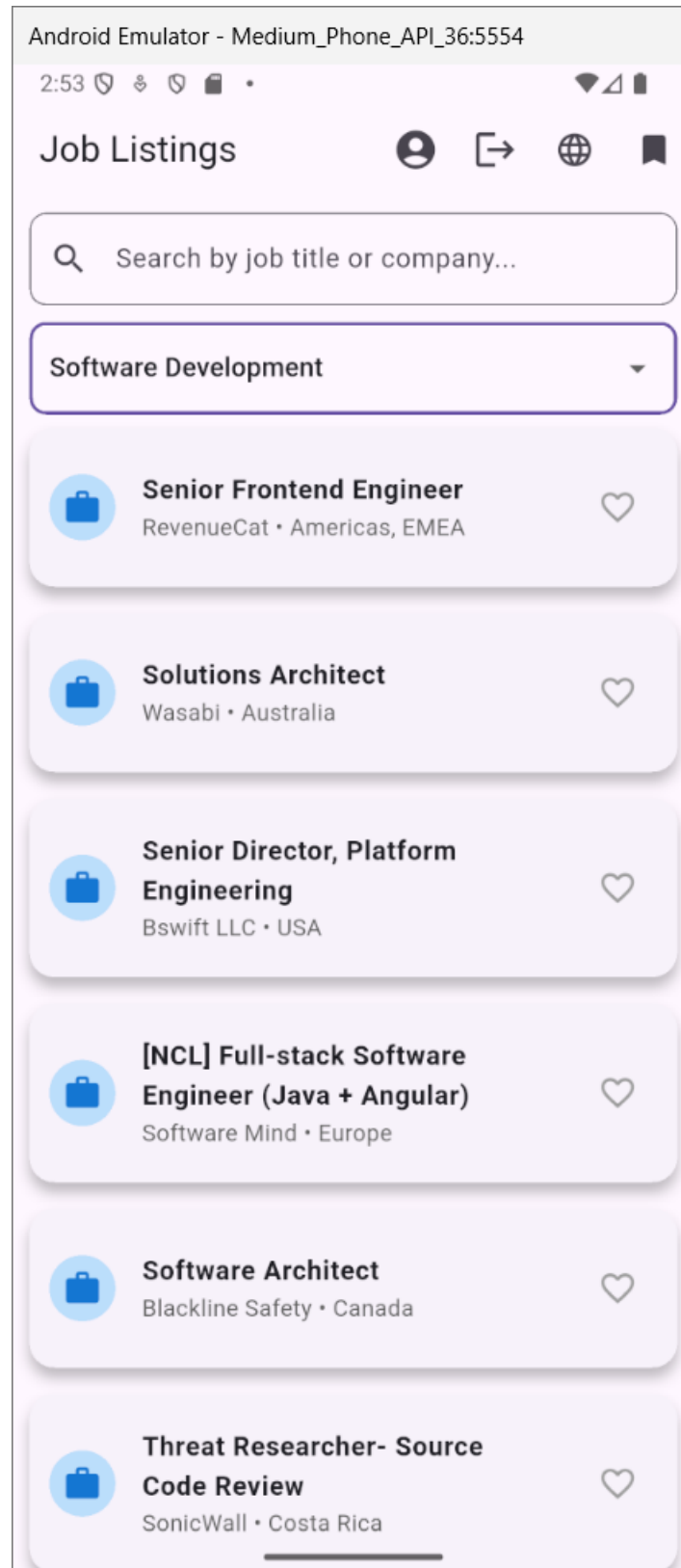
- Users can search for jobs by keywords (e.g., "Software Engineer").
- Filter jobs by category:
  - IT & Software
  - Marketing
  - Design
  - Finance, etc.

### Purpose:

- Enhances the job search experience.
- Allows users to find relevant jobs quickly.
- Improves usability by reducing the time spent browsing unnecessary listings.

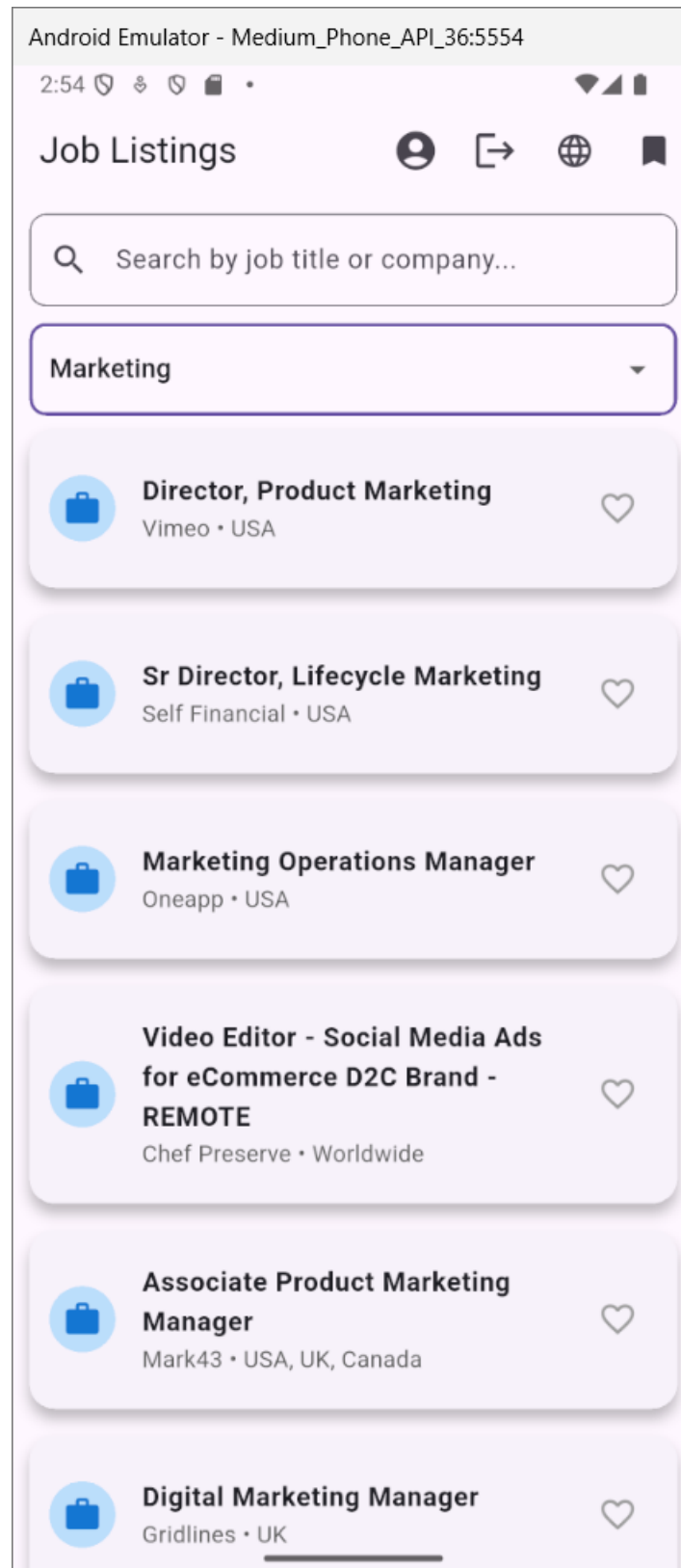


After selecting the required filter (**Software development**)

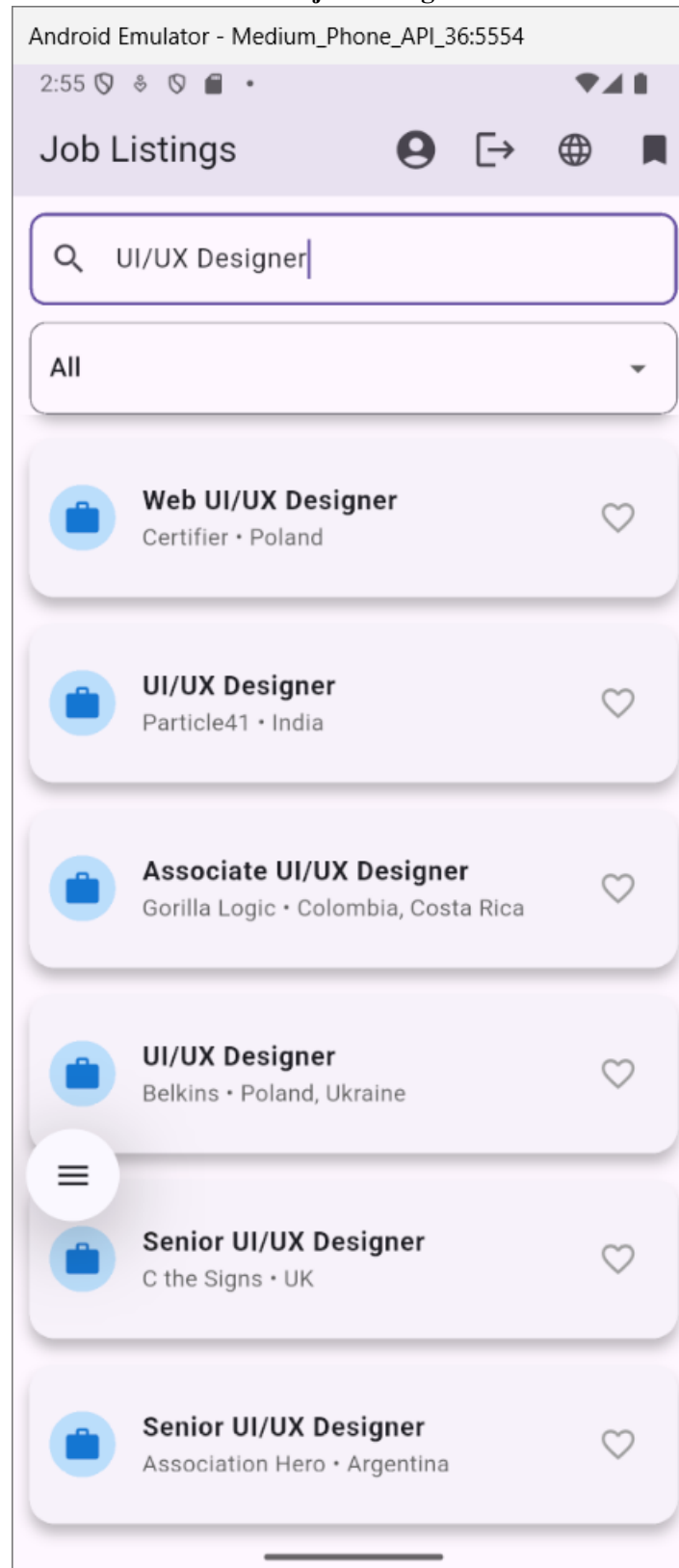




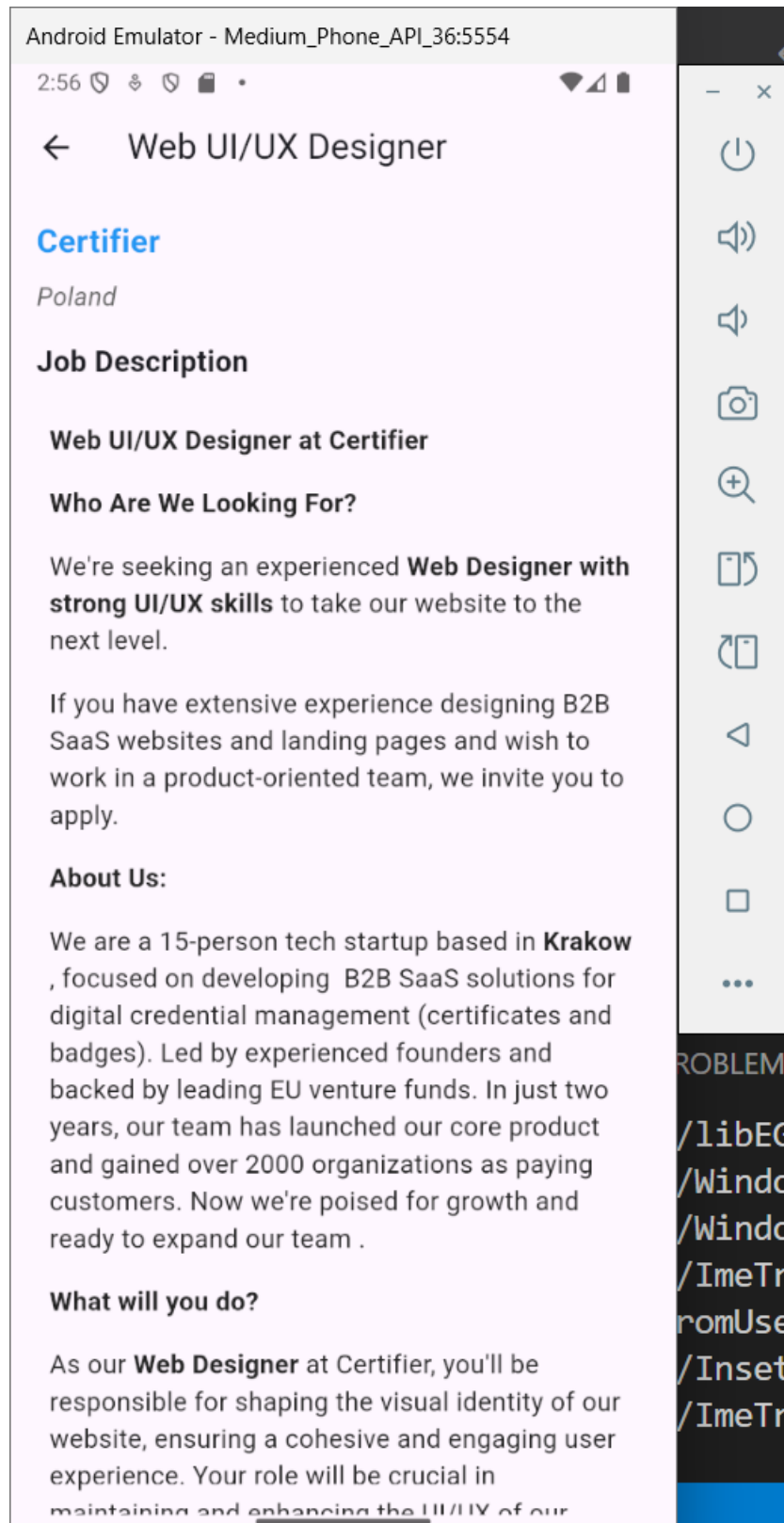
After selecting the required filter (**Marketing**)



**We can also search jobs using the search bar:**



Displaying job description inside each job everything is fetched from API.



**Step :9****Saved Jobs Screen (USING DATABASE)****Saving jobs:**

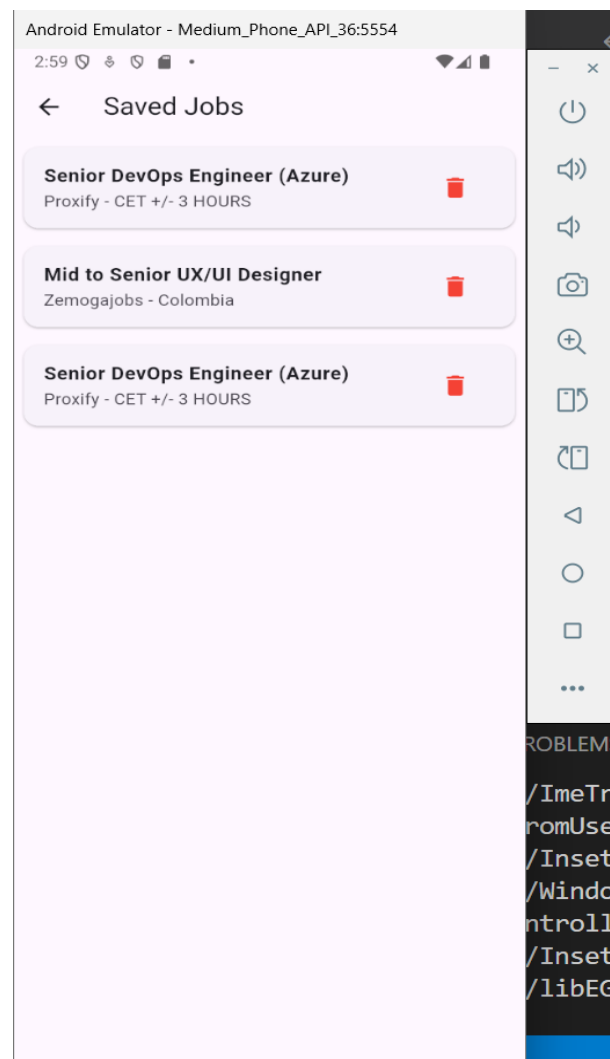
- If we click the heart icon the job will be stored into the database.  
Here I clicked the Senior DevOps Engineer (Azure) it shows a popup that it is saved.
- The save button in the top right will hover to saved jobs page when it is clicked.

**Saved jobs:**

- Displays jobs that the user has saved.
- Saved jobs are stored in Firebase Firestore.
- Users can click on a job to view full details or remove it from saved jobs.

**Purpose:**

- Allows users to keep track of interesting jobs.
- Helps users apply later when ready.
- Improves user experience by personalizing job recommendations.



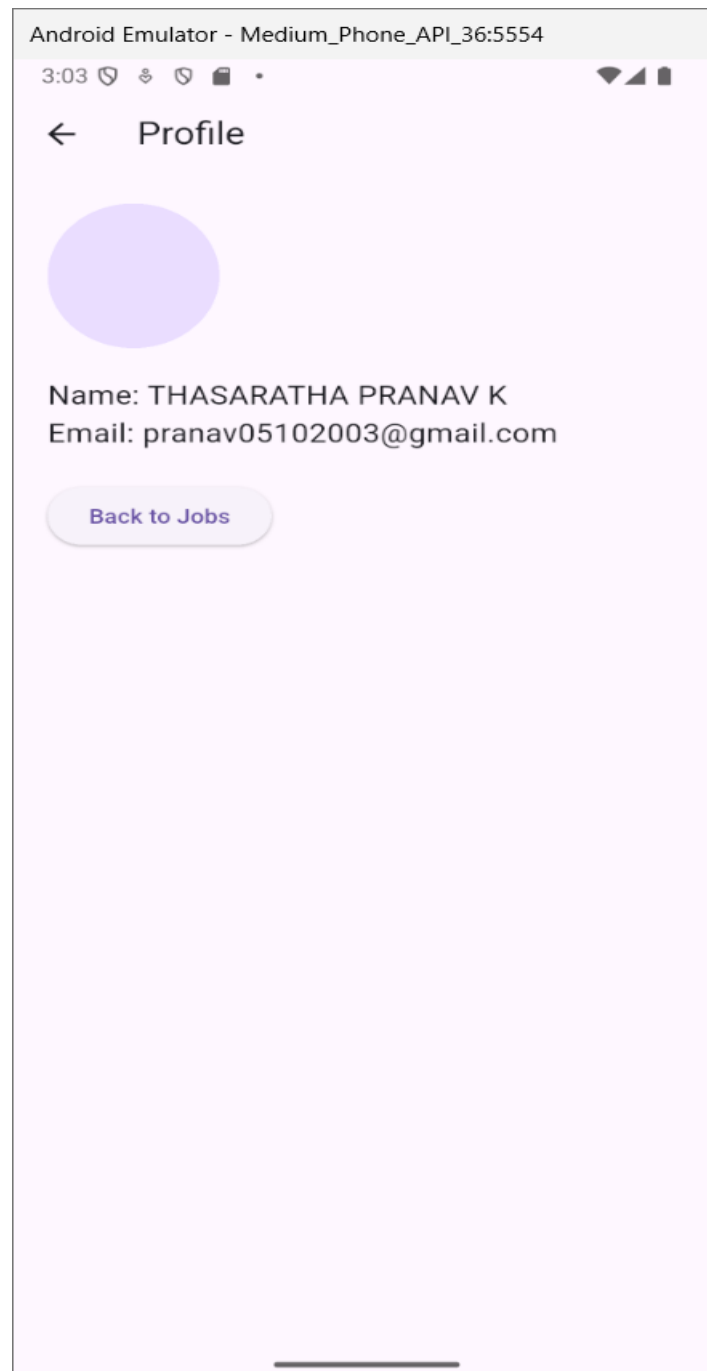
## Step :10

### User Profile Screen

- Displays user details, including:
  - Profile Picture
  - Name & Email
- Users can edit their profile or log out.

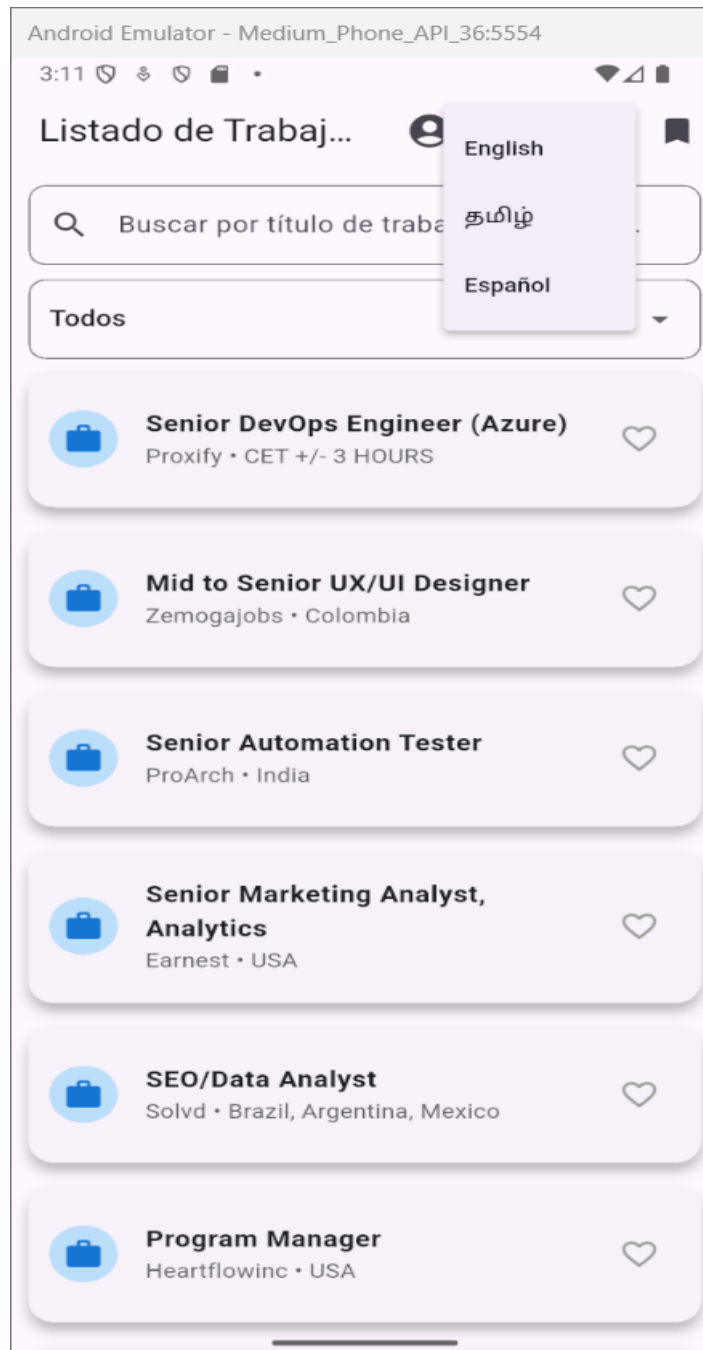
### Purpose:

- Personalizes the app experience.



**Localization:**

- Localization implementation for this a language switch button placed if we click any language clicked then it will be changed according to it.
- Here it is changed to Spanish.



Herer it changes into □□□□□.

