Diagram, logo

Description automatically generated

MEHRAN UNIVERSITY

OF ENGINEERING & TECHNOLOGY

JAMSHORO, PAKISTAN

***Mobile Application Development***

***Complex Engineering Problem***

***Report***

|  |  |
| --- | --- |
| **Roll No:** | **22SW007, 22SW037** |
| **Section** | **I** |

1. **Real World Problem Identification**

After any event such as a wedding, birthday party, university function, or conference, people capture hundreds of photos. However, these photos remain scattered across multiple devices and social media platforms.  
It becomes difficult to **collect, organize, and share** all the photos in one place while maintaining their quality and privacy.  
Traditional methods (like WhatsApp sharing or USB transfer) often **reduce image quality, mix photos from different events,** and **take a lot of time.**

1. **Proposed Solution**

**EventPix** is a photo-sharing application that solves these issues by providing a **centralized platform** for event photo collection and management.  
The app allows users to:

* Create a new event album and generate the event code.
* Join events easily by entering the event code shared by the owner.
* Upload photos directly to that event gallery.
* Owner can moderate the content and delete any unwanted or inappropriate photo from the event gallery.
* View, download(single and multiple) images
* Maintain privacy through code base access.
* Access high-quality images without compression.

This makes it simple for event organizers and participants to store all photos in one organized, secure, and easily accessible location.

1. **Data Storage (With justification for using a particular database)**

EventPix uses Firebase Firestore for storing user and event data on the cloud because our app requires multiple users to interact, upload, and download images.

We used Cloudinary for storing images on the cloud because it provides a free tier of 25GB storage, which is suitable for our app's requirements. Cloudinary also offers automatic image optimization and fast CDN delivery, ensuring quick loading times for users regardless of their location.

1. **[Optional Section] APIs/Packages/Plug-ins (if used with justifications for using them).**

We used several packages in EventPix to implement core functionalities. Below is a detailed explanation of each package and its purpose:

1. **firebase\_core, firebase\_auth, cloud\_firestore:** These are the core Firebase packages used to enable and implement Firebase services in our app, including authentication and storing user and event data on the cloud.
2. **image\_picker**  
   This package simplifies media selection compatibility across devices. It provides a straightforward API for accessing the gallery and camera. It is used in the EventGalleryScreen to select photos from the mobile gallery and upload them to the event gallery.
3. **cached\_network\_image**  
   This package caches images locally so users don't need to reload images from the server repeatedly. It reduces network calls and allows the app to display images even when offline, improving performance and user experience. It is used in the EventGalleryScreen's gallery grid view for efficient image display.
4. **http**  
   This package is required for making HTTP requests to Cloudinary's API for image uploads and downloads. It is used in CloudinaryService for uploading images and downloading them for storage.
5. **http\_parser**  
   This package is required for multipart file uploads to Cloudinary. It ensures proper content-type headers are set during image uploads.
6. **mime**  
   This package automatically identifies image formats (JPEG, PNG, etc.) to ensure files are processed correctly by both the app and server during upload or download operations. It is also used in CloudinaryService.
7. **intl**  
   This package formats dates effectively in a user-friendly format. It is used to display the event date in a format that is easy to read.
8. **permission\_handler**  
   This package is required for accessing storage and photos on modern Android and iOS devices. It ensures compliance with the latest security and privacy models by handling platform-specific permissions like manageExternalStorage for Android and photos for iOS. It is used in the download function to first ask for permission to access storage and then download the images.
9. **path\_provider**  
   This package allows downloaded images to be saved in organized folders. It is used to store images in the DCIM/EventPix folder.
10. **provider**  
    This package implements efficient state management across the app. Initially, we were not using Provider for state management, so we had to make database calls multiple times whenever we needed user ID, name, or other data. By using Provider, we reduced database calls and gave our app global access to user data.
11. **Issues and Bugs Encountered and Resolved during Development:**

During the development of EventPix, several issues and bugs were encountered. Each was carefully analyzed and resolved to ensure a stable and responsive application. Below is a summary of the major challenges faced and the solutions implemented:

**1. NDK Mismatch Error ("NDK not found")**

* **Cause:** The project was referencing to an outdated NDK version 26.3.11579264, which caused an error during the build.
* **Solution:** We updated the NDK on our development machine to version 27.3.13750724 and updated the local.properties file to point to the latest installed NDK path, which resolved the build error.

**2. Permission Error – "Permission Denied" while Saving Files**

* **Cause:** Android's new storage model (Scoped Storage) restricted direct access to device storage.
* **Solution:** We added the manageExternalStorage permission and implemented proper runtime permission checks for Android and photo access for iOS.

**3. Photo Grid Responsiveness Issue**

* **Cause:** The photo grid did not adapt correctly to different screen sizes.
* **Solution:** We implemented a responsive GridView layout using MediaQuery to dynamically adjust the column count based on screen width.

**4. Missed Validator During Event Join**

* **Cause:** User input for joining events was not validated, and no error message was shown to the user.
* **Solution:** We added proper input validation to ensure required fields are correctly filled before joining an event, along with user-friendly error messages.

**5. Images Downloaded but No Feedback on Success**

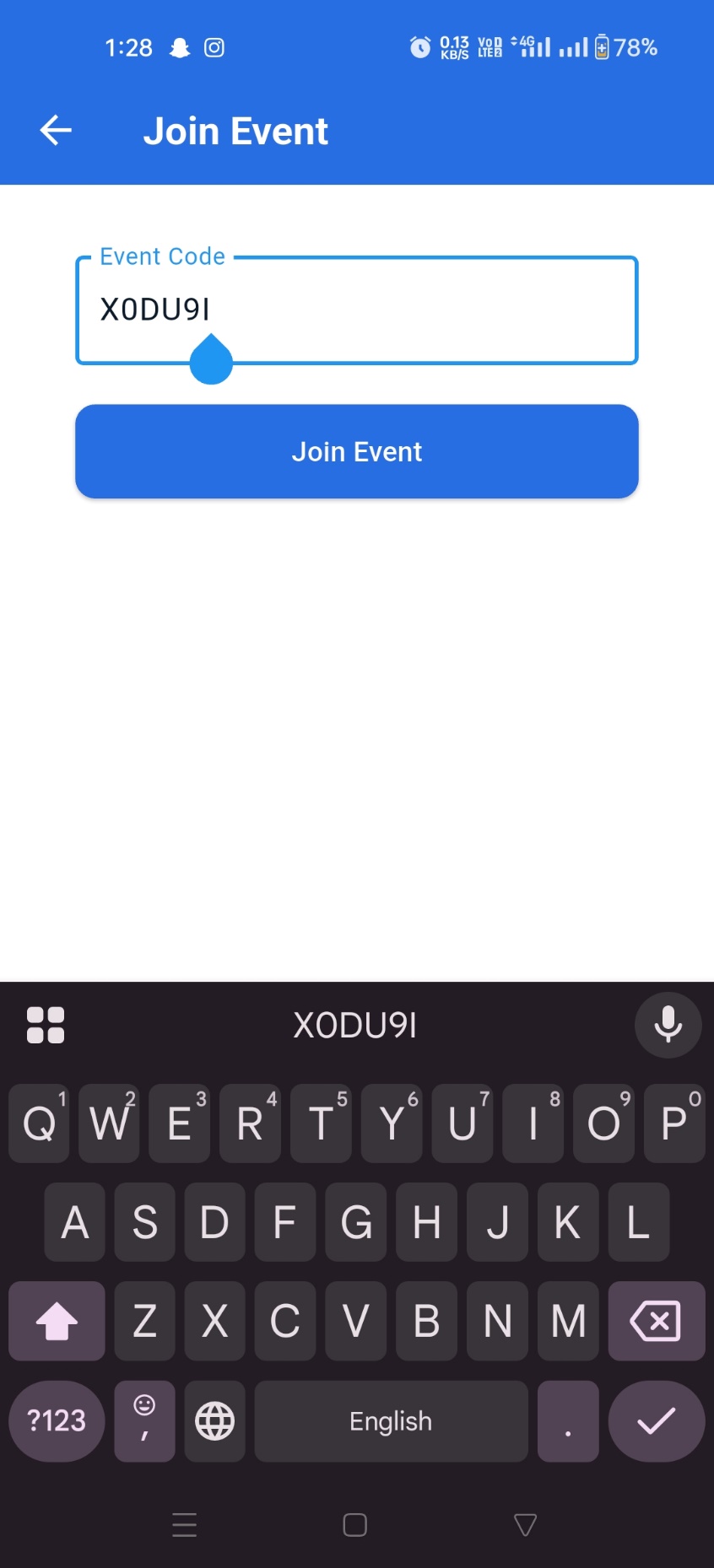
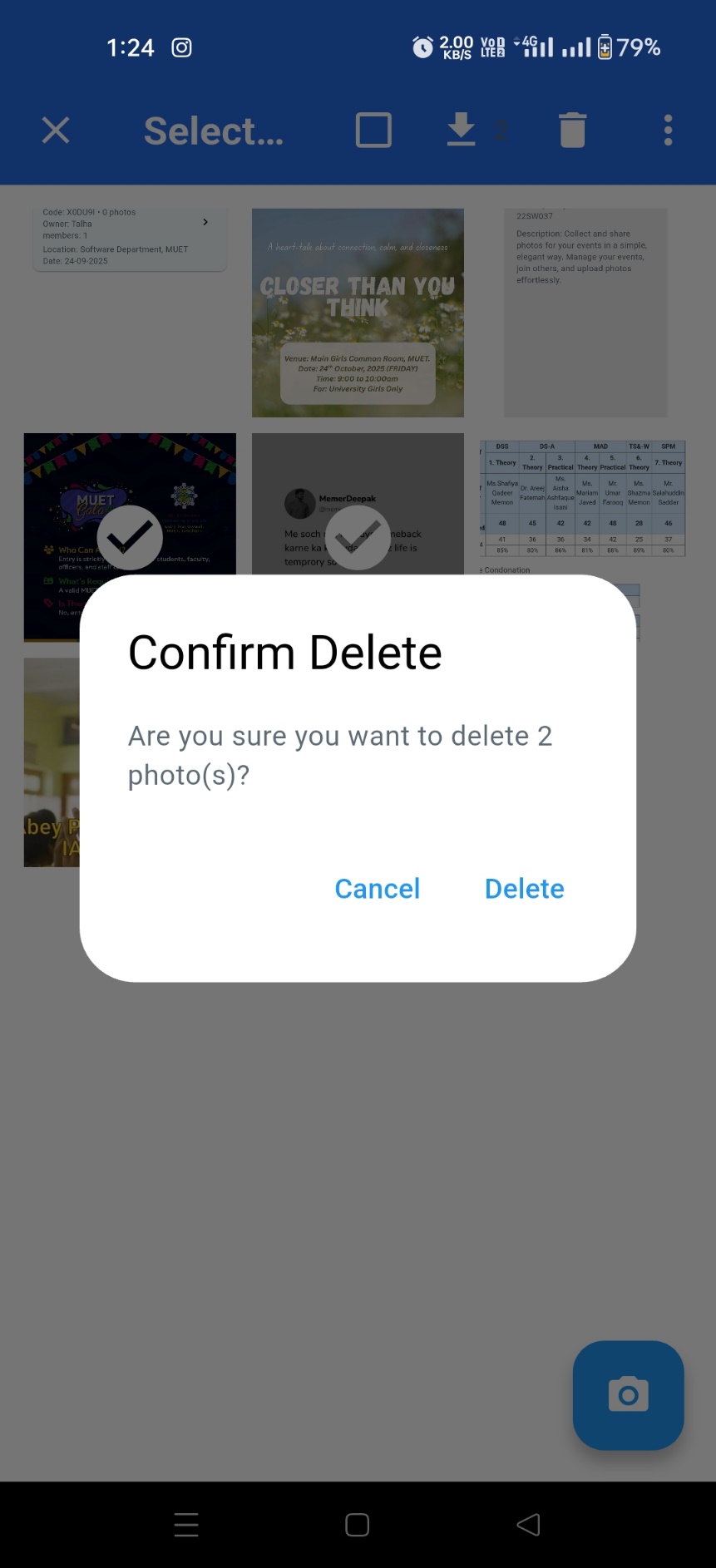
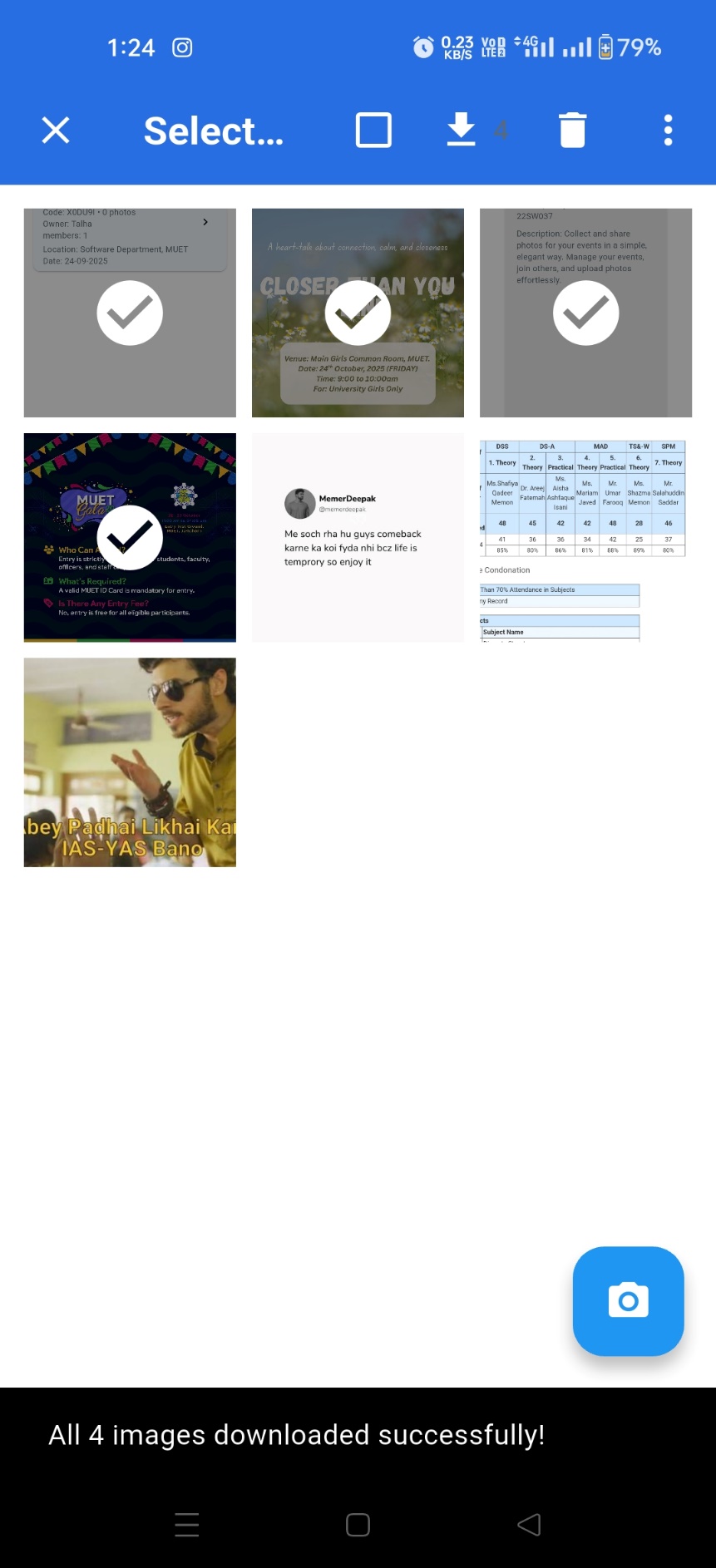
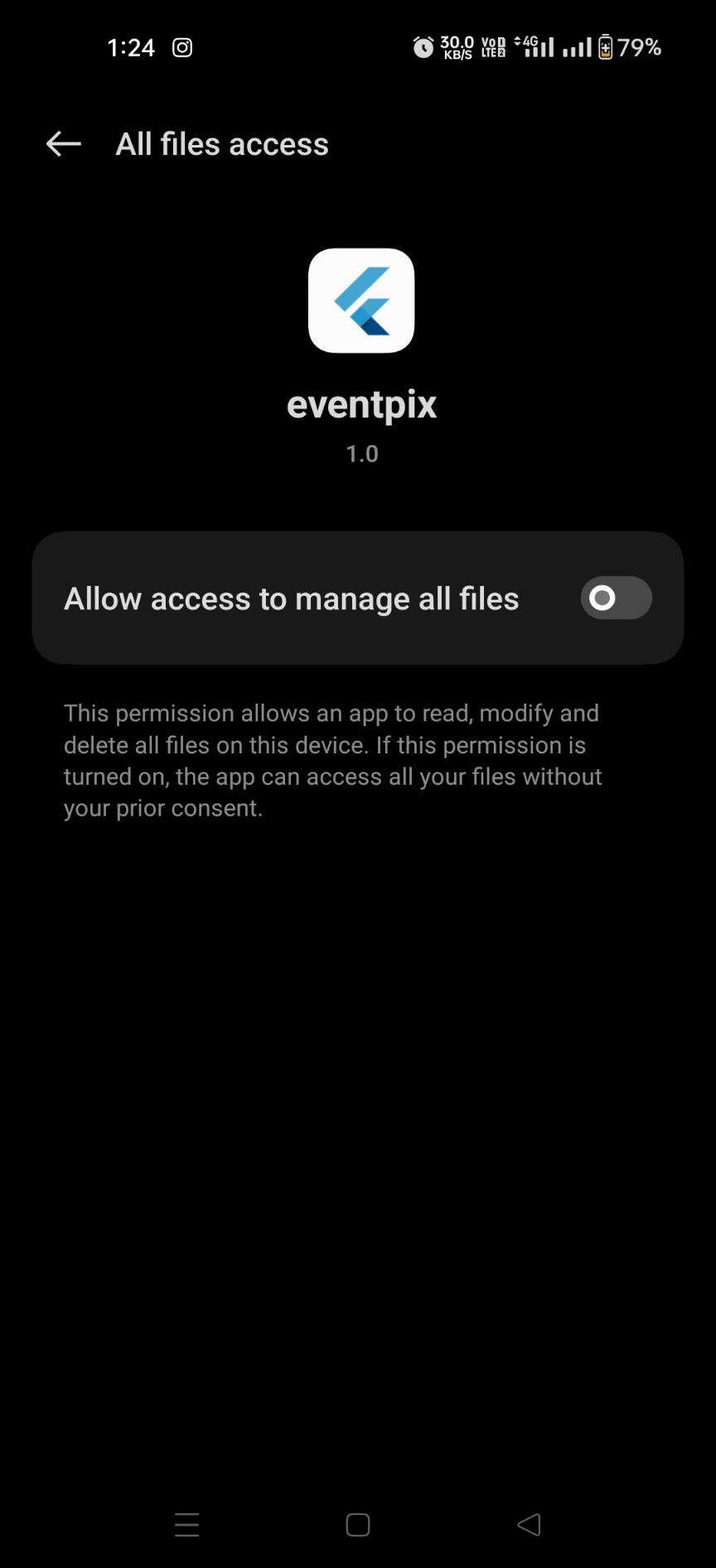
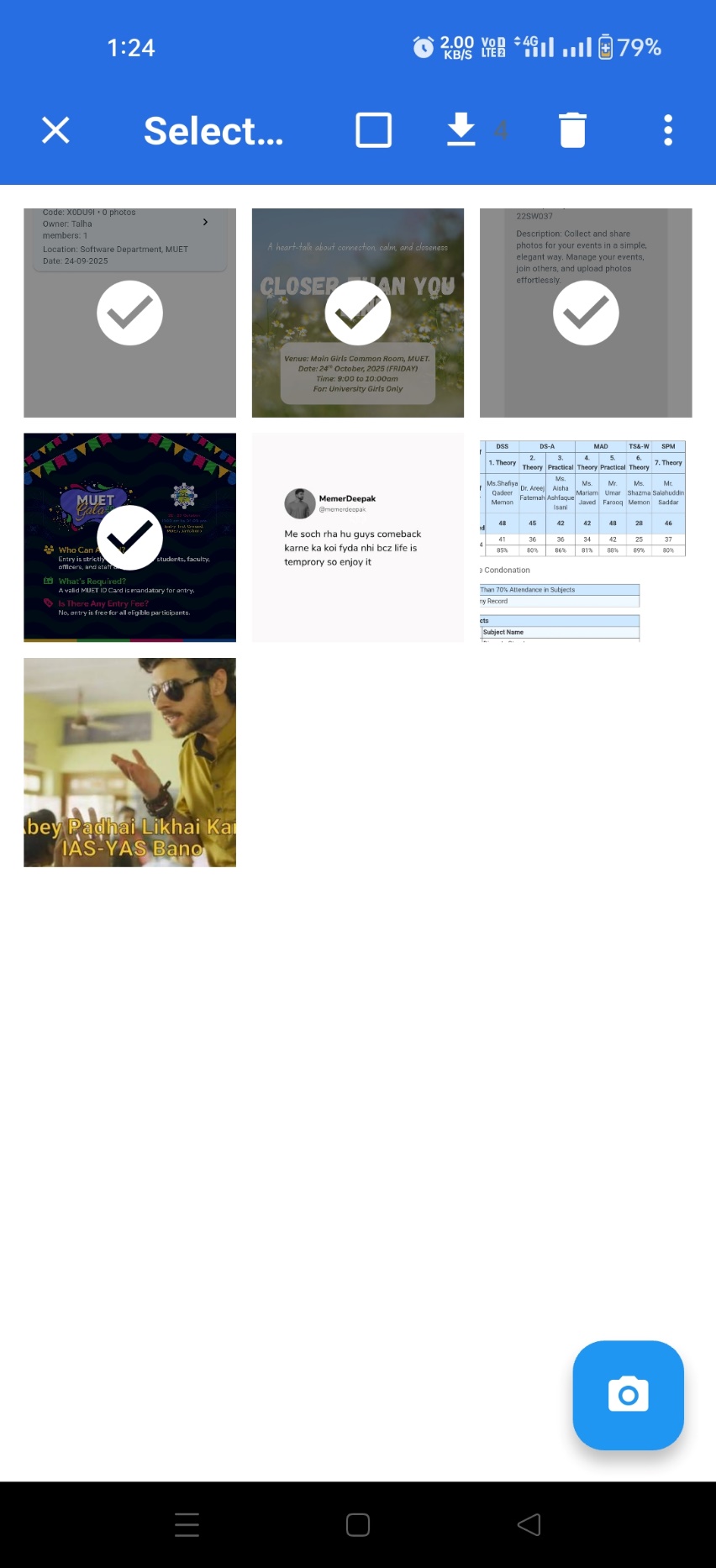
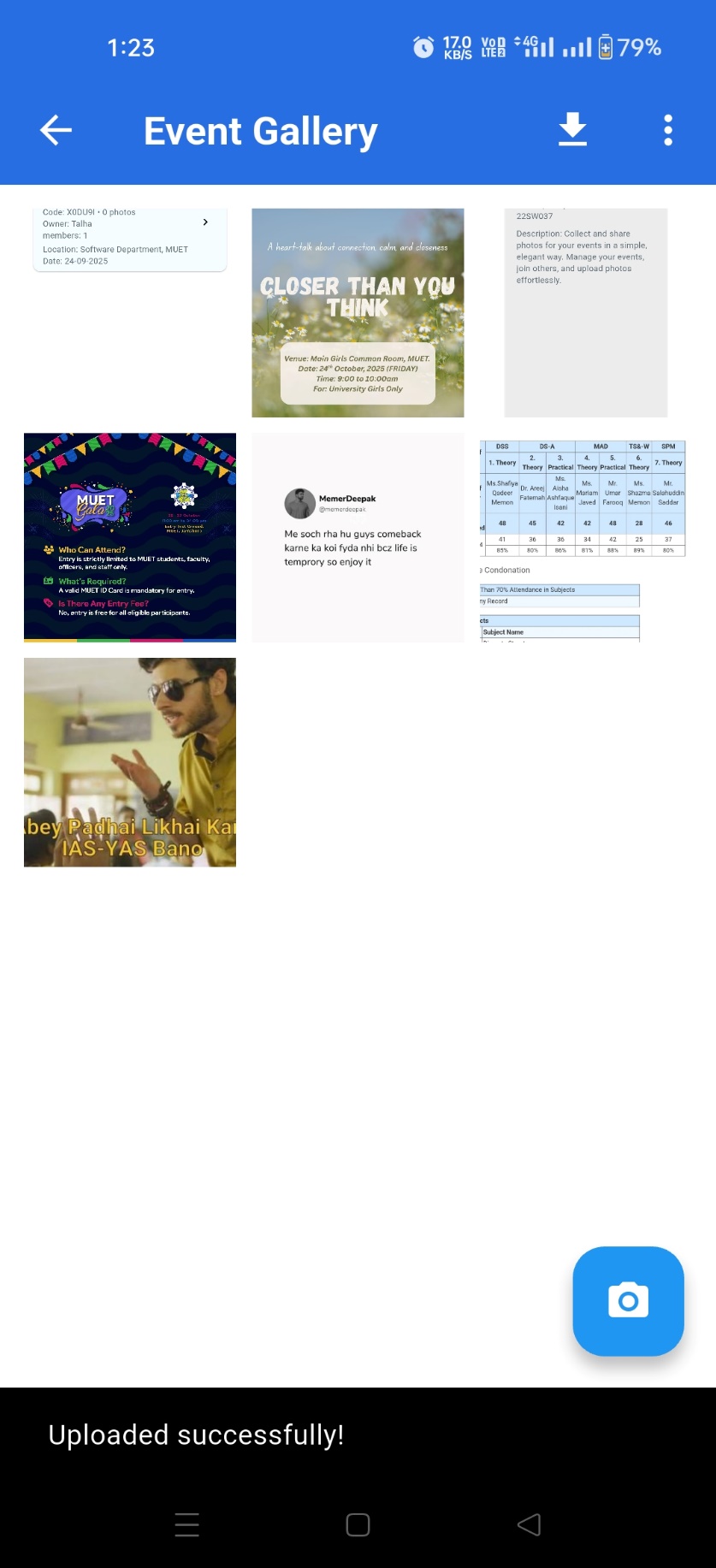
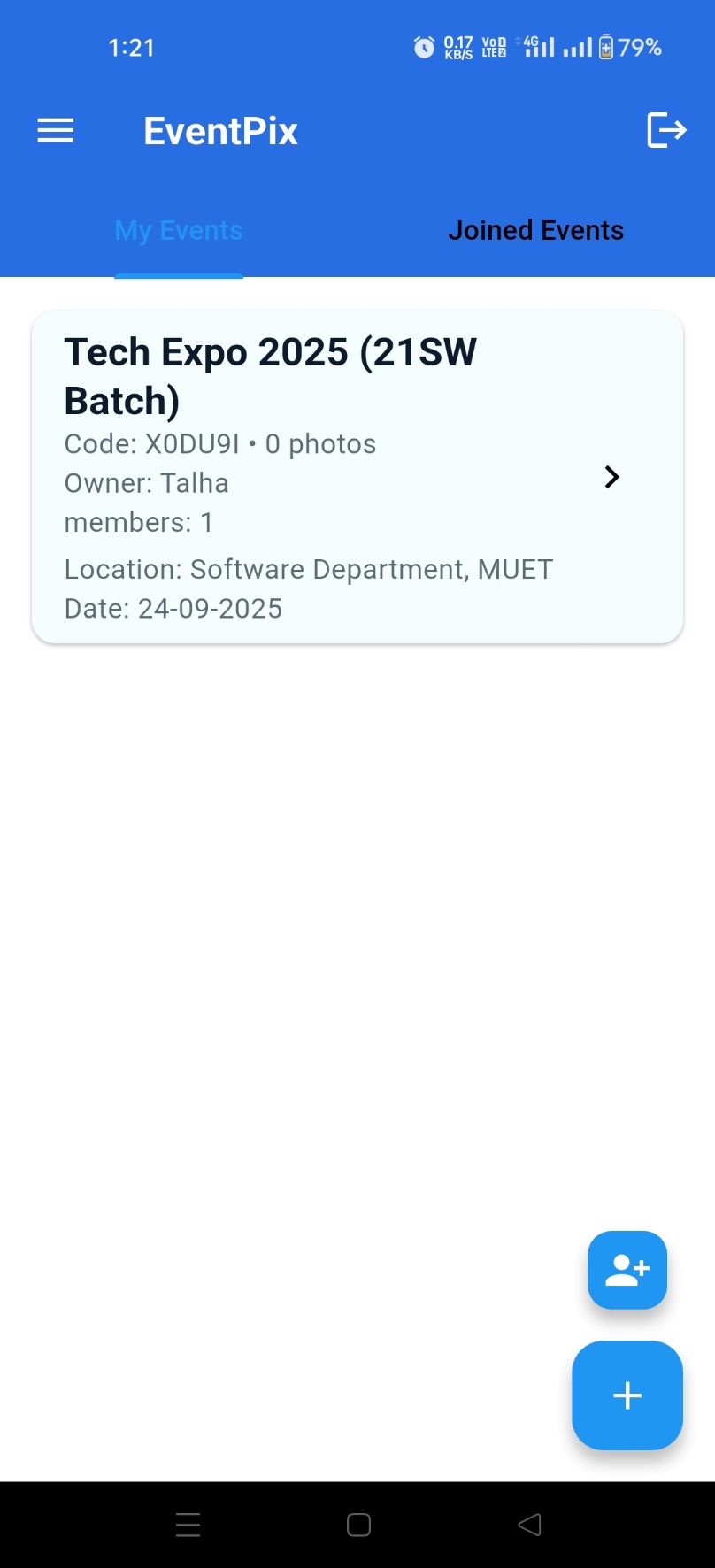
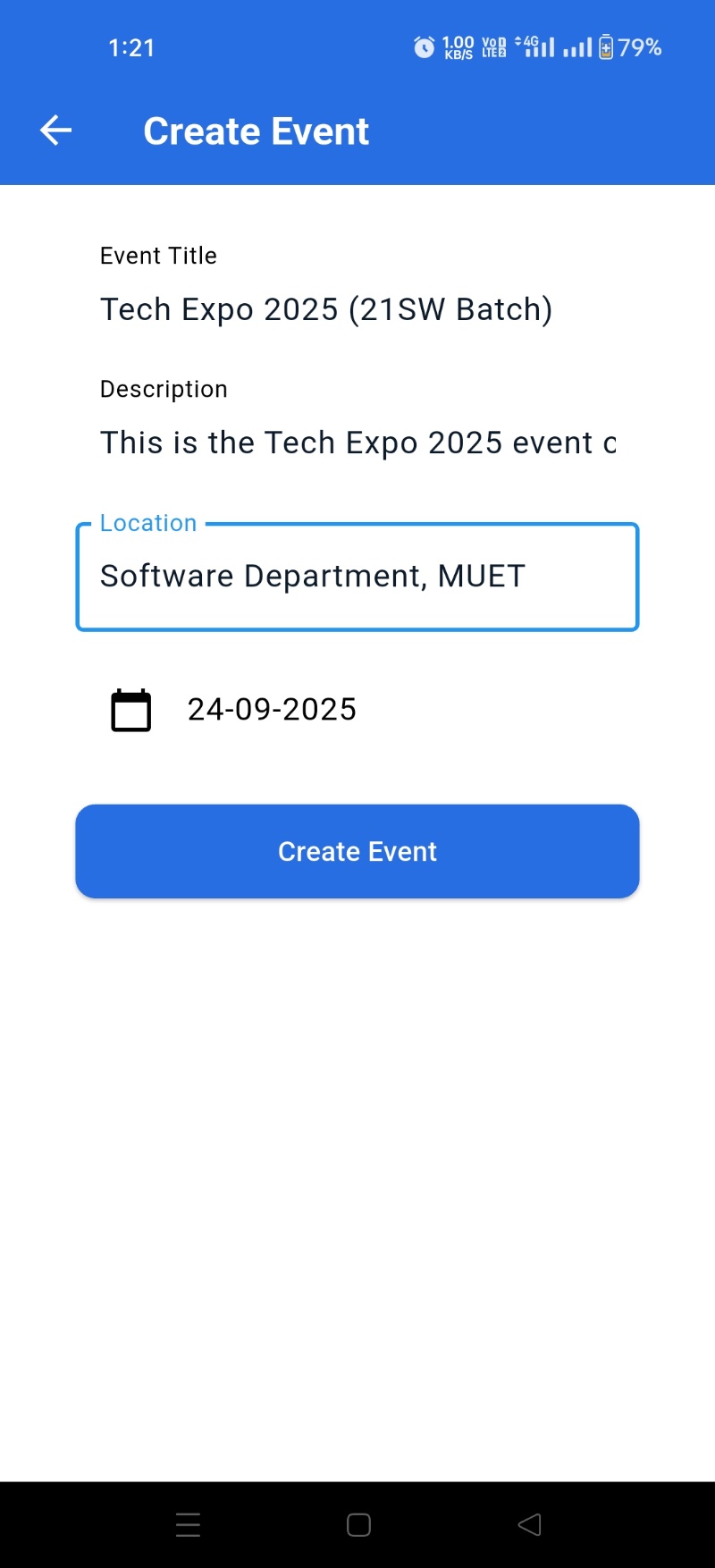
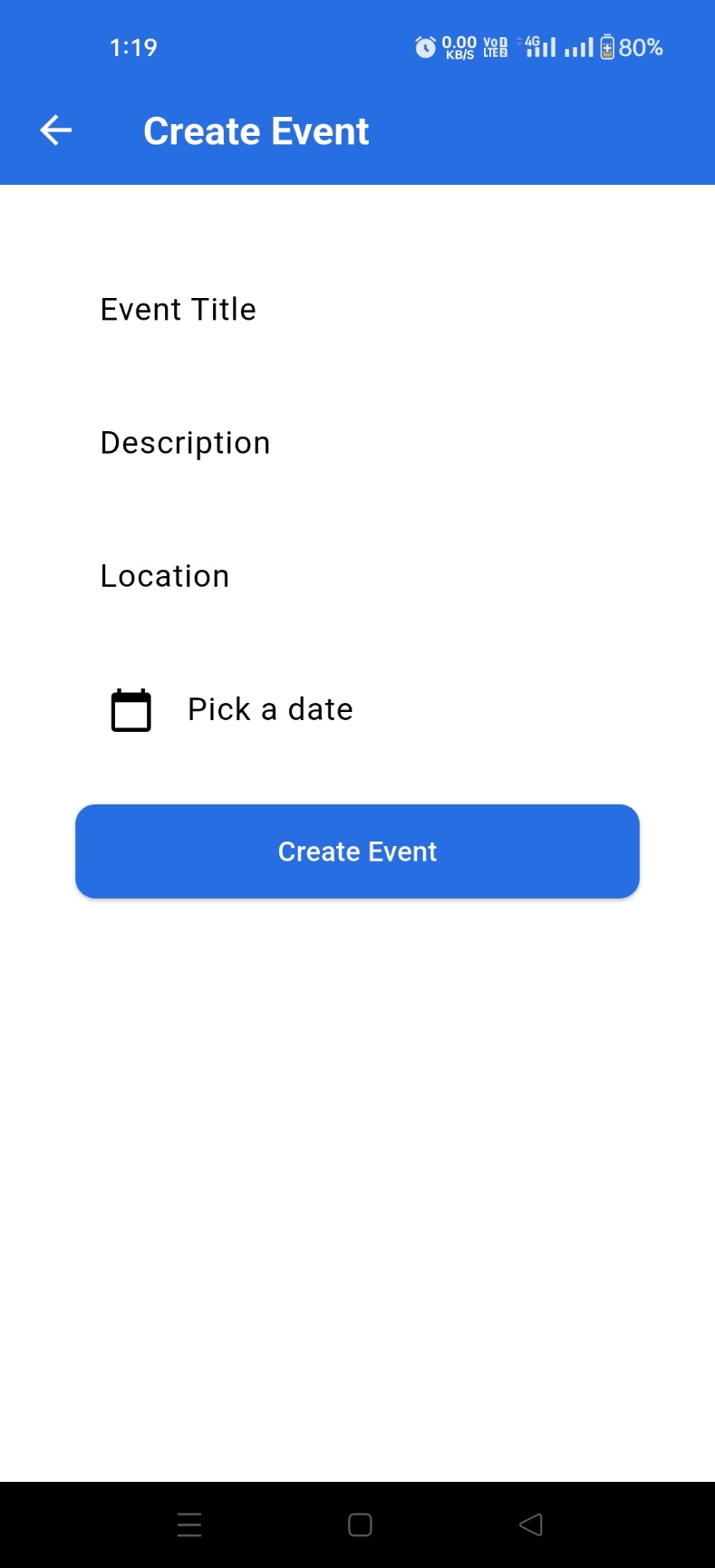
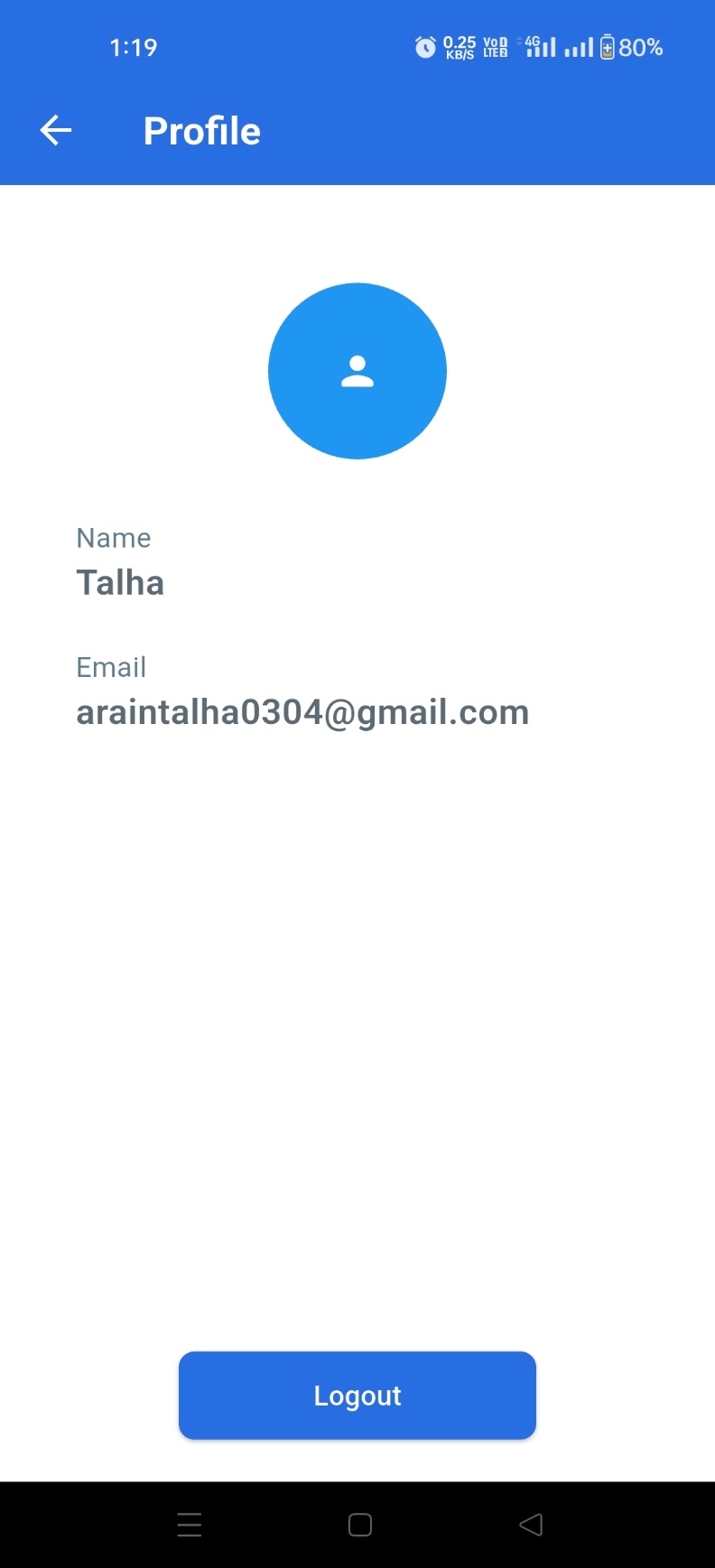
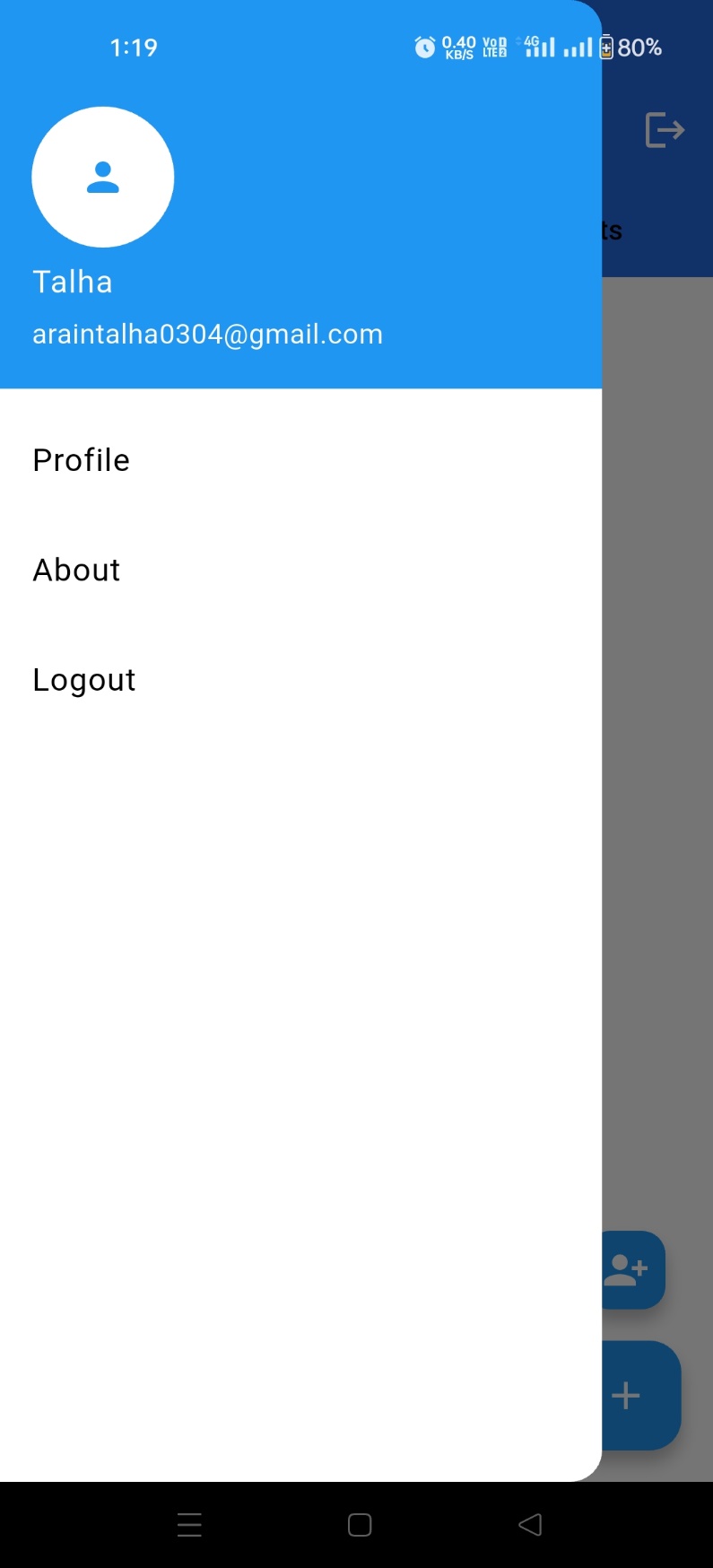
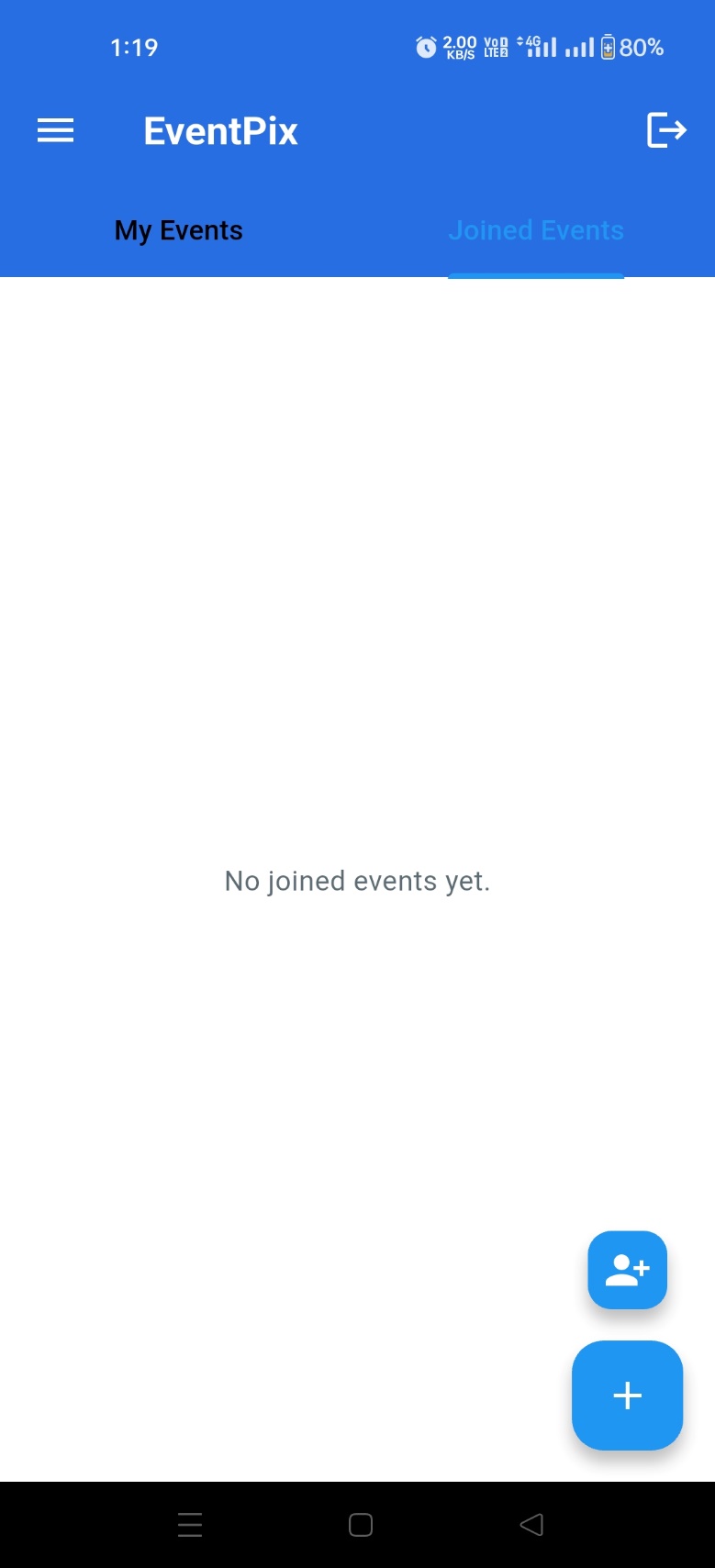
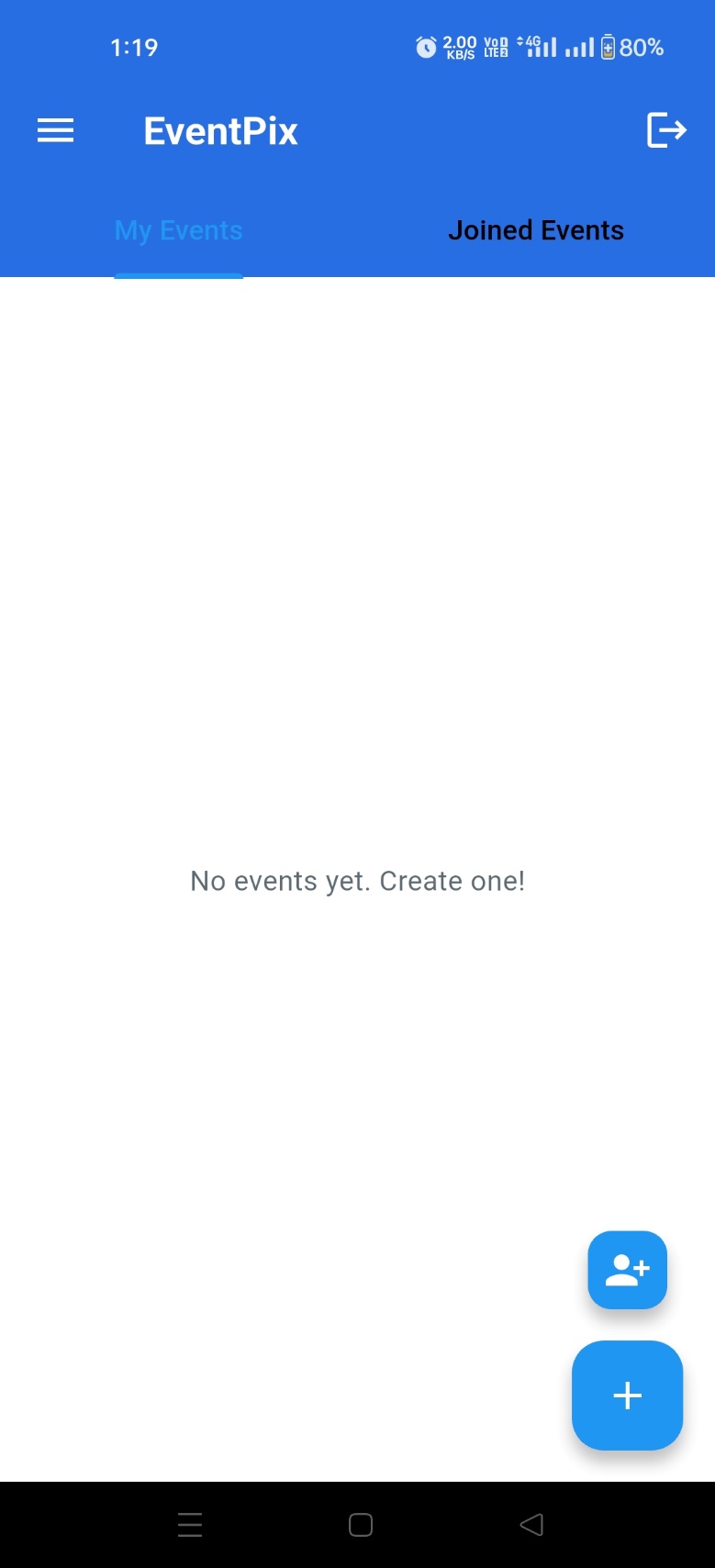
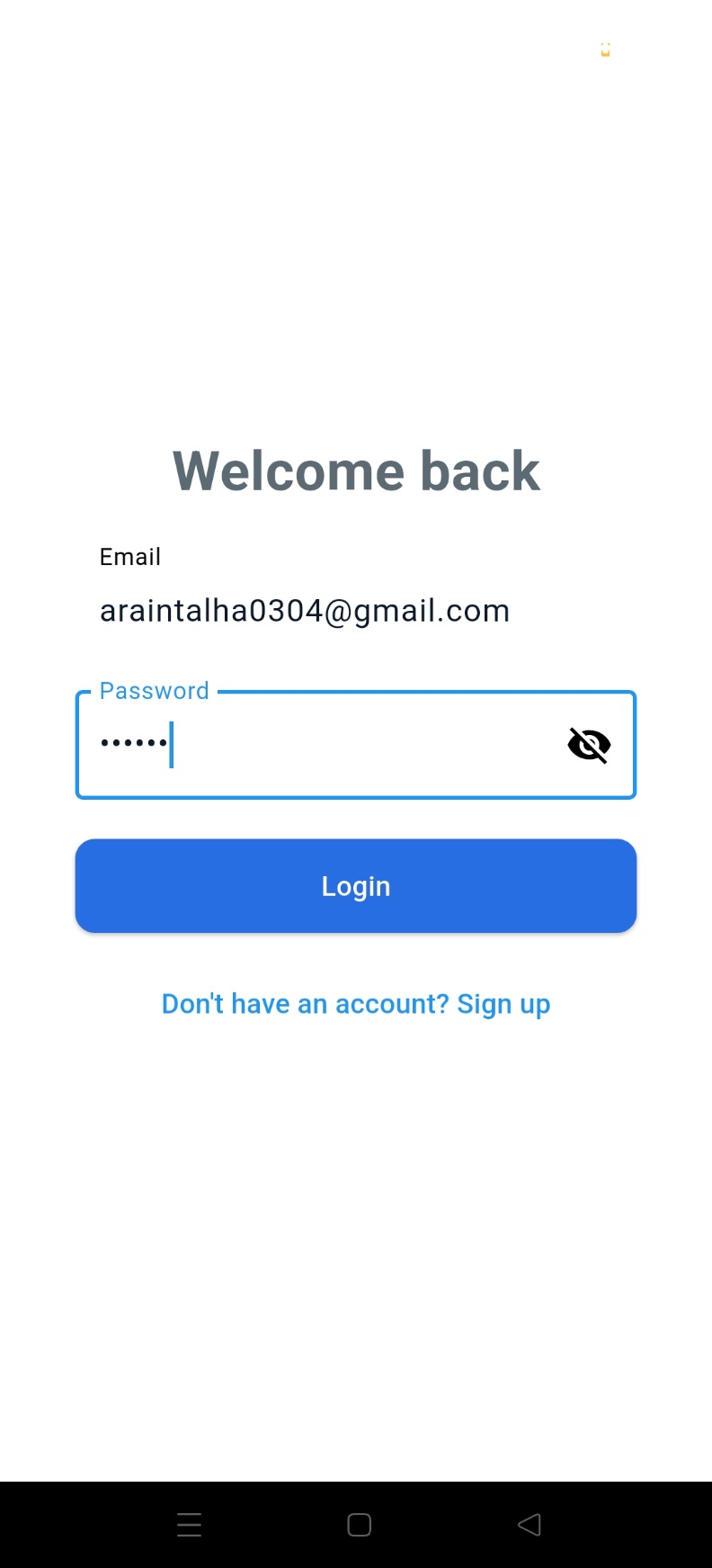
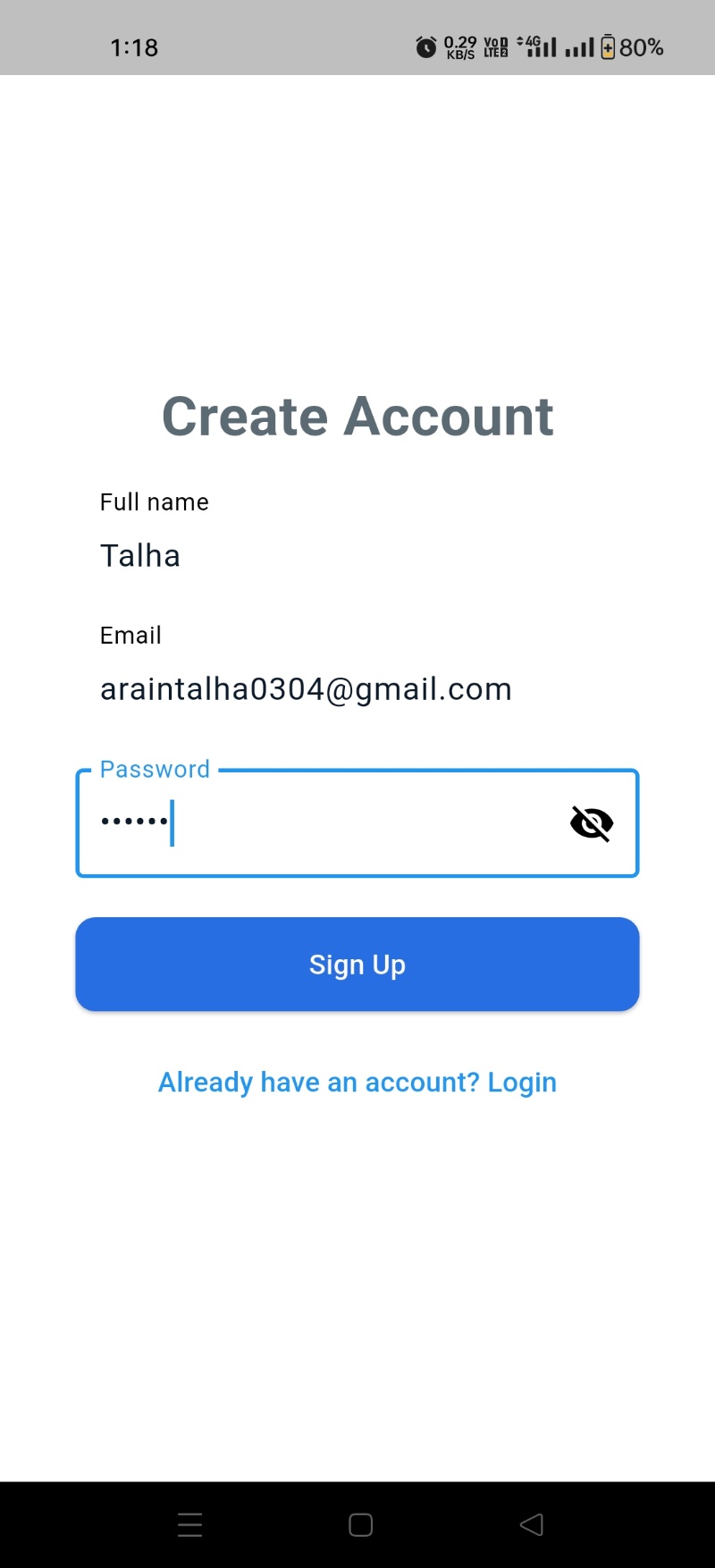
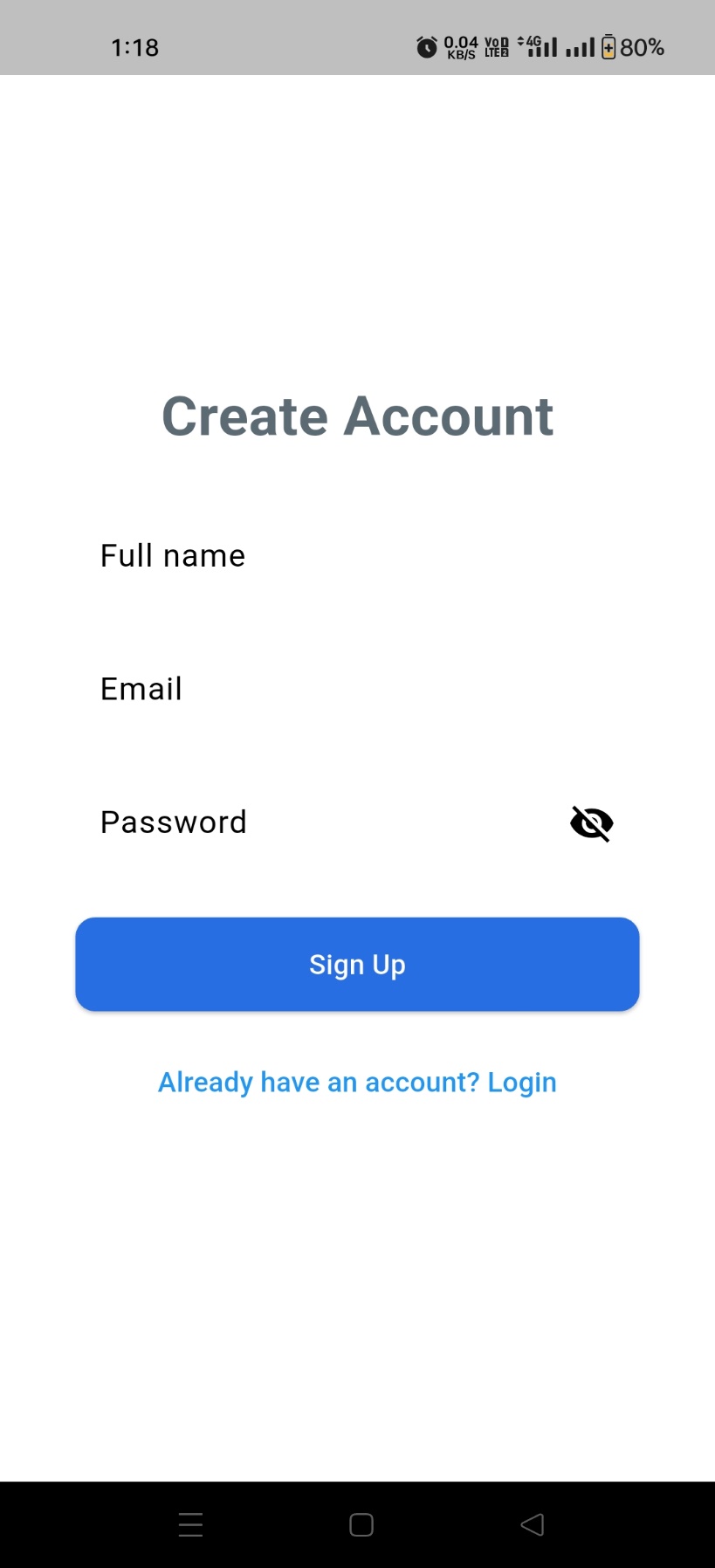
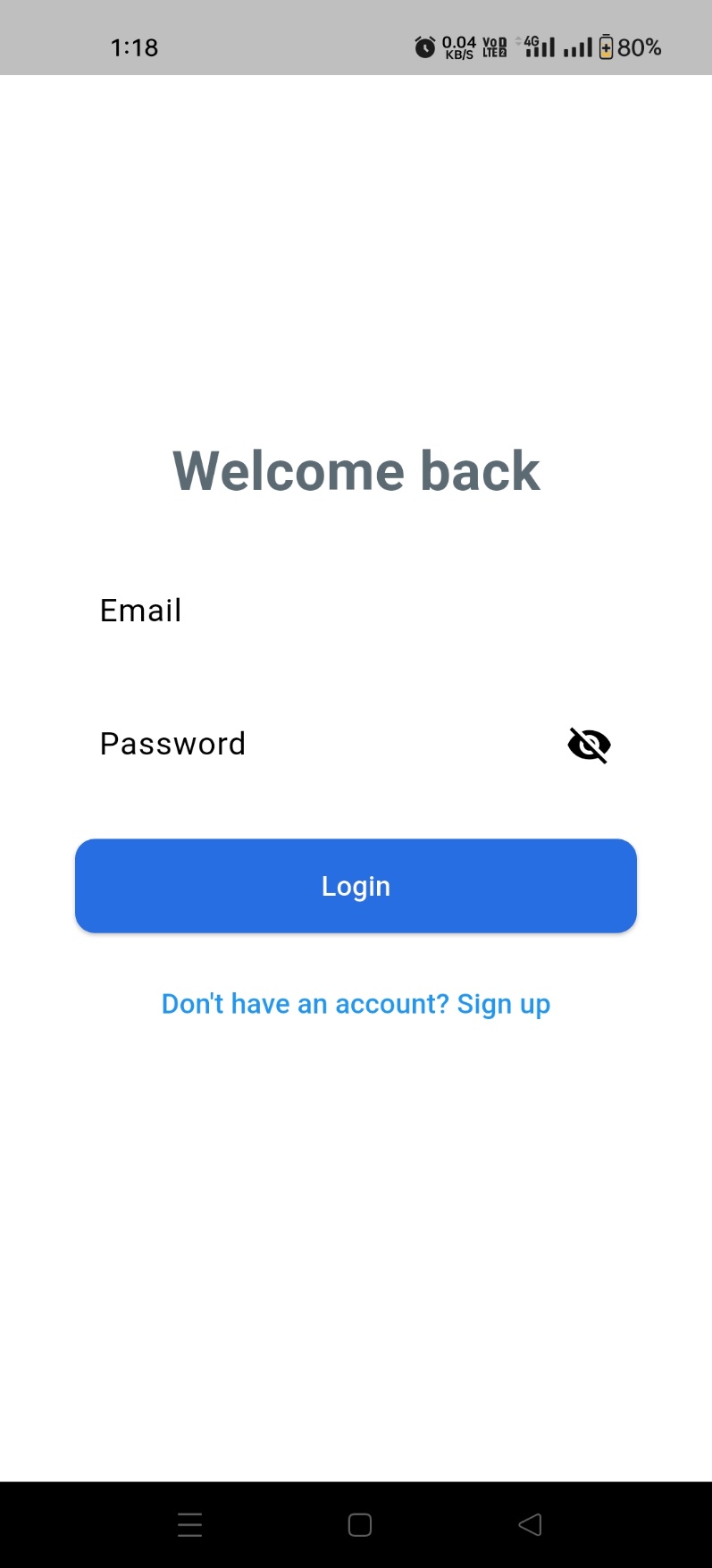
* **Cause:** Images were downloaded, but the user received no feedback or success message.
* **Solution:** We added Snackbar messages showing the progress and success of image downloads.

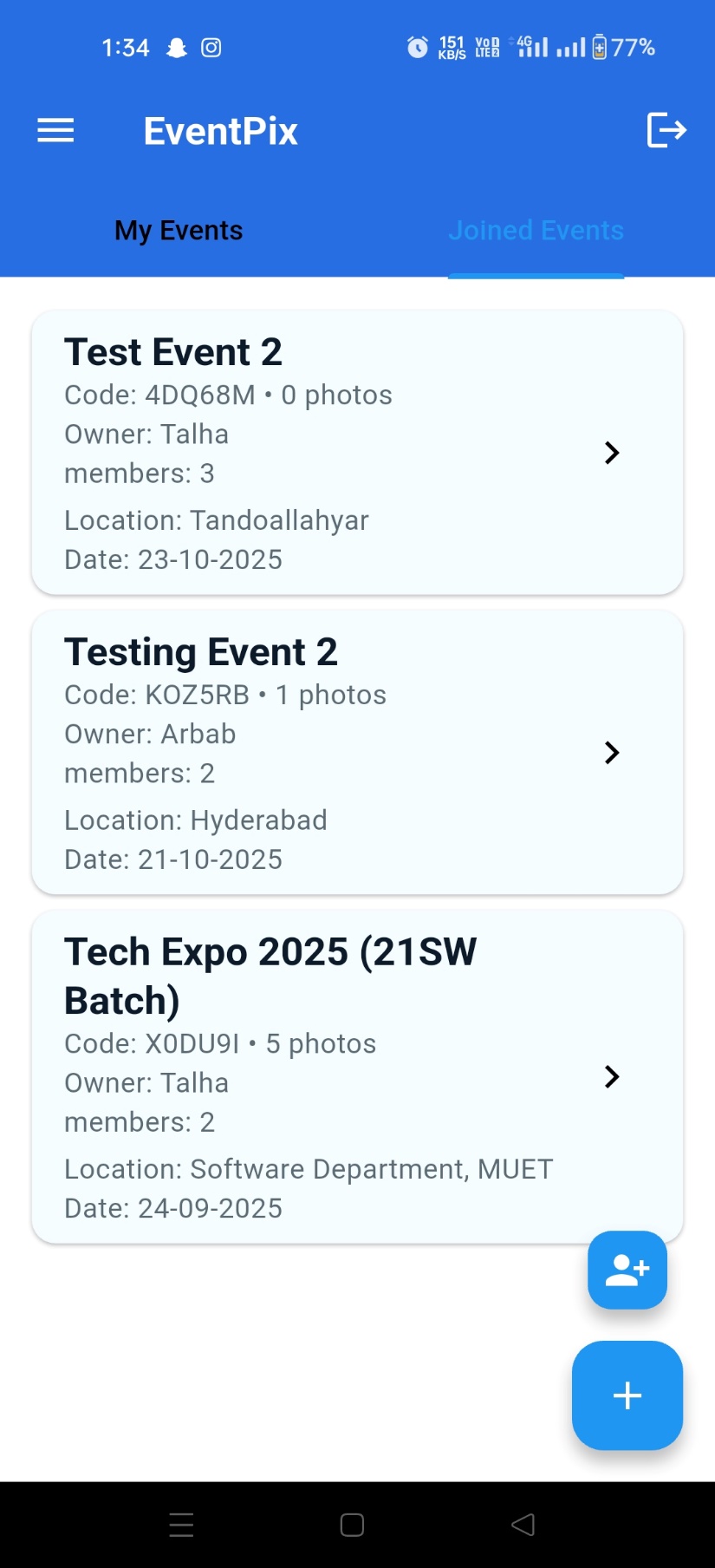
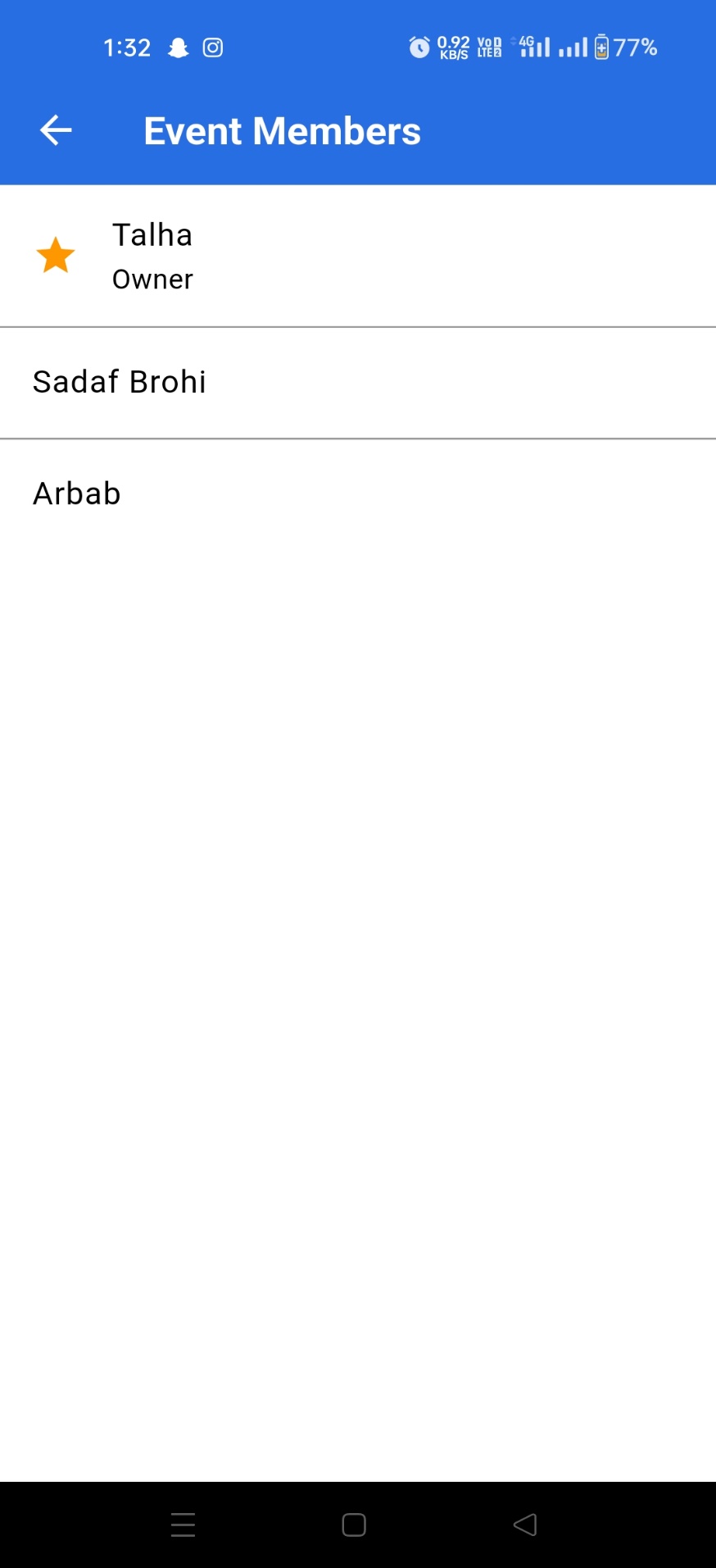
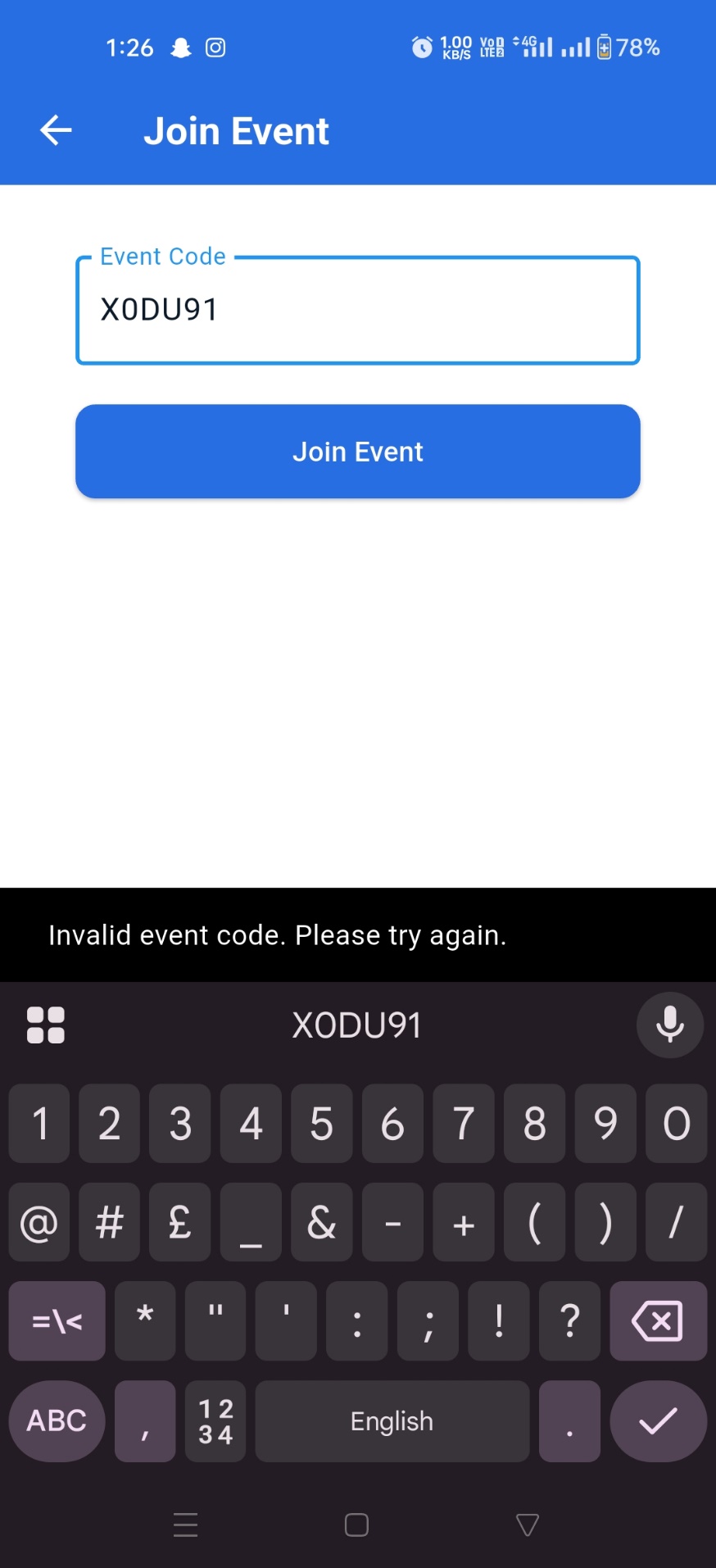
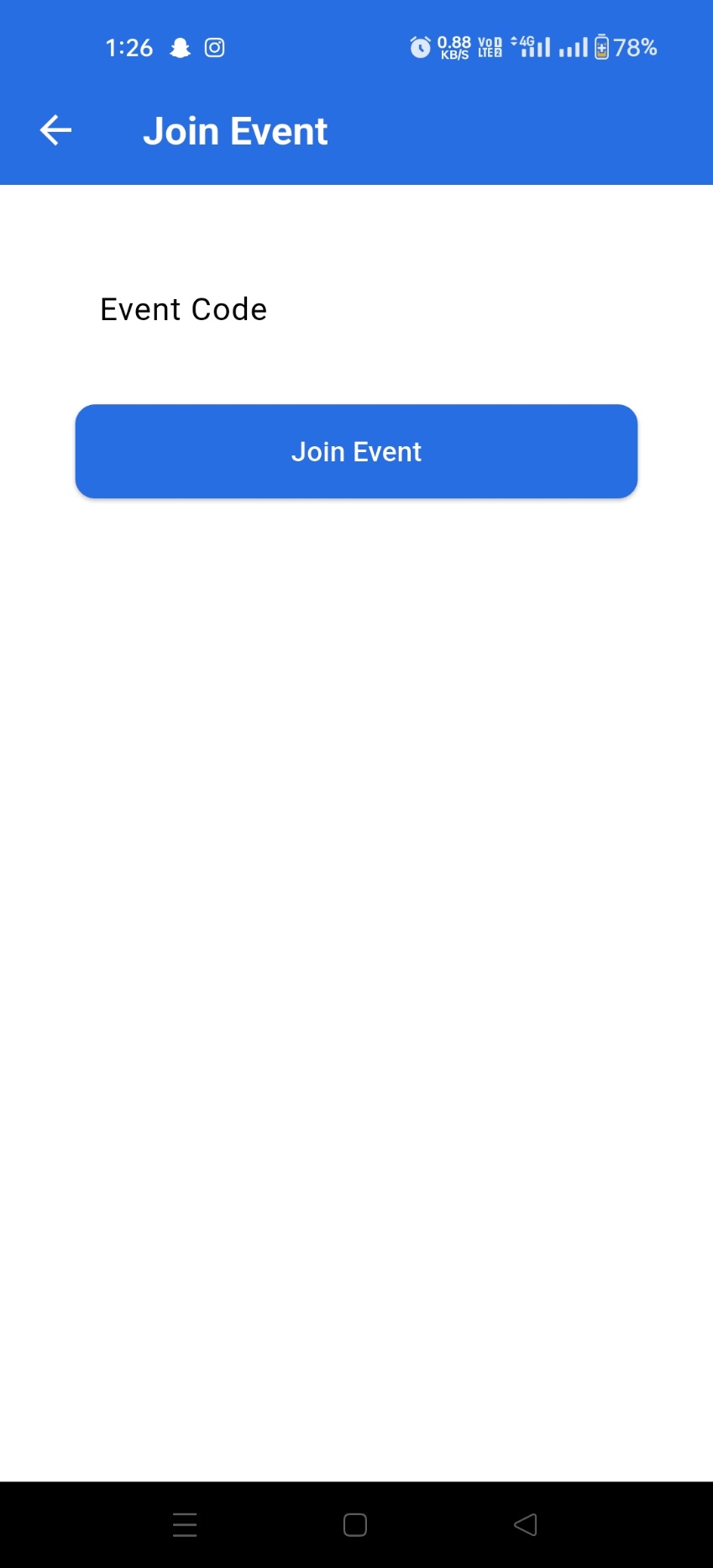
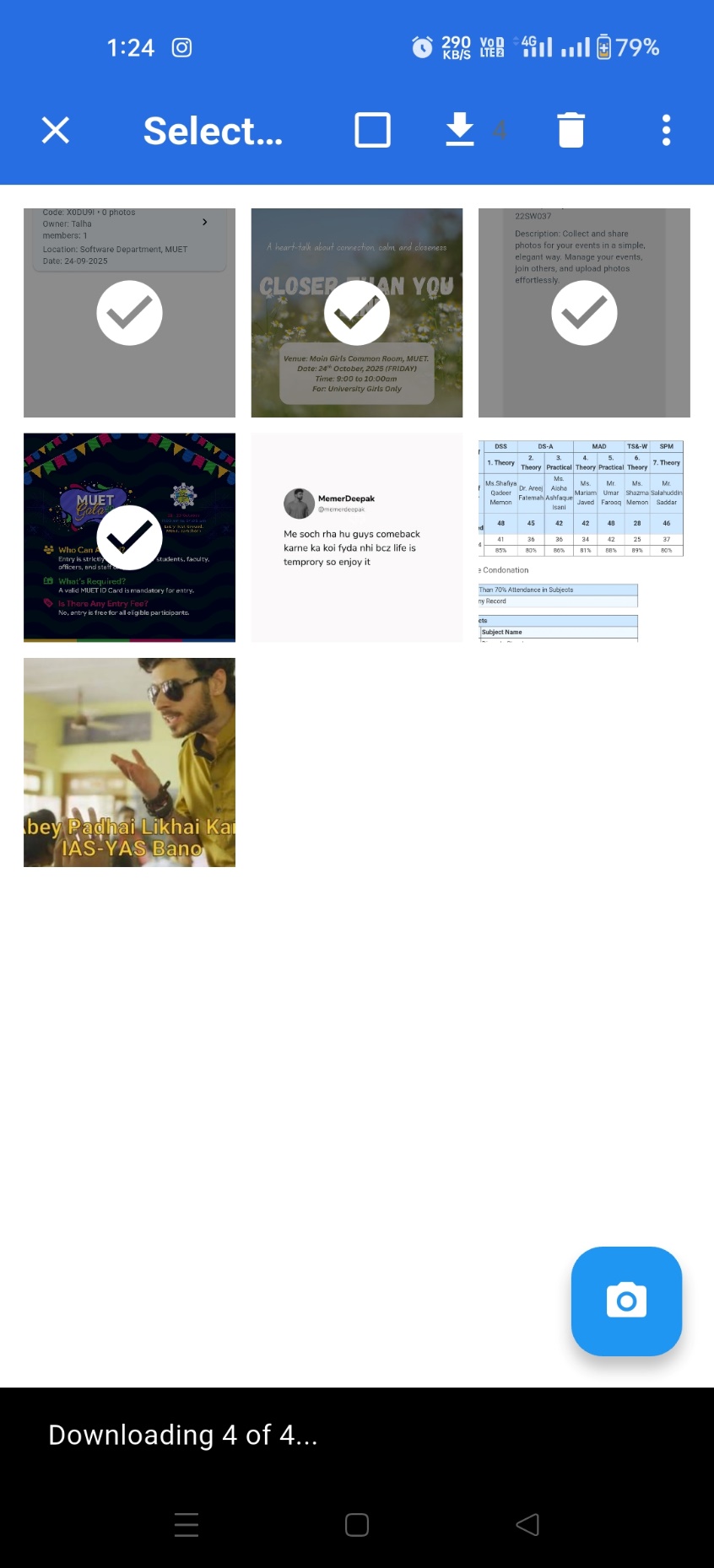
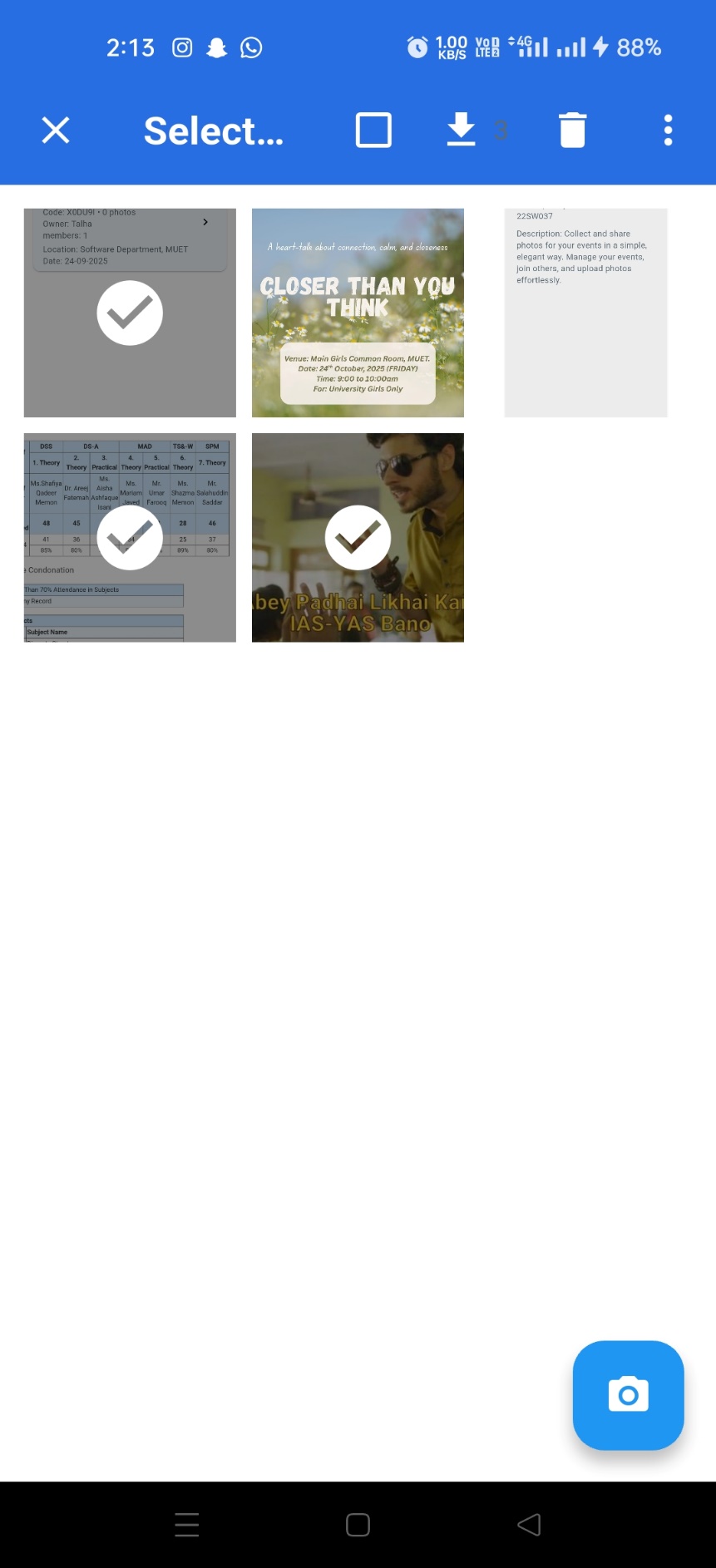
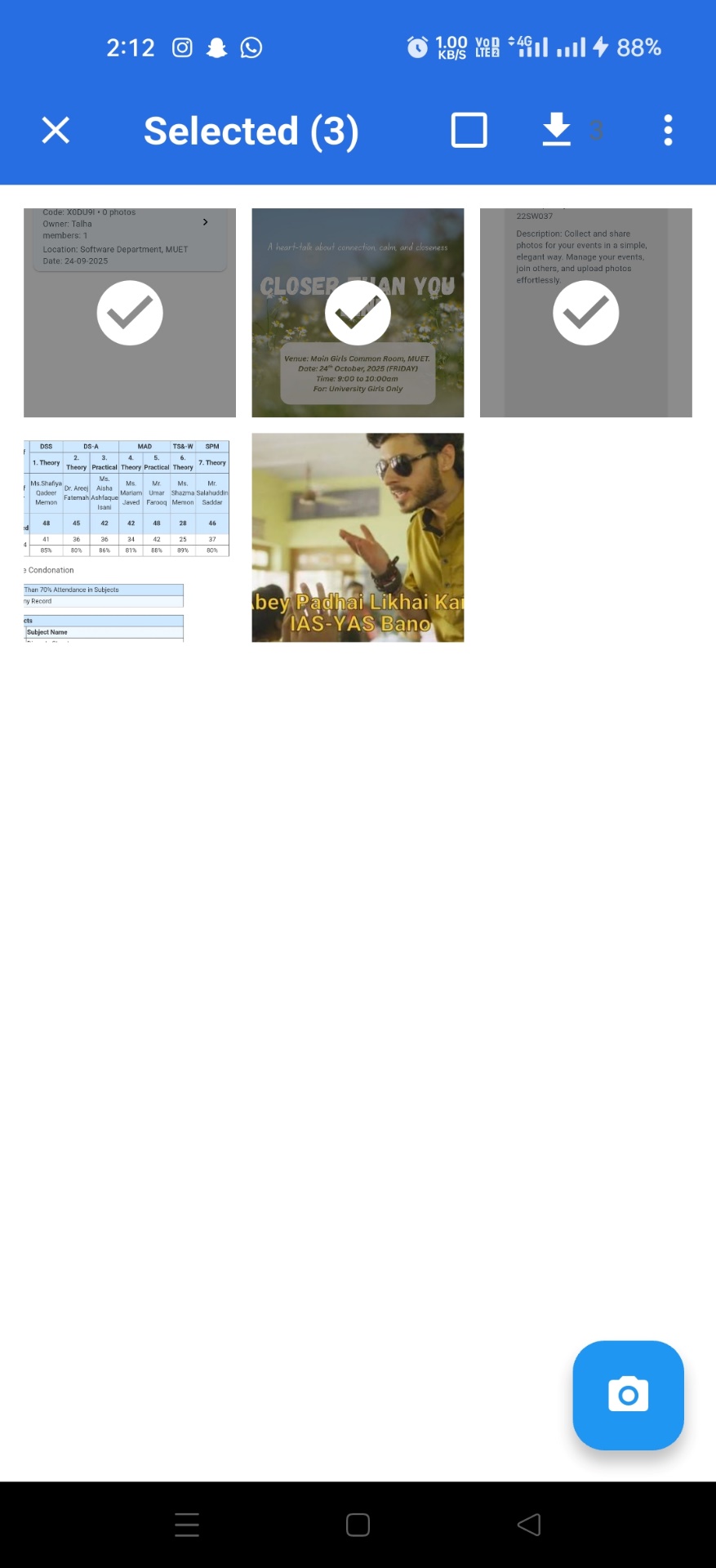
**6. Database Calls Without State Management**

* **Cause:** Initially, we were not using Provider, so database calls were made multiple times whenever user data such as user ID or name was needed.
* **Solution:** We implemented the Provider package to manage user state efficiently. This loads user data after login or signup and stores it in the Provider, reducing multiple database calls.

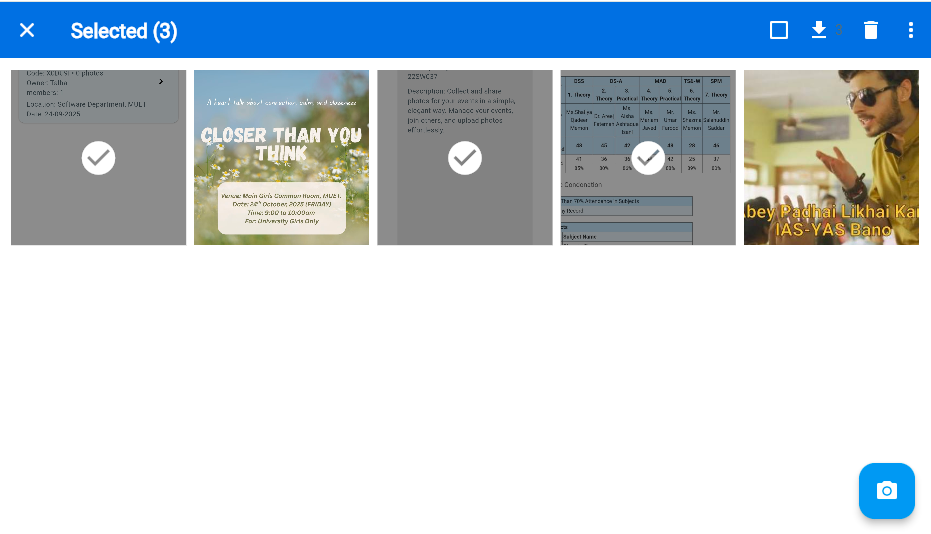
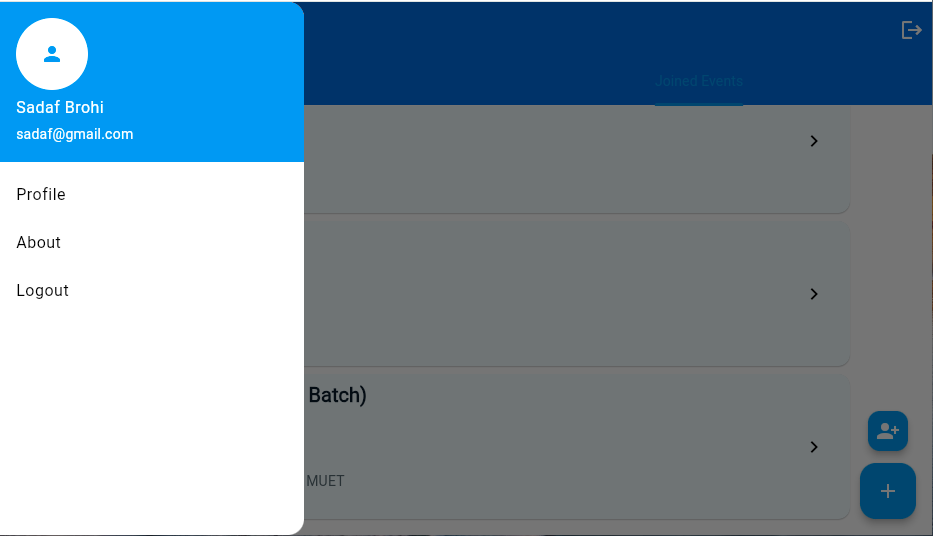
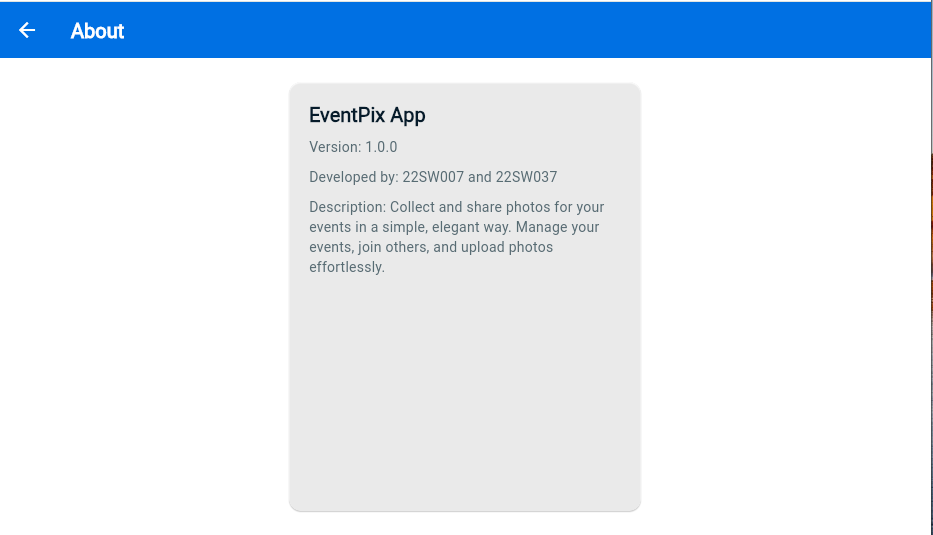
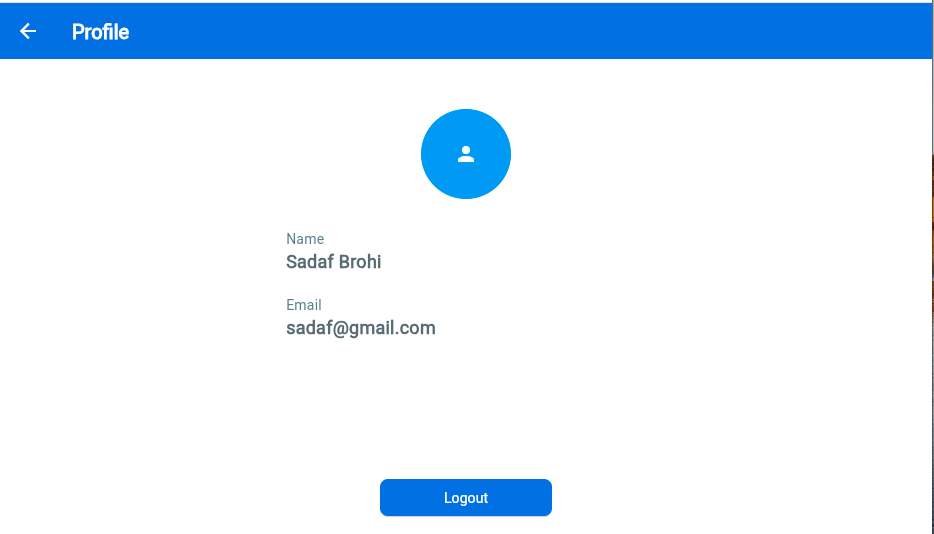
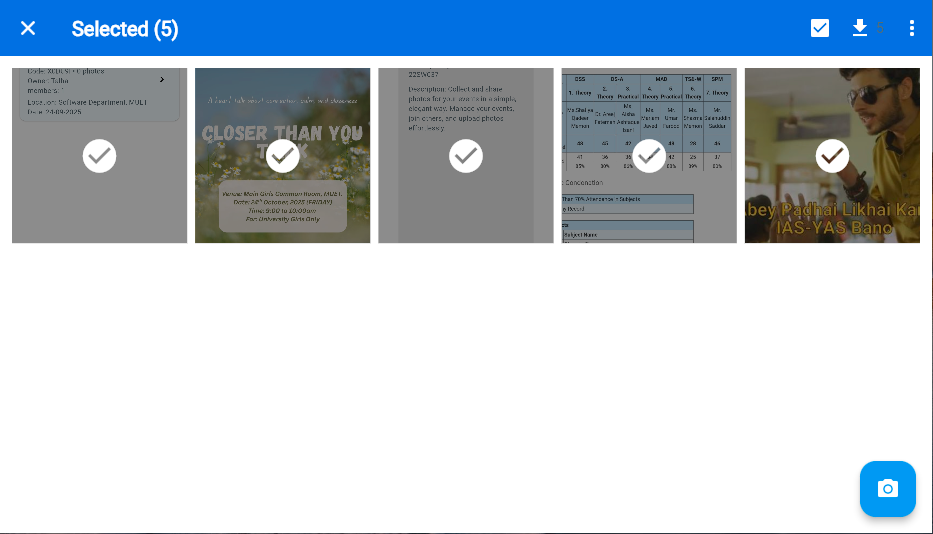
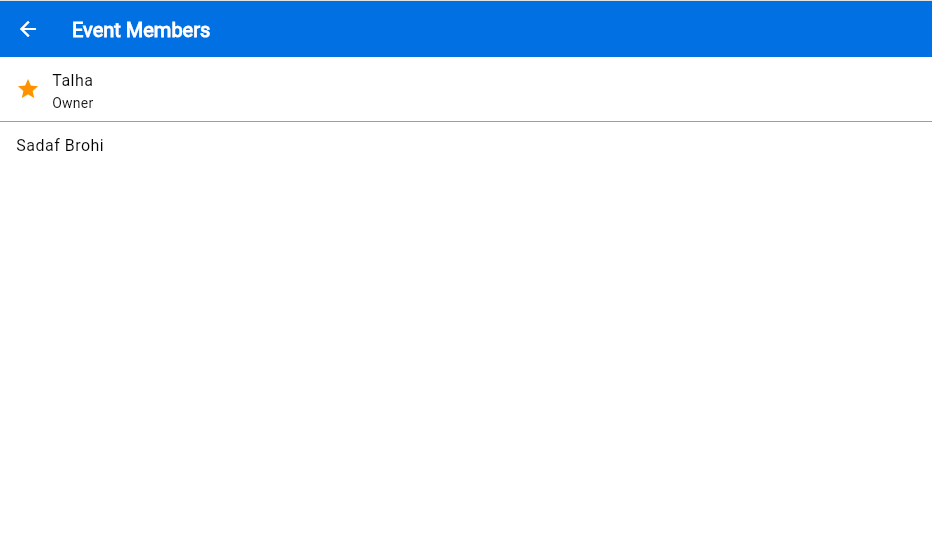
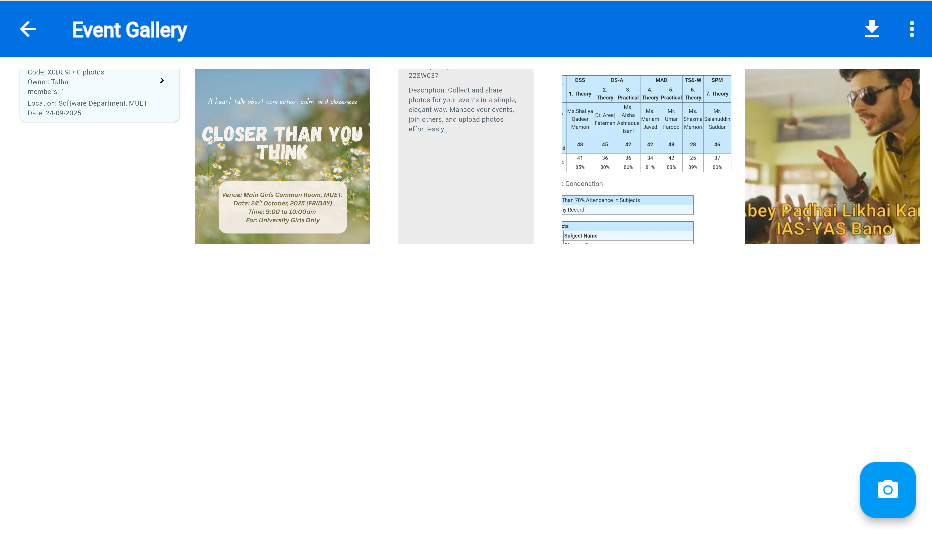
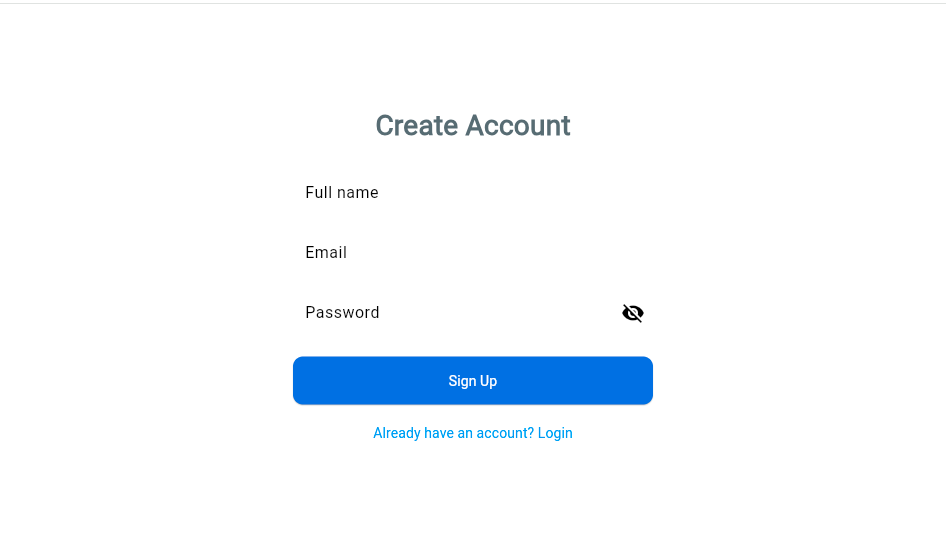
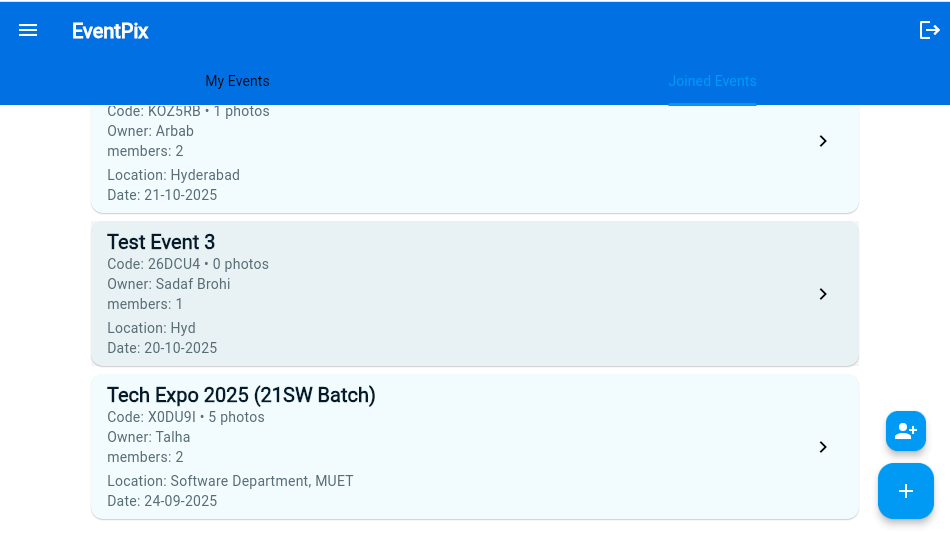
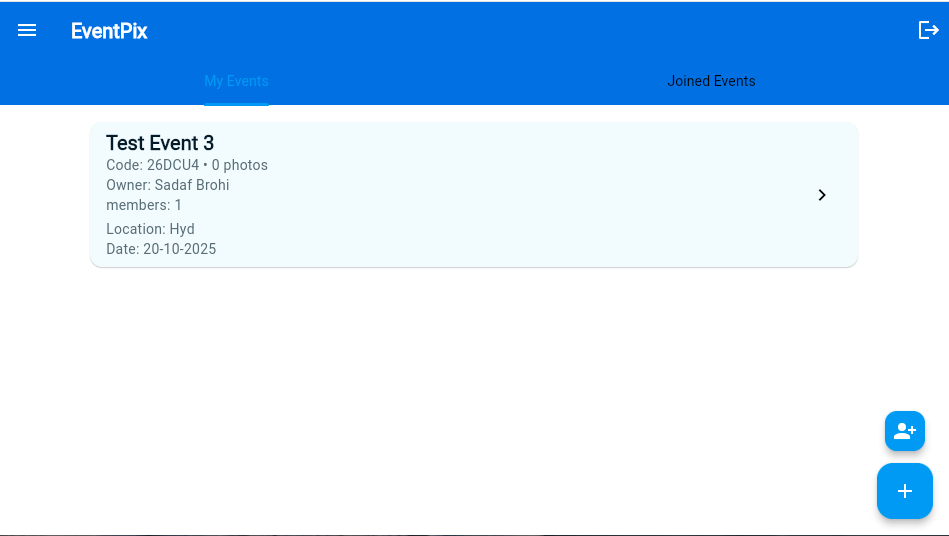
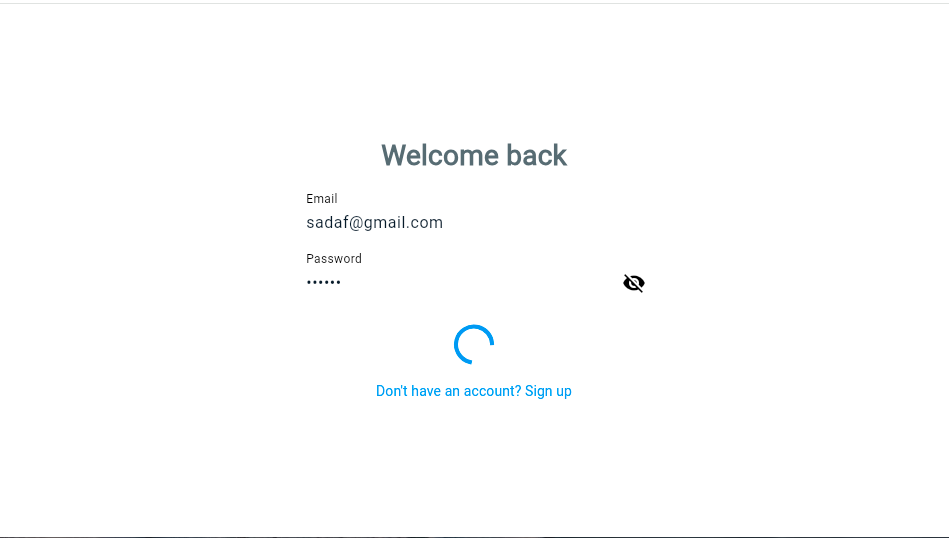
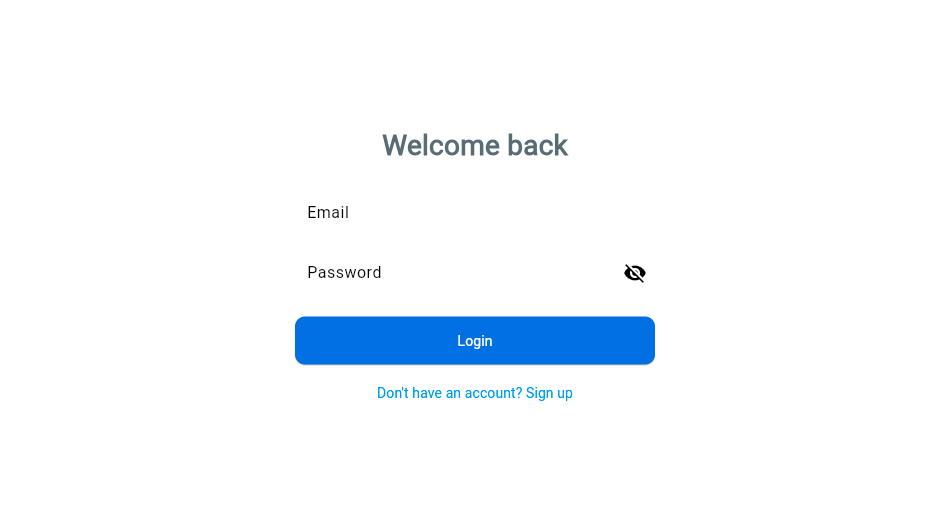
1. **Responsive User Interface:**

**Mobile Images:**

****

****

**Tablet Images:**



**Laptop Images:**

