Documentation

This document provides details on how the GeoMarker application interacts with the API, the challenges faced during development, and the solutions implemented.

API Interaction

The application communicates with a REST API to perform CRUD (Create, Read, Update, Delete) operations on geographic entities.

- **Base URL:** https://labs.anontech.info/cse489/t3/

Endpoints

- **GET /api.php:**
  - **Description:** Retrieves a list of all geographic entities.
  - **Implementation:** The EntityRepository fetches the entities from this endpoint. The data is then stored in the local Room database for offline access.
- **POST /api.php:**
  - **Description:** Creates a new entity.
  - **Implementation:** When a user saves a new entity in the EntityFormFragment, the EntityFormViewModel calls the repository to send a POST request with the entity's title, latitude, longitude, and image.
- **PUT /api.php:**
  - **Description:** Updates an existing entity.
  - **Implementation:** When a user edits and saves an entity, the EntityFormViewModel triggers a PUT request with the updated data.
- **DELETE /api.php:**
  - **Description:** Deletes an entity by its ID.
  - **Implementation:** The EntityListFragment allows users to delete entities, which triggers a DELETE request to the API.

Challenges and Solutions

1. Handling Deprecated Permission APIs

- **Challenge:** The traditional requestPermissions and onRequestPermissionsResult methods in Android are deprecated. Using them results in warnings and is not the recommended approach.
- **Solution:** I refactored the permission handling logic to use the modern ActivityResultLauncher API. This approach is cleaner, more maintainable, and aligns with current Android development best practices.

2. Offline Caching and Data Synchronization

- **Challenge:** The application needed to support offline viewing of entities. This required a mechanism to store data locally and keep it synchronized with the remote server.

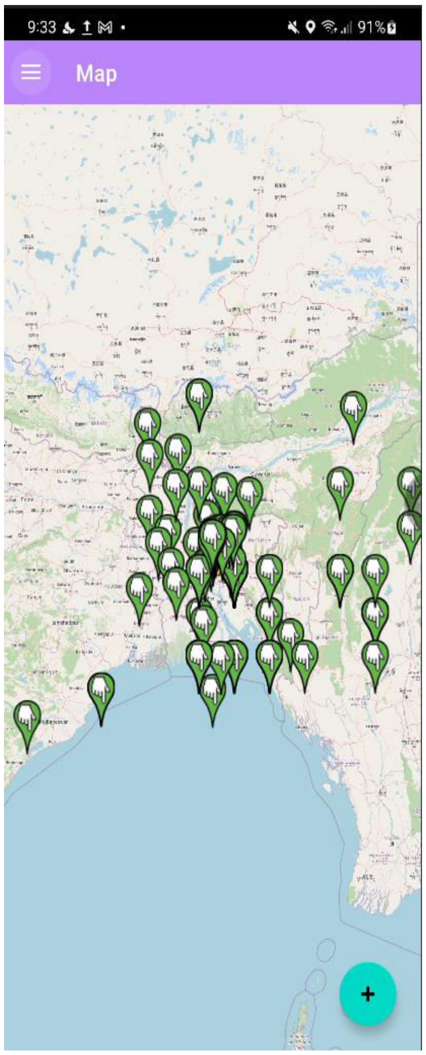
- **Solution:** I implemented a local database using the Room Persistence Library. The EntityRepository acts as a single source of truth, fetching data from the API and caching it in the Room database. The UI observes the local database for changes, ensuring a consistent and offline-capable experience.

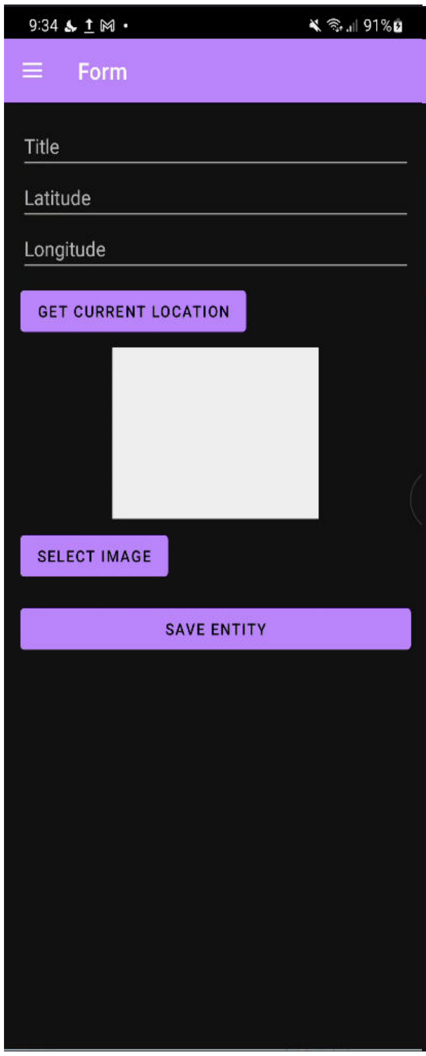
3. Image Handling

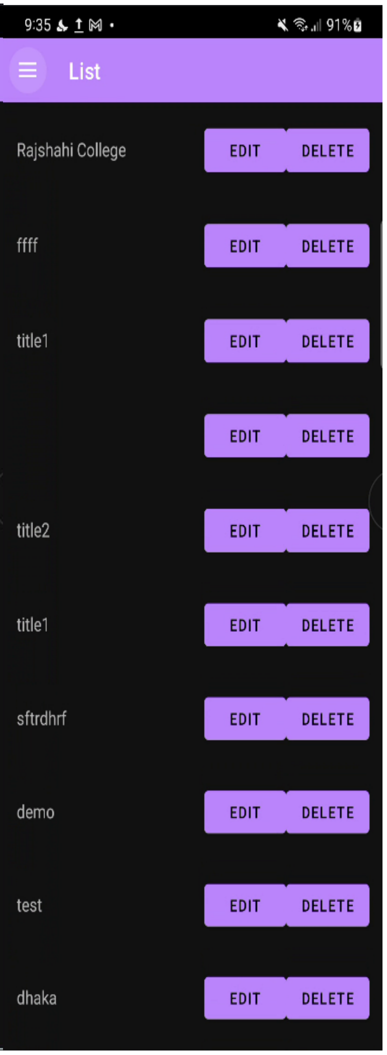
- **Challenge:** The application needed to handle image selection, previewing, and uploading.
- **Solution:** I used the ActivityResultLauncher for selecting images from the device's gallery. The Glide library is used for efficiently loading and displaying images from URLs and local URIs. For image uploads, the image is sent as a MultipartBody.Part in the Retrofit request.

Screenshots

I have also attached some screenshots of the screens of this project app which the user can interact with.







Rajshahi College	EDIT	DELETE
ffff	EDIT	DELETE
title1	EDIT	DELETE
	EDIT	DELETE
title2	EDIT	DELETE
title1	EDIT	DELETE
sftrdhrf	EDIT	DELETE
demo	EDIT	DELETE
test	EDIT	DELETE
dhaka	EDIT	DELETE