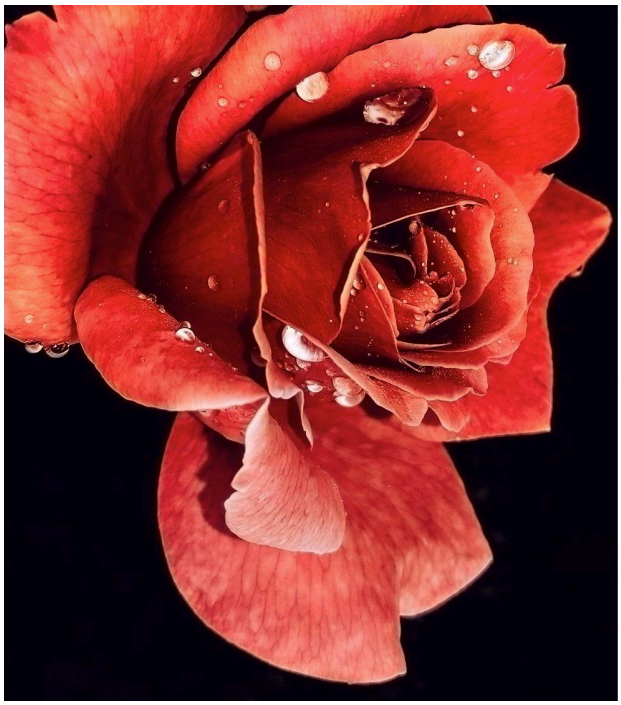
Mobile Computing

ScanApp



Introduction

This project aims to develop a mobile application that captures images using a smartphone's camera, sends them to a custom-designed API on a server, and receives processed images and display back to user.

1.Front Activity

- Purpose and Functionality: Acts as an initial loading activity, where a simulated heavy operation is executed using `AsyncThread` before navigating to `MainActivity`.

- Methods:

  - `onCreate()`: Initializes the activity layout and triggers the background task.

  - `startheavytask()`: Starts the background asynchronous task.

- `longop (AsyncTask Subclass)`:

- `doInBackground()`: Simulates a heavy operation by pausing the thread for 3.5 seconds in total.

- `onPostExecute()`: Transitions to `MainActivity` upon task completion.

2. Select Activity

Member Variables:

- Two buttons (`btngallery` and `btncamera`) are declared but not initialized immediately (`lateinit var`).

onCreate` Method:

- Lifecycle method called when the activity is starting.

- Sets the layout of the activity to `activity_select.xml` using `setContentView`.

- Initializes the `btngallery` and `btncamera` buttons by finding them in the layout using their IDs (`button2` and `button3`).

Event Listeners:

- Camera Button: Adds a click listener to `btncamera` that triggers an intent to start the `MainActivity`.

- Gallery Button: Adds a click listener to `btngallery` that triggers an intent to start the `gallery` activity.

7. Intent Handling:

- Each button click starts a new activity using `Intents`, which facilitate inter-component communication in Android apps.

8. Navigation:

- The app has simple navigation logic to either the main activity(camera based) or a gallery view based on user interaction.

3. Main Activity (`MainActivity` Class)

- User Interface Components:
  - Button for image selection by capturing from the camera.
  - ImageViews to display the selected and processed image.
- Functionality:
  - Permission Management: Checks and requests camera permissions.
  - Image Handling:
    - Camera Access: Opens a camera intent for capturing images.
  - Image Processing (`preprocessImage()`):
    - Executes in a background thread using Kotlin coroutines.
    - Encodes and sends the image to a server via a Retrofit-configured API, then displays the processed image.

4. Gallery Activity

- User Interface Components:
  - Button for image selection from Gallery.
  - ImageViews to display the selected and processed image.
- Functionality:
  - Permission Management: Checks and requests Storage permissions.
  - Image Handling:

- Gallery Access: Opens a gallery intent for image selection.

- Image Processing (`preprocessImage()`):

- Executes in a background thread using Kotlin coroutines.

- Encodes and sends the image to a server via a Retrofit-configured API, then displays the processed image.

5. Network Communication Setup (`ApiService` Interface)

- **API Definition:**

- preprocessImage(): Defined to post an image as multipart/form-data to the server.

- Retrofit Configuration:

- Utilizes `HttpLoggingInterceptor` for detailed logging of network requests and responses.

- Builds the Retrofit instance with Gson for JSON conversion and a base URL pointing to the intended server.

- Retrofit and OkHttpClient Configuration:

- A logging interceptor is included to capture detailed logs of all network traffic, which aids in debugging.

- Constructs and configures the Retrofit service with the necessary converter factory and client settings.

- Backend Used:-

For server side programming we have created a Python-Django based API backend and it is performing below functionality:-

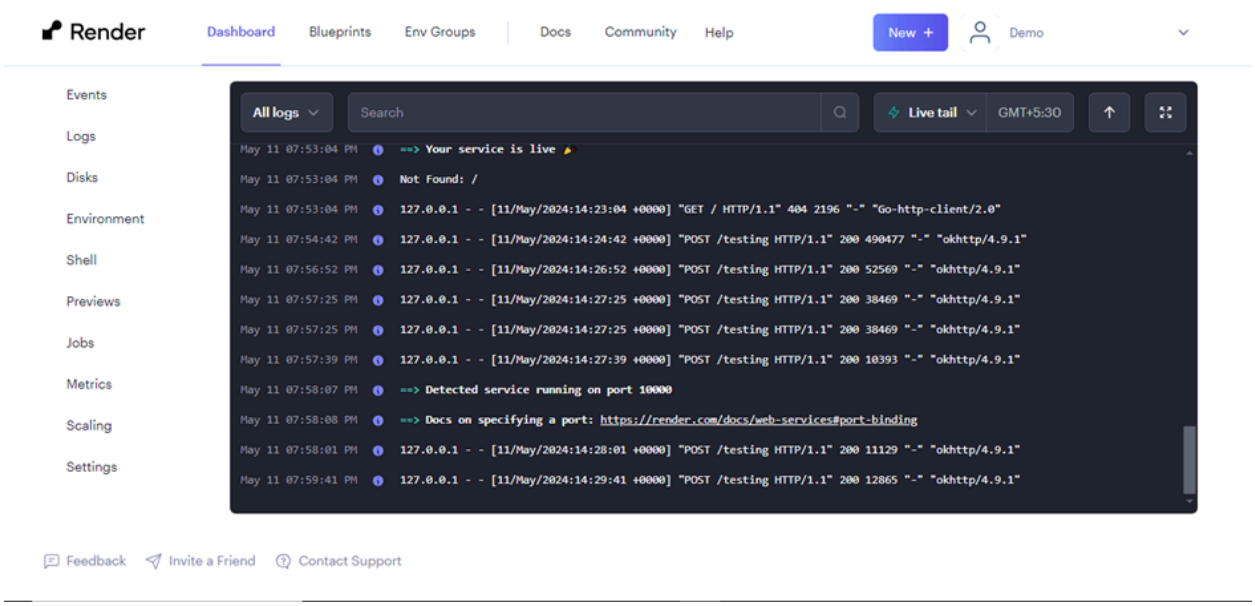
For request mapping we have created request handling end point as name “testing”.

This Django view function preprocess_image is designed to handle the preprocessing of uploaded images.

It accepts image files via a POST request and preprocesses them by converting them to black and white using the Python Imaging Library (PIL).

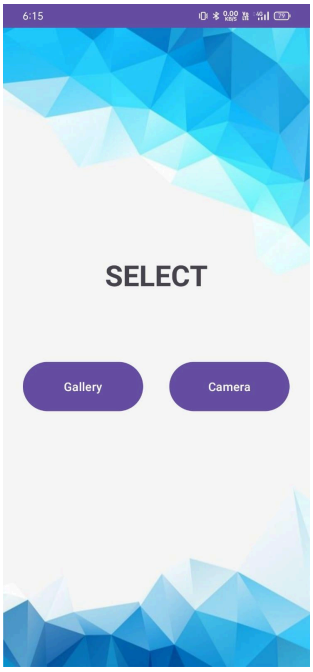
The Preprocessed image is then encoded as a base64 string and returned as a JSON response containing the processed image URL.

And this deployed to free web service provided environment for easily accessing and for functioning with help of Github.

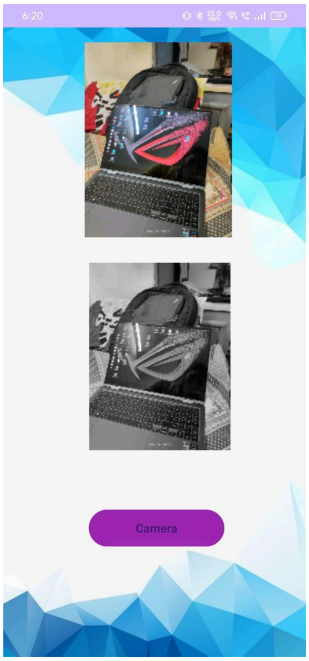


OUTPUT:


QUICK SCAN
Designed By :
Piyush Kumar Saini
Manish KUmar


SELECT
Gallery
Camera


select image


Camera