**Assignment-1a(Image search app)**

**Github link:**

https://github.com/21341A05F1-RNareshReddy/-Team-Charmers-EH-image\_search\_app-

**Frontend (React-Native CLI)**

**Overview**

The front-end application is built with React-Native CLI, providing a mobile interface for searching and viewing images.

**Project Structure**

* **App.js**: Entry point for the React Native application that sets up navigation.
* **components/**: Contains reusable UI components.
  + **ImageCard.js**: Displays a thumbnail of an image with basic details.
  + **ImageDetailScreen.js**: Shows a detailed view of a selected image.
  + **SearchScreen.js**: Provides search functionality and lists image results.
* **navigation/**: Manages navigation between different screens.
  + **AppNavigator.js**: Defines navigation routes and stack.

**Setup and Configuration**

1. **Install Node.js**:
   * Ensure Node.js and npm (or Yarn) are installed on your machine.
2. **Initialize a React Native Project**:
   * Create a new React Native project using the React Native CLI.
3. **Install Dependencies**:
   * Install required libraries such as React Navigation and Axios.
4. **Configure React Navigation**:
   * Set up navigation according to the React Navigation documentation.
5. **Run the Application**:
   * Start the application on Android or iOS using the React Native CLI commands.

**Component Overview**

1. **App.js**
   * **Purpose**: Entry point for the React Native application that handles navigation setup.
2. **ImageCard.js**
   * **Purpose**: Displays a thumbnail image and basic information.
   * **Props**:
     + image: Data object containing image details.
     + onPress: Function to handle image selection.
3. **ImageDetailScreen.js**
   * **Purpose**: Shows detailed information about a selected image.
   * **Props**:
     + route: Contains the image data passed from the previous screen.
4. **SearchScreen.js**
   * **Purpose**: Provides the interface for searching images and displays the results.
   * **State**:
     + query: Search term input by the user.
     + images: List of images retrieved from the backend.
   * **Methods**:
     + searchImages(): Sends a request to the backend and updates the image list.
5. **AppNavigator.js**
   * **Purpose**: Defines navigation routes and stack for the application.
   * **Components**:
     + SearchScreen: Main screen for searching images.
     + ImageDetailScreen: Screen for viewing detailed image information.

**Backend (Flask with MySQL)**

**Overview**

The backend service uses Flask and MySQL to offer an API for searching images from an external service and storing image metadata in a MySQL database.

**Project Structure**

* **app.py**: Main application file that contains Flask routes and business logic.
* **config.py**: Configuration file for database connection settings.
* **models.py**: Defines the SQLAlchemy models for the database schema.
* **requirements.txt**: List of required Python packages for the backend.

**Setup and Configuration**

1. **Install MySQL**:
   * Ensure MySQL server is installed and running.
   * Create a new database for your project.
2. **Create a Virtual Environment**:
   * Set up a virtual environment to manage dependencies.
3. **Install Required Packages**:
   * Use pip to install the packages listed in requirements.txt.
4. **Configure Database**:
   * Update config.py with your MySQL database connection details such as username, password, and database name.
5. **Initialize the Database**:
   * Run initialization commands to create the necessary tables in the MySQL database.

**API Endpoints**

1. **Search Images**
   * **URL**: /search
   * **Method**: GET
   * **Query Parameters**:
     + query: Search term for images.
     + orientation: Optional filter for image orientation.
     + category: Optional filter for image category.
   * **Response**: Returns a list of images with metadata.
   * **Error Handling**: Handles missing or incorrect parameters by returning appropriate error messages.
2. **Get Image Details**
   * **URL**: /images/<id>
   * **Method**: GET
   * **URL Parameters**:
     + id: ID of the image to retrieve.
   * **Response**: Provides detailed information about the requested image.
   * **Error Handling**: Returns a 404 error if the image with the specified ID is not found.

**Code Description**

* **app.py**: Contains the core logic of the Flask application, including handling image search requests and storing data in MySQL.
* **config.py**: Manages configuration settings for Flask and SQLAlchemy.
* **models.py**: Defines the database schema for storing image metadata using SQLAlchemy.