Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

PROJECT AND TEAM INFORMATION

TO-DO LIST WITH MOBILE NOTIFICATION

Student Information	Team Roles	SuperCoders
Team member 1 (Team Lead) Name: Lakshya Singhal University Roll no.: 2418617 Student ID: 24011865 Email: lsinghal.graphichill@gmail.com Course: BTECH COMPUTER SCIENCE ENGINEERING (CSE)	**Backend Developer** Design and implement server-side components, including user authentication and login management. **Tech Stack:** Language: Python Framework: Kivy/Flask (for Backend) Database: SQLite (local storage)	
Team member 2 Name: Priyanshu Sharma University Roll no.: 2418791 Student ID: 240111054 Email: c137sharma666@gmail.com Course: BTECH COMPUTER SCIENCE ENGINEERING (CSE)	**File Sync & Notification System Developer** Develop notification scheduling and delivery systems for both mobile and desktop clients. **Tech Stack:** Language: Python Framework: Kivy Libraries: Plyer (for notifications)	
Team member 3 Name: Antriksh Rana University Roll no.: 2418308 Student ID: 240111774 Email: antrikshrana187@gmail.com Course: BTECH COMPUTER SCIENCE ENGINEERING (CSE)	**Desktop Application Developer** Develop the desktop client with a user-friendly UI for task management and login. **Tech Stack:** Language: Python Framework: Kivy Libraries: Plyer (for desktop notifications)	
Team member 4 Name: Akhil Padiyar University Roll no.: 2418205 Student ID: 24011840 Email: akhilpadiyar2@gmail.com Course: BTECH COMPUTER SCIENCE ENGINEERING (CSE)	**Mobile Application Developer** Develop the mobile application with an optimized UI for task management and login. **Tech Stack:** Language: Python Framework: Kivy / KivyMD Libraries: Plyer (for Android notifications)	

PROJECT PROGRESS DESCRIPTION

Project Abstract

In an era of increasing digital distraction, effective task management is crucial for personal and professional productivity. This project presents the design and implementation of a cross-platform To-Do List application developed using Python and the KivyMD framework. The application addresses the limitations of traditional, local-only task managers by implementing a client-server architecture that ensures real-time data synchronization across multiple devices within a local network.

The system consists of two main components: a client-side Android application and a centralized Flask-based server. The client application features a modern, Material Design user interface that allows users to create, edit, delete, and track the completion status of tasks. It utilizes a local SQLite database for session management and relies on RESTful API calls (GET, POST) via the `requests` library to communicate with the central server.

Updated Project Approach and Architecture

Approach: The project initially utilized a monolithic architecture with a local SQLite database stored directly on the device. However, to achieve the goal of multi-device synchronization, the approach was shifted to a **Client-Server Architecture**.

- **Client (Frontend):** Developed using Python and **KivyMD 1.2.0**. It handles the user interface, input validation, and local session management. Instead of querying a local database for tasks, it now uses the `requests` library to communicate with the backend.
- **Server (Backend):** A lightweight **Flask** web server acts as the centralized API. It manages the "master" SQLite database, handles authentication (hashing passwords), and processes CRUD (Create, Read, Update, Delete) requests from client devices.
- **Communication:** The client and server exchange data via RESTful API endpoints (GET and POST requests) over the local Wi-Fi network.

Tasks Completed

Task Completed	Team Member
User Interface (UI/UX): • Implemented a Material Design interface using KivyMD. • Created distinct screens for Login, Signup, and the Main To-Do list. • Designed a custom "Add Task" dialog with integrated Date and Time pickers.	• Antriksh Rana
Backend & Logic: • **Authentication:** Built a secure login/signup system using SHA-256 hashing to protect user passwords on the server. • **Data Management:** Implemented full Create, Read, Update, and Delete (CRUD) functionality via API.	• Lakshya Singhal
• **Synchronization:** Successfully established real-time data syncing. Tasks added on one device appear on others upon reload. • **Notifications:** Integrated `plyer` and Android permissions (`POST_NOTIFICATIONS`) to alert users when tasks are due, synchronized with the server status.	• Priyanshu Sharma
Deployment: • Configured `buildozer.spec` with necessary permissions (`INTERNET`, `VIBRATE`). • Successfully compiled the Python code into an Android APK (`.apk`).	• Akhil Padiyar

Challenges/Roadblocks

During development, several technical hurdles were encountered and resolved:

- **Kivy/KV Initialization Timing:**
  - *Issue:* The app crashed with `KeyError` when trying to access widget IDs (like `date_button`) inside the Python `__init__` method before the KV file was fully loaded.
  - *Resolution:* Restructured the `build()` method to load the KV string first and utilized `on_release` events within the KV file to handle button logic.
- **Sync Strategy:**
  - *Issue:* Initially considered syncing the physical `.db` file via FTP, which poses risks of data loss and file corruption (locking errors).
  - *Resolution:* Migrated to a REST API architecture using Flask, ensuring atomic database transactions and preventing race conditions.
- **Android Permissions:**
  - *Issue:* The app crashed on Android when attempting to access the network or send notifications.
  - *Resolution:* Updated `buildozer.spec` to explicitly request `INTERNET` and `POST_NOTIFICATIONS` permissions and added runtime permission requests in the Python code.
- **UI Styling:**
  - *Issue:* Default KivyMD elevation created unappealing heavy shadows on cards and bars.
  - *Resolution:* Manually set `elevation: 0` on `MDCard`, `MDTopAppBar`, and `MDFloatingActionButton` for a cleaner design.

Tasks Pending

Task Pending
• **None.** (All core features and sync requirements have been implemented and verified).

Project Outcome/Deliverables

The project has successfully produced the following deliverables:

1. **Android Application (`todosync-2.0-debug.apk`):** A fully functional, installable APK file that runs on Android devices.
2. **Backend Server Script (`server.py`):** A Python Flask script capable of running on any networked computer to manage the database.
3. **Source Code:** Clean, modularized Python code (`main.py`) and Kivy Language (KV) definitions for the user interface.
4. **Documentation:** Comprehensive setup guide and architectural overview.

Progress Overview

The project has reached its conclusion with **100% of the planned objectives achieved**. We have successfully transformed the initial concept of a local to-do list into a robust, network-synced application using a Client-Server architecture. All critical features—including secure user authentication, real-time task synchronization across devices, and local push notifications—have been implemented and validated on physical hardware.

While the migration from a local SQLite database to a RESTful API presented significant architectural challenges regarding data consistency and widget initialization, these were resolved without delaying the project timeline. Consequently, no tasks are currently pending or behind schedule. The software is fully functional, stable, and ready for final presentation and deployment.

Codebase Information

Repository Link - https://github.com/lakshya-singhal/To-Do-List-with-Mobile-Notification.git
Branch – Main branch

- **Repository Structure:**
  - `main.py`: Main entry point for the Android Client application.
  - `server.py`: The Flask backend server.
  - `buildozer.spec`: Configuration file for compiling the Android APK.
  - `master_todo.db`: The server-side database (auto-generated).
  - `local_user.db`: Client-side session storage.
- **Key Commits/Milestones:**
  - *Commit:* `Initial UI Setup` - Basic KivyMD screens.
  - *Commit:* `Add Notifications` - Integrated Plyer logic.
  - *Commit:* `Server Migration` - Replaced SQLite queries with `requests.post/get`.
  - *Commit:* `Fix UI Crash` - Resolved KV loading order and ID access.
  - *Commit:* `Final Build Config` - Updated spec with Internet permissions and exclusions.

Testing and Validation Status

Testing Component	Status	Outcome/Notes
Unit Testing (Backend)	**Passed**	Verified all API endpoints (`/login`, `/add_task`, `/get_tasks`) using manual requests. The server correctly handles JSON data and updates the SQLite database.
Integration Testing	**Passed**	Confirmed successful communication between the Android Client and Flask Server over a local Wi-Fi network. Data synchronization (Add/Delete/Update) is instant.
Device Testing (Android)	**Passed**	The `.apk` installed successfully on a physical Android device. Permissions for `INTERNET` and `POST_NOTIFICATIONS` were requested and granted correctly at runtime.
Notification Testing	**Passed**	Verified that local push notifications trigger exactly at the scheduled time and that the "notified" status syncs back to the server to prevent duplicate alerts.

Deliverables Progress

Key Deliverable	Status	Notes
Android Application Package (.apk)	**Completed**	The final build (`todosync-2.0-debug.apk`) is compiled, signed, and successfully tested on a physical Android device.
Backend Server Script (`server.py`)	**Completed**	The Flask server is fully operational, capable of handling concurrent requests for login, task management, and sync.
Client Source Code (`main.py`)	**Completed**	The frontend code is finalized, featuring the updated Material Design UI and fully integrated API logic.
Project Documentation	**Completed**	The project abstract, architectural overview, and setup instructions have been drafted and finalized.