PROJECT ABSTRACT

Offline Note-Taking App with Sync Dashboard

An offline-first Android note-taking app with rich-text editing, tags and search, plus a sync centre that queues changes offline, syncs across linked devices, and resolves edit conflicts with a review screen — built-to-order with source code, report, PPT and viva kit.

01. Title

Offline Note-Taking App with Sync Dashboard

02. Introduction / Background

Students take notes everywhere — in lectures with patchy Wi-Fi, in the metro, in libraries — and the notes that matter are the ones captured in the moment. Cloud-only note apps fail exactly there: no network means no capture, or worse, silently lost edits. This project builds an offline-first note-taking app: every keystroke is saved to the device instantly, a sync queue records each change, and when connectivity returns the queue replays against the server. A sync dashboard shows pending changes, per-device status and — the interesting engineering part — a conflict-review screen for notes edited on two devices before syncing.

03. Problem Statement

Students either depend on cloud note apps that break offline or on local notes that never leave one device. Moving between a phone and a laptop creates version confusion: which edit is the latest, and what happens when both changed the same note? The project solves this with an offline-first design plus explicit, user-visible sync state and conflict resolution, instead of silent last-writer-wins data loss.

04. Objectives

- Build a rich-text note editor (bold, italic, headings, lists, checklists) with instant local save.
- Implement tags, full-text search and note organization.
- Design an offline-first sync queue: every mutation recorded with a sequence number, replayed in order when online.
- Build a sync dashboard showing pending changes, sync progress and per-device status.
- Implement conflict detection and a side-by-side review screen for notes edited on multiple devices.
- Support linking multiple devices (phone, laptop, tablet) to one account.

05. Proposed Methodology

The editor writes to the local database on every change (debounced), and each mutation is appended to a sync queue with a monotonically increasing sequence number and a device ID. When the network is available, the queue replays in order; server acknowledgements advance the high-water mark. Each note carries a version vector; if the server’s version has advanced on a different device since this device’s last sync, the incoming and local versions are both preserved and surfaced in the conflict-review screen, where the user picks a winner. The sync dashboard reads queue depth, last-sync timestamps per device and conflict counts directly

from these structures.

06. System Architecture / Working

Three layers: (1) Editor — rich-text editing, tags, search index, all backed by the local database; (2) Sync engine — mutation queue, sequence numbers, replay logic, version vectors and conflict detection; (3) Sync centre UI — progress, pending-change list, linked-device statuses and the conflict-review screen. The working flow is: type (saved locally instantly) → change queued → network returns → queue replays → dashboard shows progress → conflicts, if any, await user review.

07. Hardware / Software Requirements

- Android phone (Android 8.0+) — no other hardware needed
- Kotlin or Java with Android SDK; SQLite (Room) for local storage
- Rich-text editor component; background work manager for the sync queue
- A simple sync server endpoint (REST) for multi-device sync; core note-taking works fully offline

08. Expected Outcomes

A working Android app where notes are captured and searchable with zero network, the sync centre shows honest pending/synced state across linked devices, and a genuine two-device edit conflict can be produced and resolved in the review screen. Sync timing figures are design targets; conflict behaviour is deterministic and demonstrated in the report.

09. Applications

- Lecture and revision notes for students
- Field notes where connectivity is unreliable
- Personal knowledge base across phone and laptop
- Template for offline-first patterns in other student apps

10. Future Scope

- End-to-end encryption of note content before sync
- Real-time collaborative editing with operational transforms
- Handwriting and sketch attachments
- Web client reusing the same sync protocol