ICTU Community Application
Technical Documentation

Generated for Presentation

February 27, 2026

Introduction

ICTU Community is a Kotlin-based mobile application tailored to university environments. It provides information flow and academic management for students, teachers, staff, and administrators. The backend is implemented with Node.js/Express while Supabase serves as the database.

1 User Roles and Features

1.1 Students

- Read feeds, news, and newsletters (field-specific: coding, marketing).
- Access class schedules and recordings.
- Receive alerts for exams, CA, resits.
- Notifications for assignments and registrations (experimental).

1.2 Teachers

- Publish courses and upload materials.
- Create and send assignments.
- Read feeds, news.
- Assign class delegates from enrolled students.

1.3 Staff

Basic features:

- Manage timetables and classrooms.
- Handle administrative requests (transcripts, registration).
- Publish campus-wide announcements.

1.4 Administrators

- CRUD user accounts and manage roles.
- Configure system settings (semester dates, policies).
- Moderate feed and news content.

2 System Architecture

2.1 Frontend

Kotlin Android application employing MVVM architecture. Components include:

- ViewModels using LiveData or StateFlow.
- Retrofit for REST API communication.
- Hilt/Koin for dependency injection.

2.2 Backend

Node.js with Express framework. Key patterns:

- Controllers and services per domain (users, feeds, notifications).
- JWT for authentication and RBAC.
- RESTful endpoints with JSON responses.

2.3 Database

Supabase (PostgreSQL) with tables for users, roles, feeds, classes, assignments, notifications. Row-level security enforces access policies.

3 Requirements

3.1 Functional Requirements

1. User authentication and authorization.
2. CRUD operations for all major entities.
3. Real-time or push notifications.
4. File uploads and streaming for lectures.

3.2 Non-Functional Requirements

- Performance: API latency ¡200ms.
- Scalability: handle thousands of concurrent users.
- Security: HTTPS, JWT, RLS.
- Maintainability: modular, documented codebase.
- Usability: intuitive UI/UX, accessibility.

4 Design Patterns

- MVVM (frontend).
- Repository pattern for data access.
- Observer pattern for notifications.
- Singleton for shared services.
- Dependency Injection (Hilt/Koin).

5 User Stories

Role	Story
Student	As a student, I want to log in and see the latest news.
Student	As a student, I want to register for classes and get notified.
Teacher	As a teacher, I want to upload course materials.
Teacher	As a teacher, I want to assign a class representative.
Staff	As staff, I want to post campus announcements.
Admin	As an administrator, I want to deactivate a user.

6 Diagrams and Models

6.1 Use Case Diagram

6.2 Class Diagram

6.3 Sequence Diagram

7 Future Enhancements

- Real-time chat.
- Analytics dashboard.
- Offline caching and synchronization.
- Multi-language support.