25 UmiDo: AI 기반 스마트 알림 시스템

소속 정보컴퓨터공학부 분과 B 팀명 UmiZoomi 참여학생 카드로브 아딜렛, 예기즈바에브 졸란 지도교수 조환규

Problem

Problem Background

Fragmented Life Management and Cognitive Overload
Unintelligent and Passive Reminder Systems
Disconnected Communication and Lack of Oversight

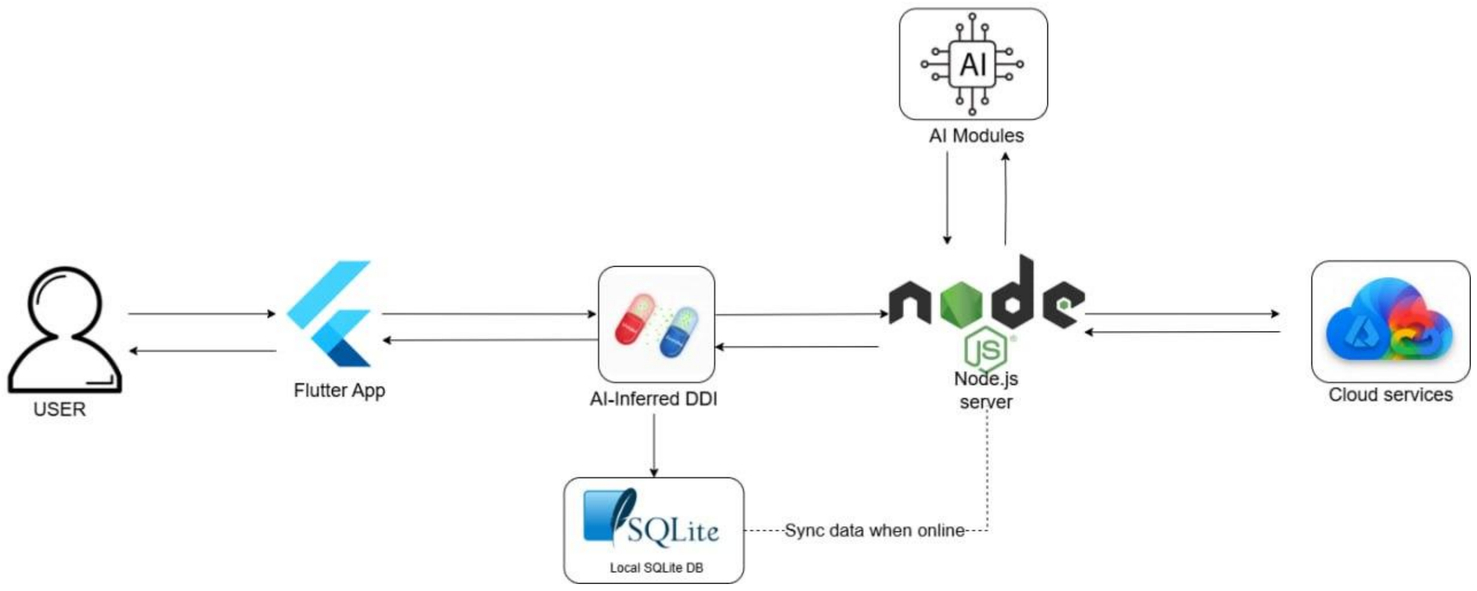
Project Objective

Build a smart notification system with AI/NLP that can:

- **Develop an AI-Powered Prescription Parser**
- **Build an Adaptive, Cross-Domain System**
- **Create an Integrated Communication and Monitoring Hub with Chatbot**
- **Implement Comprehensive Dashboards for Tracking and Oversight**

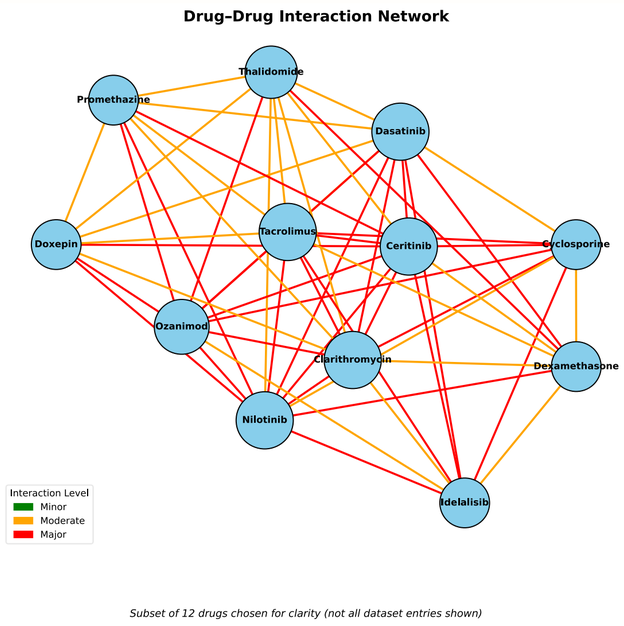
System

Workflow Explanation



Authentication: Login & user management handled via **Google Cloud** or **Node.js**.
Data Storage: Tasks saved in both **SQLite (local offline)** and **Azure SQL (cloud)**.
Offline Mode: If offline, data stays in **SQLite** and later **syncs to Azure** when online.
Backend: **Node.js server** manages sync, requests, and communication.
AI/NLP Modules: Includes **chatbot**, **document parser**, and **NLP** for smart document parsing.
Drug–Drug Interaction (DDI): Dedicated module checks for medication conflicts.
Notifications: Smart reminders generated and delivered back to the user.

AI Modules Pipeline



Subset of 12 drugs chosen for clarity (not all dataset entries shown)

DDI

Data Collection: Normalize drug names; gather chemical structures, pharmacology, and side-effect info → **Feature Engineering**: Generate drug pairs; compute chemical similarity, shared targets, and side-effect overlaps → **Signal Detection & Labeling**: Extract potential interactions from literature, co-prescription patterns, and FAERS → **Model Training & Analysis**: Train multiple ML algorithms to predict interaction likelihood and severity; interpret key features

Parser

Prescription Parsing: Extract medication names and instructions from prescription → **Medication Matching (RapidFuzz)**: Match extracted medication names to the normalized drug list using string similarity → **Instruction Processing (NLP → JSON)**: Parse instructions and convert them into structured JSON suitable for SQL storage

Chatbot

User Input Processing: Capture and preprocess the user's text → **Primary Matching**: Quickly match input to predefined instructions with high confidence → **Secondary Matching (Fallback)**: Attempt a more flexible match if primary fails → **Gemini API (Fallback of Fallback)**: Handle ambiguous or unmatched inputs → **Query Execution & Response Generation**: Run the SQL query and generate the response