Project Report Format

1. INTRODUCTION

1.1 Project Overview

IntelliSQL is a natural language-to-SQL interface designed to simplify data retrieval. By utilizing the **Gemini 1.5 Flash** model, it allows users to query a student database using plain English commands.

1.2 Purpose

The purpose is to bridge the gap between human language and complex database syntax, enabling non-technical staff to extract insights without writing code.

2. IDEATION PHASE

2.1 Problem Statement

Administrators often cannot access critical student data because they lack SQL skills, leading to a bottleneck where they must wait for IT assistance for simple reports.

2.2 Empathy Map Canvas

- **Says**: "I wish I could just ask the computer for the average marks."
- **Thinks**: "The database is too complicated for me."
- **Feels**: Frustrated by the technical barrier and dependency on others.
- **Does**: Manually searches through paper records or waits days for IT replies.

2.3 Brainstorming

We explored using traditional query builders but decided on an **LLM-driven approach** because it offers the most intuitive user experience by mimicking human conversation.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. **Trigger**: User needs a placement report.
2. **Action**: User types "Which students were placed at Google?" into IntelliSQL.
3. **Process**: AI translates query $\rightarrow$ Regex cleans code $\rightarrow$ SQLite executes.
4. **Outcome**: User receives a clean data table instantly.

3.2 Solution Requirement

- **Functional**: Must convert English to SQL and render database tables.
- **Technical**: Requires a secure connection to the Gemini API and a local SQLite instance.

3.3 Data Flow Diagram

1. **User Input** (English) → **Gemini API**
2. **Gemini API** (Raw SQL)→ **Regex Filter**
3. **Regex Filter** (Clean SQL→ **SQLite Database**
4. **Database** (Result Set) → **Streamlit UI**

3.4 Technology Stack

- **Language**: Python
- **AI Model**: Google Gemini 1.5 Flash
- **Frontend**: Streamlit
- **Database**: SQLite

4. PROJECT DESIGN

4.1 Problem Solution Fit

The solution fits the customer's behavioral pattern of "searching" rather than "coding," removing the stress of syntax errors.

4.2 Proposed Solution

An intelligent dashboard where users can query student records, track placements, and analyze performance marks through a simple chat-like interface.

4.3 Solution Architecture

A modular **3-Tier Architecture** consisting of a Presentation Layer (Streamlit), Logic Layer (Gemini/Regex), and Data Layer (SQLite).

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

We utilized a 2-Sprint agile schedule over 10 days, prioritizing the database setup and API integration first, followed by UI polishing and performance testing.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

- **Response Time**: Averaged **1.8 seconds** for query generation and execution.
- **Concurrency**: Successfully handled rapid-fire requests without API throttling.

7. RESULTS

7.1 Output Screenshots

- **Home Page**: Professional dark-themed dashboard.
- **Query Result**: Interactive data table showing student placements.

8. ADVANTAGES & DISADVANTAGES

- **Advantages**: No coding required, rapid data access, and highly scalable.
- **Disadvantages**: Occasional "hallucinations" on extremely complex queries; requires an active internet connection for the API.

9. CONCLUSION

IntelliSQL successfully democratizes data access by allowing anyone, regardless of technical background, to interact directly with structured databases.

10. FUTURE SCOPE

- **Voice-to-SQL**: Adding voice commands for hands-free querying.
- **Multi-DB Support**: Expanding to connect with MySQL and PostgreSQL cloud instances.

11. APPENDIX

Source Code

- **App Logic**: Located in app.py.
- **DB Script**: Located in sql.py.

Dataset Link

- Local SQLite file: data.db.