Project Design Phase
Solution Architecture

Date	20 February 2026
Team ID	LTVIP2026TMIDS66135
Project Name	IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro
Maximum Marks	4 Marks

Solution Architecture:

Solution Architecture bridges the gap between business needs and technical implementation. For **IntelliSQL**, the goal is to ensure a scalable, responsive, and secure Text-to-SQL system using **Streamlit**, **Python**, and **Google Gemini AI**, with clear data flow for administrators and non-technical users.

Objectives of the Architecture:

- **Define Component Interaction:** Establish how the Streamlit UI, Gemini API, and SQLite database interact.
- **Separation of Concerns:** Maintain clear boundaries between the UI, AI logic, and data storage.
- **Smooth Data Flow:** Enable rapid translation of natural language into SQL and immediate data retrieval.
- **Support Future Scaling:** Allow for easy upgrades to more powerful LLMs (e.g., Gemini Pro) or cloud databases.

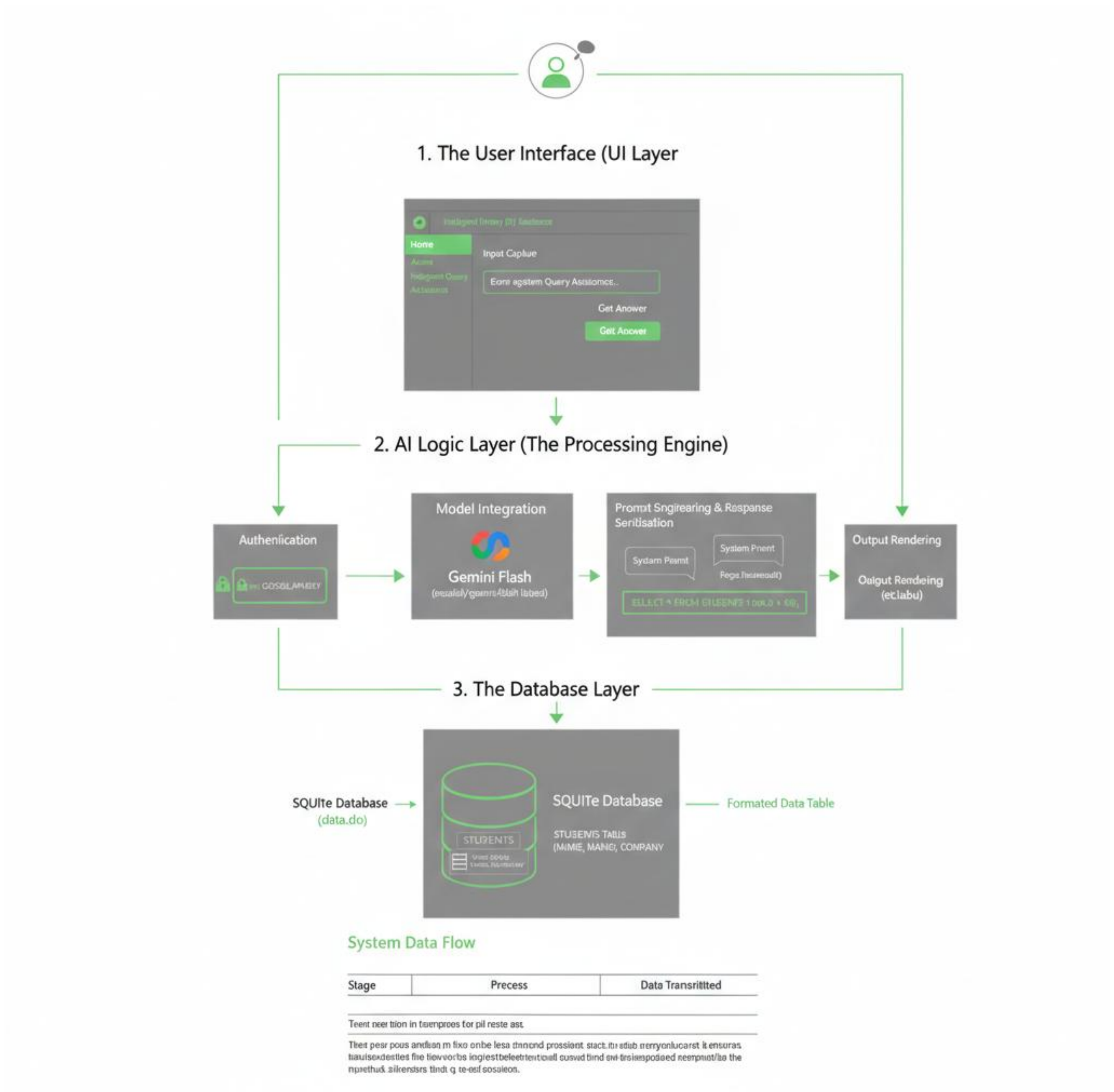
Architecture Layers and Components:

Layer	Components / Tools Used	Description
Frontend	Streamlit, HTML/CSS	Web interface featuring a dark theme and sidebar navigation for Home, About, and Query pages.
Backend	Python, Google Generative AI SDK	Core logic for prompt engineering, SQL generation via Gemini Flash, and Regex-based sanitization.
Database	SQLite (data.db)	Local relational storage for the STUDENTS table including Names, Classes, Marks, and Companies.
Security	.env (python-dotenv)	Secure management of the GOOGLE_API_KEY to prevent exposure in source code.
Dev Tools	Git, VS Code, Regex	Development environment used for building, testing, and isolating valid SQL from AI responses.

Data Flow Summary

1. **User Journey:**
   - User accesses dashboard → inputs natural language question → system attaches context prompt → Gemini generates SQL → Regex cleans query → data displayed in table.
2. **Admin/System Journey:**
   - System loads API key → initializes database connection → validates SQL commands against the STUDENTS table schema.

Example - Solution Architecture Diagram:


1. The User Interface (UI Layer
Home
Input Caplue
Eore agstem Query Assiomce..
Get Anower
Get Anower
2. AI Logic Layer (The Processing Engine)
Authenlication
Model Integration
Gemini Flash
(eeaisly/gemrs4blsh labed)
Promnt Sngirearing & Raspanse Seritisation
Sydam Pesnt
System Pnent
Fega Ihsesoult)
ELLECT * FROM GILEENF 1 00L0 + 60,
Output Rendering
Oaigut Rendeing (eclabu)
3. The Database Layer
SQUIte Database
(data.do)
SQUITe Database
STUDENTS
STUSENIS TAlLls (MIMIE, MANG, CONRANY
Formated Data Table
System Data Flow
Stage
Precess
Data Transritited
Teent neer tion in tsemproes tor pil reste ast
Tbet pear pous andlan m fixo onbe lesa tnnond prossiant stact ltu stlab rerryonlucarst it ensuras tiauisexdesties fne tiewvorbs ingiestbeleetntentioall cusvd tind evi-tisiaspodoed neerpnot/ba the npaethuk silkensisrs tindt q te-esf sosaleon.

Feature	Handled By	Status
Natural Language Processing	Google Gemini Flash	✓ Implemented
SQL Extraction/Cleaning	Python Regex (re)	✓ Implemented
Database Persistence	SQLite	✓ Implemented
Secure Key Handling	.env, python-dotenv	✓ Implemented
Modular UI Navigation	Streamlit Sidebar	✓ Implemented
Performance Optimization	Gemini 1.5 Flash Model	✓ Optimized

Notes:

- The current design supports future enhancements like multi-table joins, voice-to-query integration, and export-to-CSV features.
- The modular 3-tier architecture ensures that the database can be migrated to a cloud-based SQL server (e.g., PostgreSQL) without rewriting the AI logic.