System Design Document

Mail Calendar AI Agent

Name: Aditya Rehpade
College: Indian Institute of Technology Bhilai

Personal Email: adityarehpade1@gmail.com
Institute Email: adityapr@iitbhilai.ac.in
GitHub: https://github.com/BitWiseNexus

Executive Summary

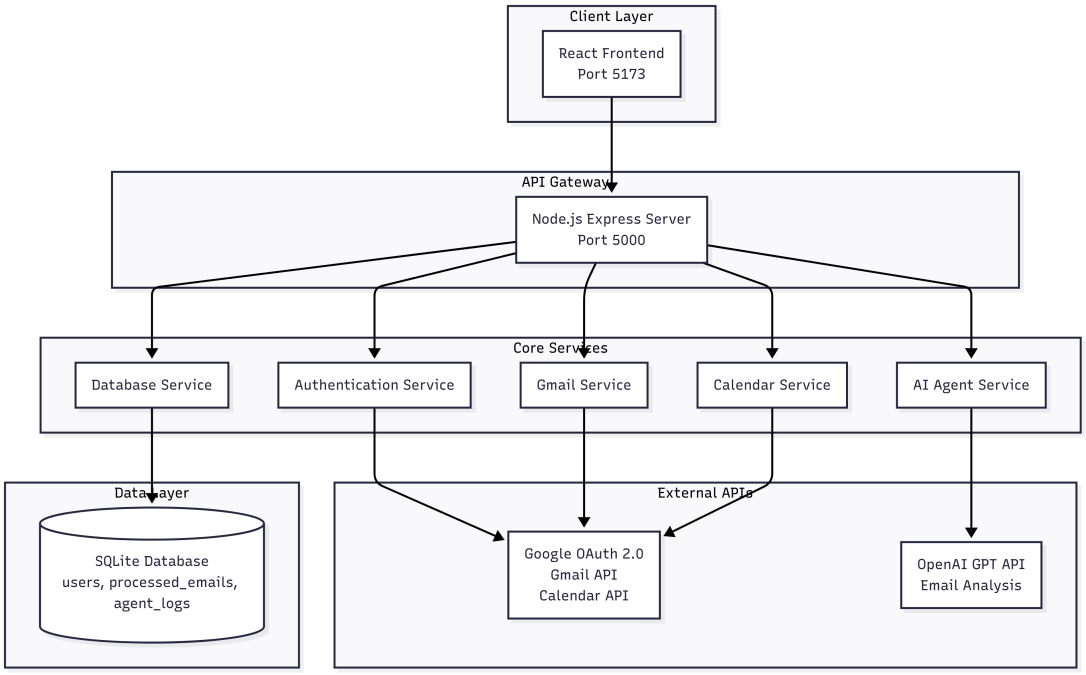
This project implements an intelligent **Mail → Calendar AI Agent** that transforms email overload into actionable calendar events. Unlike simple summarization tools, the system follows a full **agent reasoning loop**: analyzing content, extracting deadlines, planning events, and executing them seamlessly in Google Calendar.

The solution features a **multi-tier architecture** with a React frontend, Node.js backend, and AI-driven email analysis using Gemini, combined with Google APIs for Gmail and Calendar integration.

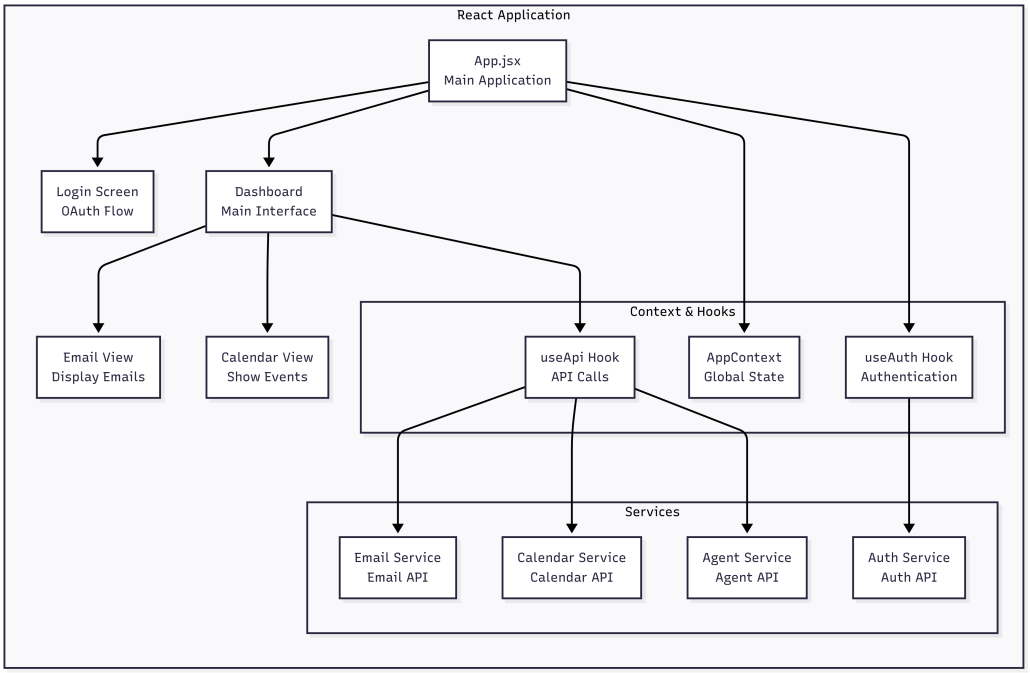
The project demonstrates originality by going beyond email classification to provide **end-to-end automation**, from reading emails to creating calendar events ensuring responsiveness, privacy, and security.

1 Architecture Overview

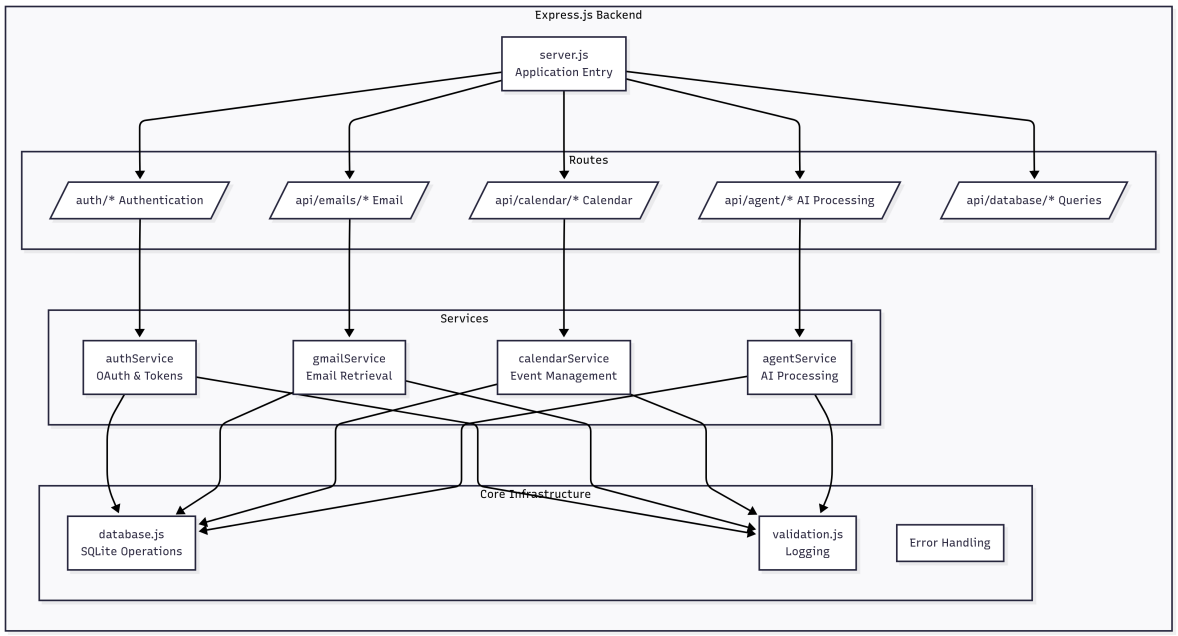
1.1 System Architecture



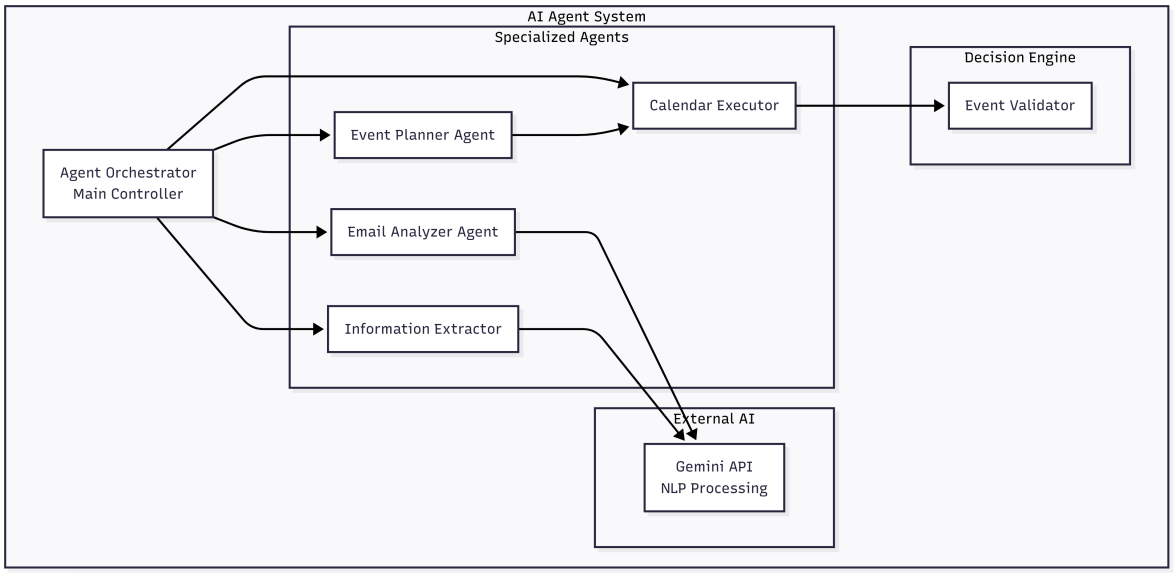
1.2 Component Breakdown (Frontend)



1.3 Backend Architecture

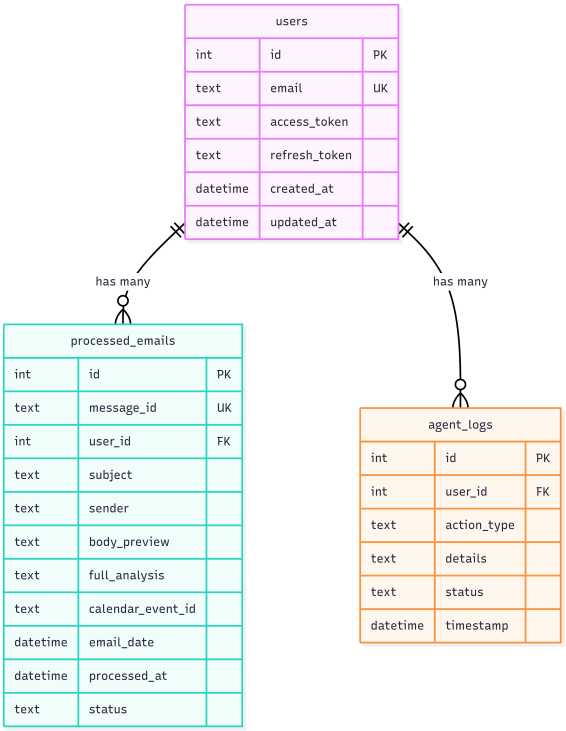


1.4 AI Agent Architecture

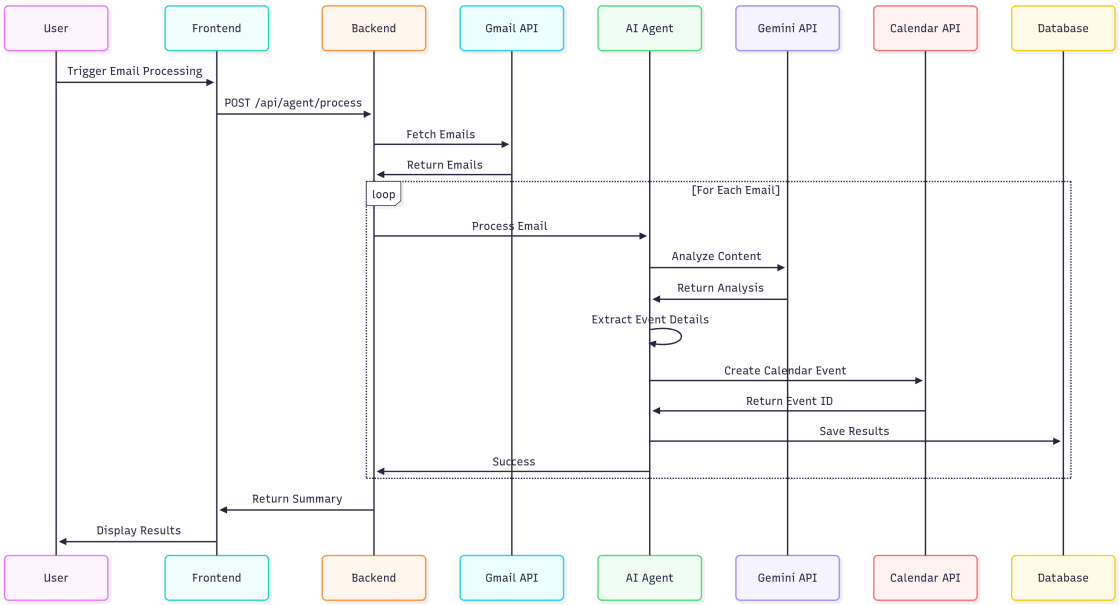


2 Data Design

2.1 Database Schema



2.2 Data Flow Architecture



3 Technology Stack

3.1 Selected Technologies

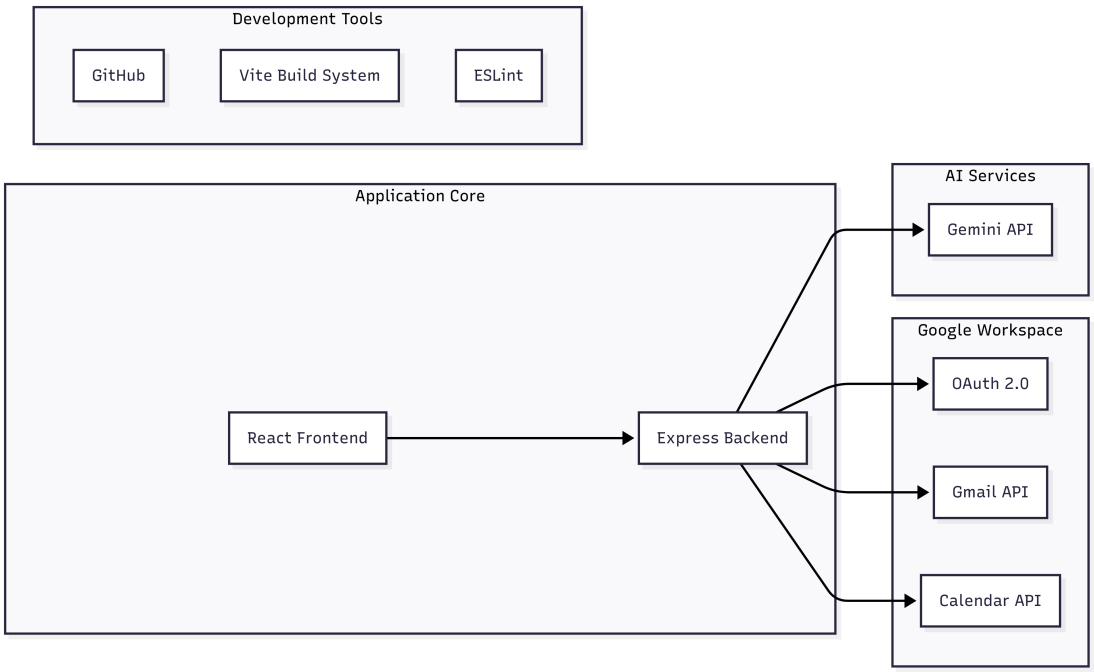
- **Frontend:** React, Vite, Tailwind CSS, Lucide React, Context API
- **Backend:** Node.js, Express.js, SQLite (single-user)
- **LLM:** Gemini via API
- **Integrations:** Google APIs- Gmail (read), Calendar (write)

Why these technologies

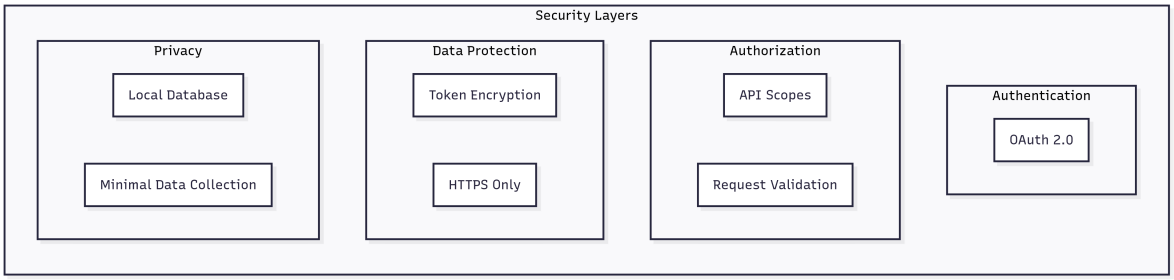
- **Frontend & Backend Stack:** I have experience with React, Vite, Tailwind CSS, Lucide React, Context API, Node.js, Express.js and SQLite, and I am comfortable using them.
- **Gemini API:** Free and available during development.

4 Integration & Security

4.1 Integration Architecture



4.2 Security Architecture



4.3 Operational & Bonus Features

- **Monitoring UI:** Logs dashboard, email processing status, event management.
- **Operational Features:** Batch processing, error recovery, retry policies.
- **Authentication:** OAuth for Google, role-based access for admin features.

5 Performance & Quality Assurance

5.1 Performance Capabilities

- Optimized for single-user with SQLite; design allows PostgreSQL migration for multi-tenant use.
- Light, event-driven workers to keep API latency low.
- Caching for processed messages to avoid repeated LLM calls.

5.2 Quality Measures

- **Code Quality:** ESLint, structured logging, input validation, error handling.
- **Testing:** Unit tests for parser/heuristics, API tests for endpoints, manual integration tests for OAuth flows.

Conclusion

The Mail Calendar AI Agent showcases modern web architecture and intelligent automation. By integrating React, Node.js, a large language model, and Google APIs, the system transforms email content into actionable calendar items while maintaining privacy and operational controls.

Repository

GitHub: https://github.com/BitWiseNexus/SDE_Deliverable_Application

LLM Interaction Log

- Log 1: ChatGPT Share #1
- Log 2: ChatGPT Share #2
- Log 3: ChatGPT Share #3
- Log 4: Claude Share