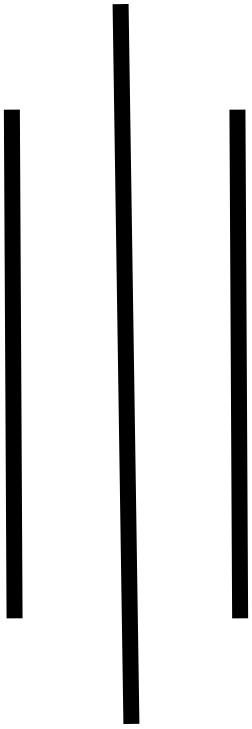
CS/CE 3354 Software Engineering
Final Project Deliverable 1



CalendrAI

Group#5

Dineshman Bajracharya
Omesh Reddy Sana
Ivan Pinon
Ryan Catarroja
Abhishek Haris
Pamela Espinoza Maldonado
Yaqub Ahmed
Aatif Mohammed Shahul Hameed

1. Project Goal/Motivation/ Task Delegation for 1st and 2nd deliverables (Updated version of Proposal)

2. OUR REPOSITORY URL LINK: https://github.com/omesh-s/CalendrAI

3. **Delegation of Tasks:**

Name of Member	Tasks
Dineshman Bajracharya	Outlining Software Process Model
Omesh Reddy Sana	Project Leader, Completed Github steps: 1.1 - 1.3, Applying Architectural
Ivan Pinon	Filling out this Delegation of Tasks, Final Project Draft Description
Ryan Catarroja	Document Submission, Designing Class Diagrams
Abhishek Haris	*Outlining the classes that will be used for the project*
Pamela Espinoza Maldonado	Creating Sequence Diagrams for Use Cases
Yaqub Ahmed	Handling Software Requirements both functional and non-functional.
Aatif Mohammed Shahul Hameed	GitHub step 1.4, Creating Use Case Diagrams, Document Revision

Table 1.1: Delegation of Tasks

4. **Software Process Model Selection**

For the development of our project, CalendrAI, our group will be following the Scrum methodology, a subset of Agile. We learned about multiple methodologies during our software engineering class and scrum is one of them. Scrum is well-suited for this project because it will allows us for incremental development, get frequent feedback, and interactive improvements, making it ideal for this, given our evolving requirements and AI integration challenges which we plan on adding later after we get the Minimum Viable Product(MVP) working for the calendar system.

4.1. Chosen Software Development Model: *Scrum (Agile)*

More technical detail on why we selected Scrum:

- Given that we will be first making the calendar system and then integrate the AI feature that will have direct access in modifying the calendar, our requirements will change frequently based on testing and feedback.
- We will first develop the core calendar functionalities such as Events, Agenda Views, Sharing features and then followed by gradual AI integration.
- We are following an MVP-first approach, meaning we will release a basic version early and improve it over time.
- Scrum enables us to frequent user testing and feedback cycles to refine the product so we can have students on campus to test our system.
- AI integration introduces technical risks, and Scrum allows us to experiment and pivot as needed.

4.2. How Scrum will be applied to our Project

- Sprint Planning: Each sprint (lasting two-three weeks) will have a set of planned deliverables.
- Weekly Standups: We will have short team meetings to track progress, address backlogs and make sure each individual team member is aligning with the overall goal.
- Sprint Review: At the end of each sprint, we will evaluate our progress and get user feedback.
- Sprint Retrospective: We will have reflection on what worked well and what can be improved for the next sprint.
- Product Backlog: We will have a prioritized list of tasks managed using ClickUp (Project Management Tool).

4.3. Sprint Breakdown (Tentative Plan): Might Change Later

Sprint	***Focus Area***	***Key Deliverables***
Sprint 1	Core Calendar Functions	Basic UI, event creation/editing, multiple views (day/week/month)
Sprint 2	Advanced Calendar Features	User authentication, recurring events, schedule conflict resolution, event sharing
Sprint 3	AI Integration (Phase 1)	AI-assisted event creation, testing API integrations
Sprint 4	AI Refinement & UX enhancements	Better UI/UX based on feedback, edge case testing on the AI system

Sprint 5	Testing & Final Adjustments	Bug Fixes, performance testing, final user feedback

Table 1.2: Sprint Breakdown

4.5. Tools:

- Version Control: Github (For source code management and collaboration)
- Task Management: ClickUp (For sprint planning, backlog management, and task tracking)
- Frontend & Middleware: Next.js + tailwind CSS (react based, efficient and developer friendly)
- Backend & Database: Google Firestore (serverless, beginner-friendly, and easy to integrate)
- Infrastructure & Deployment: Vercel (seamless deployment with Next.js, free and easy to use)
- AI Integration: OpenAI API (for AI-powered scheduling and NLP-based event creation)
- Testing: Unit & integration testing after each sprint to ensure stability and reliability

5. Software Process Model

5.1. Functional Requirements

1. Users must be able to create, modify, delete, and view events in the calendar system.
2. Users must be able to switch to different views, such as Agenda, Month, Weekly, and Daily views.
3. Users must be able to synchronize their events using the service Google Calendar.
4. The system must provide the user with AI-based schedule times and names upon request.
5. The system must allow users to share events.

5.2. Nonfunctional Requirements

1. The system should be able to handle multiple calendar events creation, modification, and editing without significant latency.
2. The system should not take more than one minute to perform Google Calendar synchronization.
3. The system must be able to support increasing numbers (1,000) of users without performance degradation.
4. User data must be stored in an encrypted manner and be encrypted in transit, adhering to encryption standards.

5. Mask or remove sensitive user data before processing through LLMs(OpenAI)

Requirement ID	Requirement Description	Priority	Mapped Use Case	Status
FR-01	Users must be able to create, modify, delete, and view events in the calendar system.	High	Create Event, Update Event, Delete Event, View Event	In progress
FR-02	Users must be able to switch to different views, such as Agenda, Month, Weekly, and Daily views.	High	View Event	In progress
FR-03	Users must be able to synchronize their events using the Google Calendar service.	Medium	Sync Events	Not Started
FR-04	The system must provide the user with AI-based schedule times and names upon request.	Medium	Create AI Task	In progress
FR-05	The system must allow users to share events.	Medium	Share Event	Not Started
NFR-01	The system should handle multiple calendar events without significant latency.	Low	Create Event, Update Event	In progress
NFR-02	Google Calendar synchronization must complete within one minute.	Medium	SyncEvents	Not Started
NFR-03	The system must support 1,000+ users without performance degradation.	High	All Use Cases	Not Started
NFR-04	User data must be encrypted in transit and storage.	Low	Sync Events, Create Event	Not Started
NFR-05	Sensitive user data must be masked or removed before LLM processing.	Low	Create AI Task	Not Started

6. Diagrams:

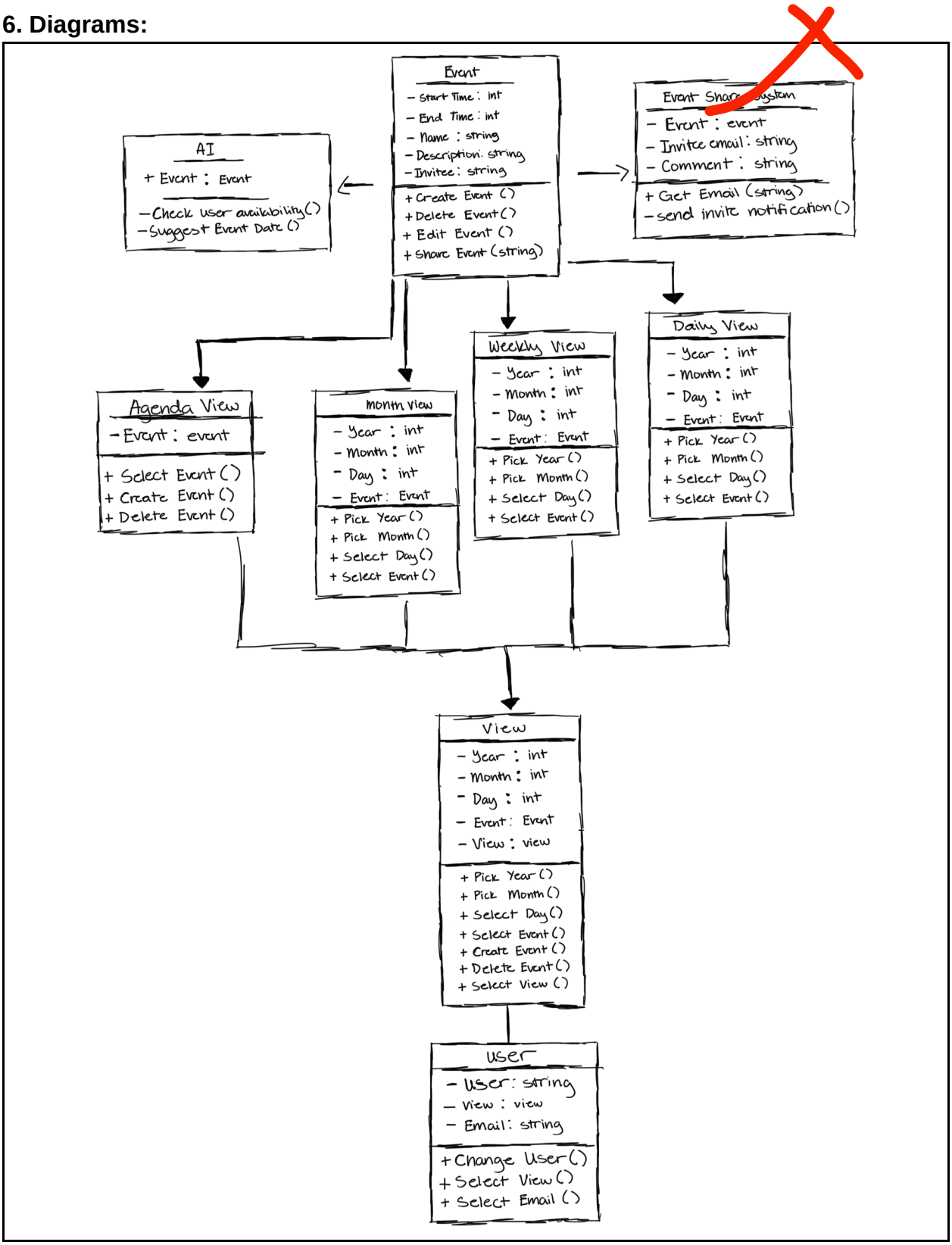


Figure 1: Class Diagram

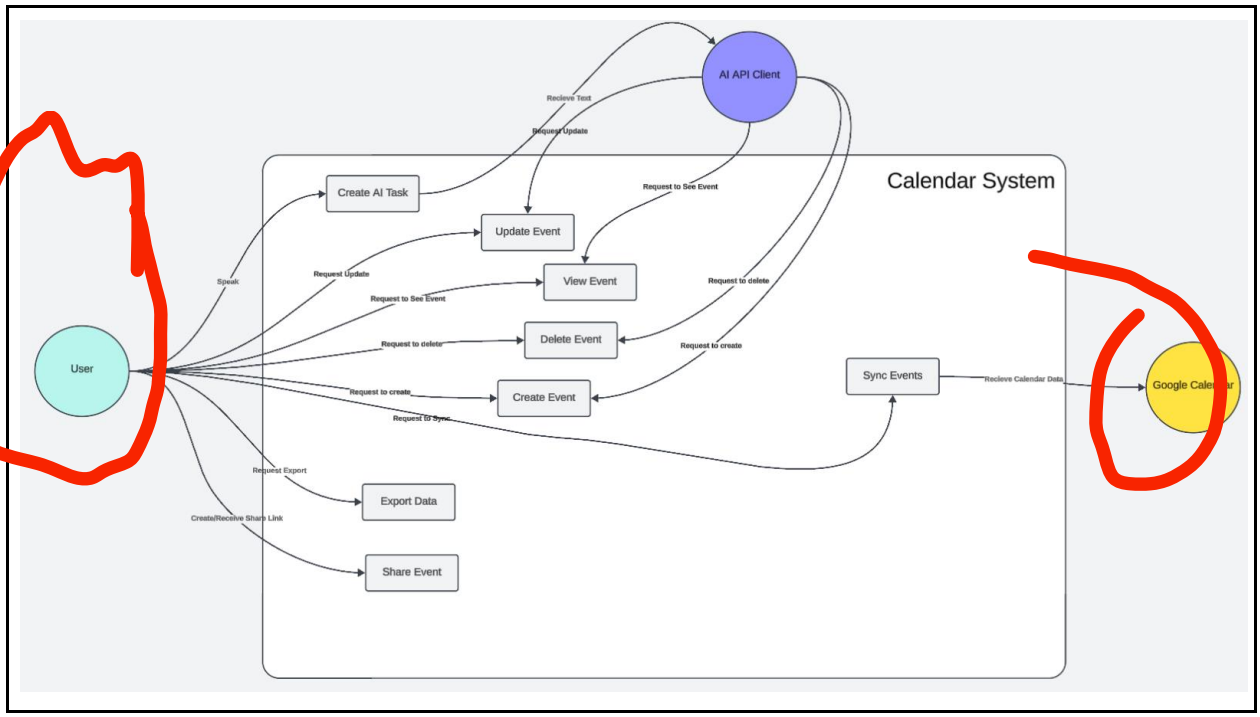


Figure 2: Use Case Diagram

Figure 4-7: Sequence Diagrams

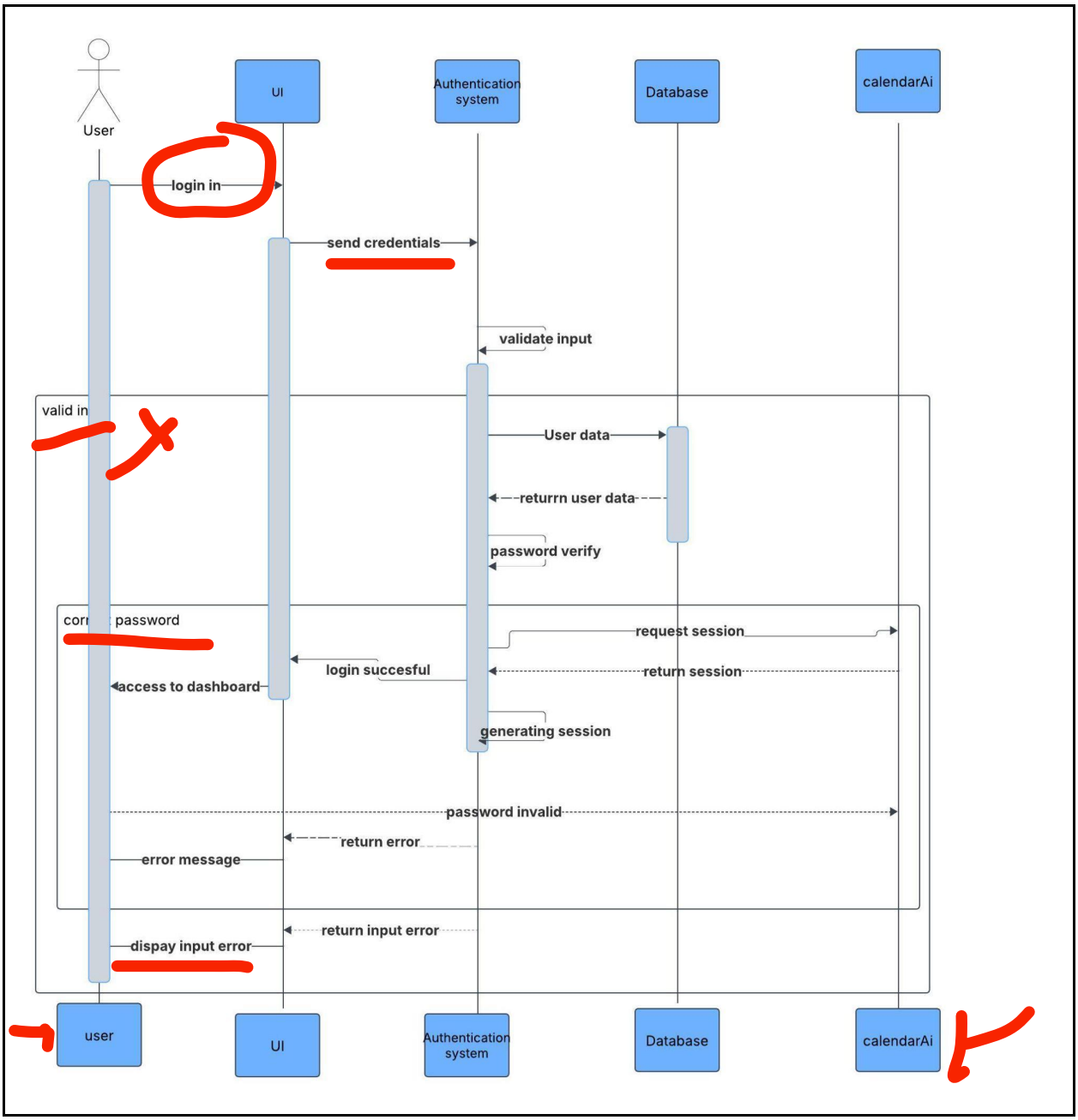


Figure 4: Login

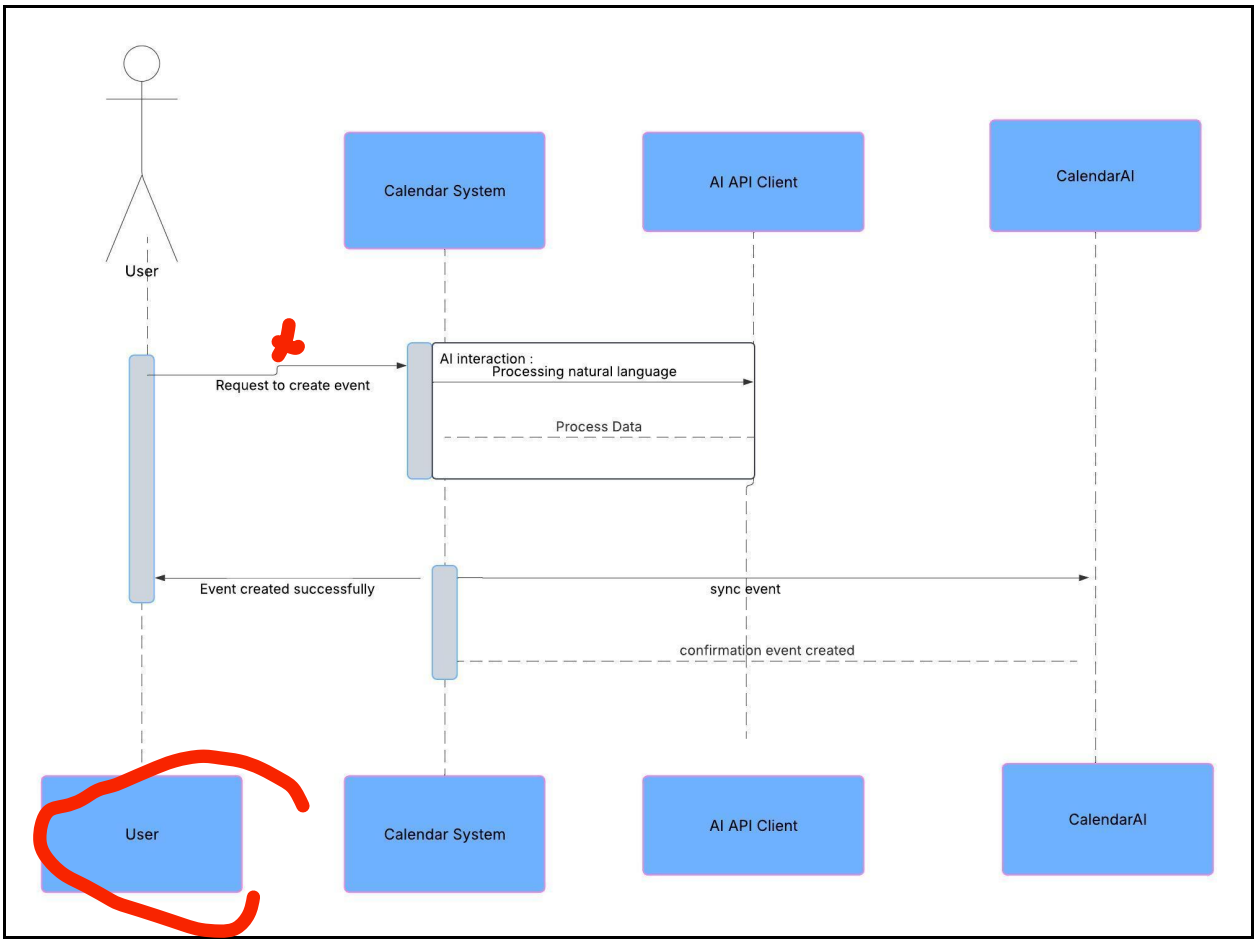


Figure 5: Create Event

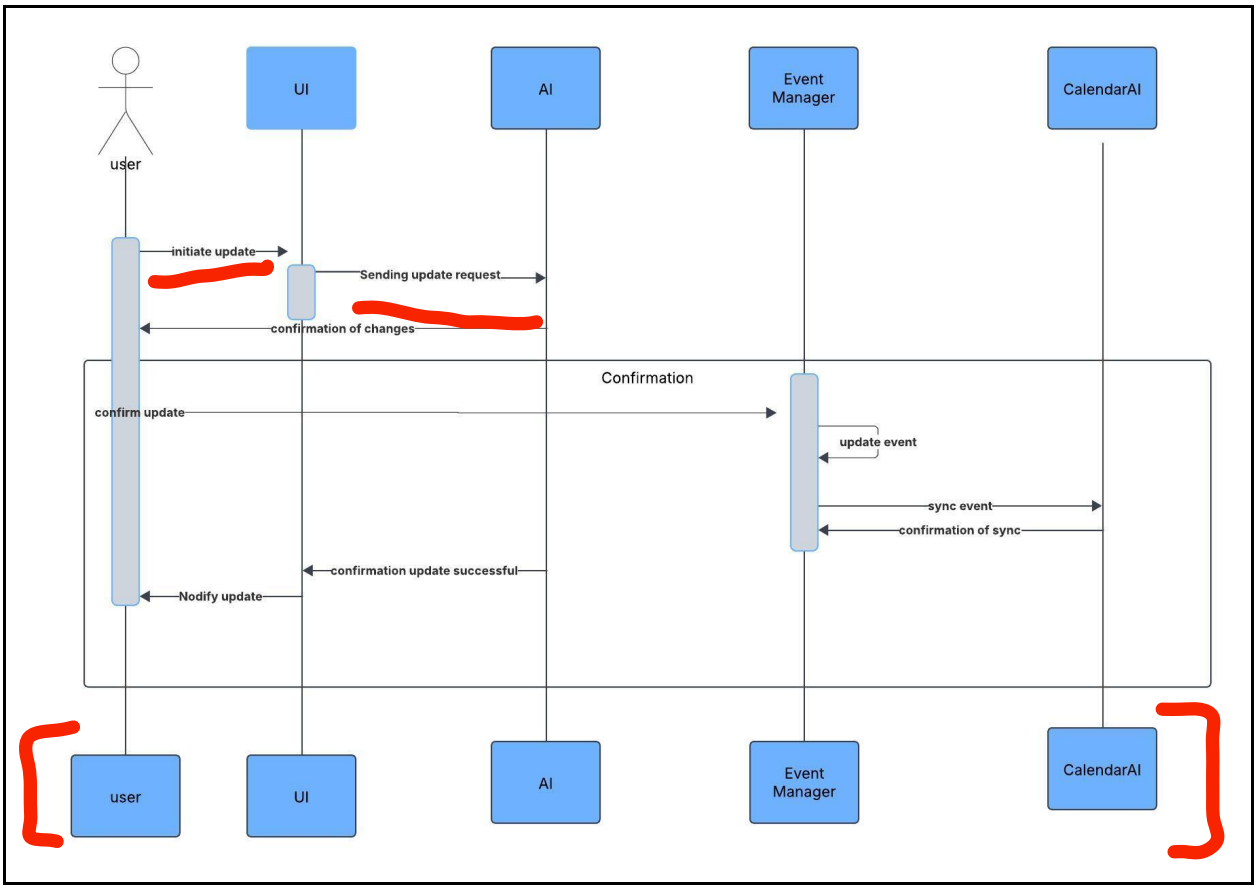


Figure 6: Update event

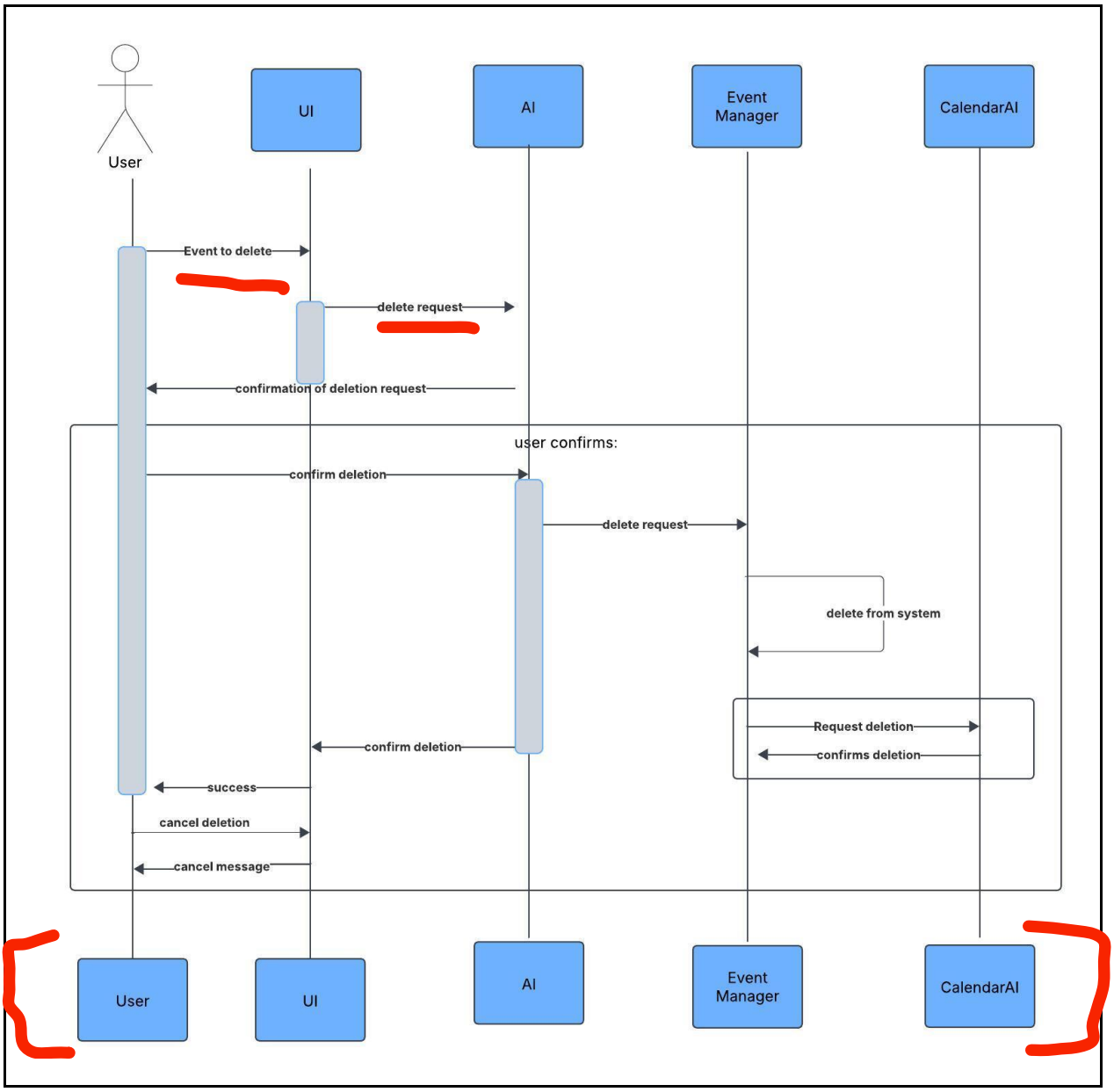


Figure 7: Delete Event

7. Architectural Design

For the architectural design of our project,Calendar AI, we choose to use Layered Architecture combined with Client-Server Pattern. Layered Architecture simplifies development, testing and maintenance by dividing the applications into different layers. The Client-Server pattern centralizes business logic and data processing on the server and allows multiple clients (web, mobile) to access the same server.

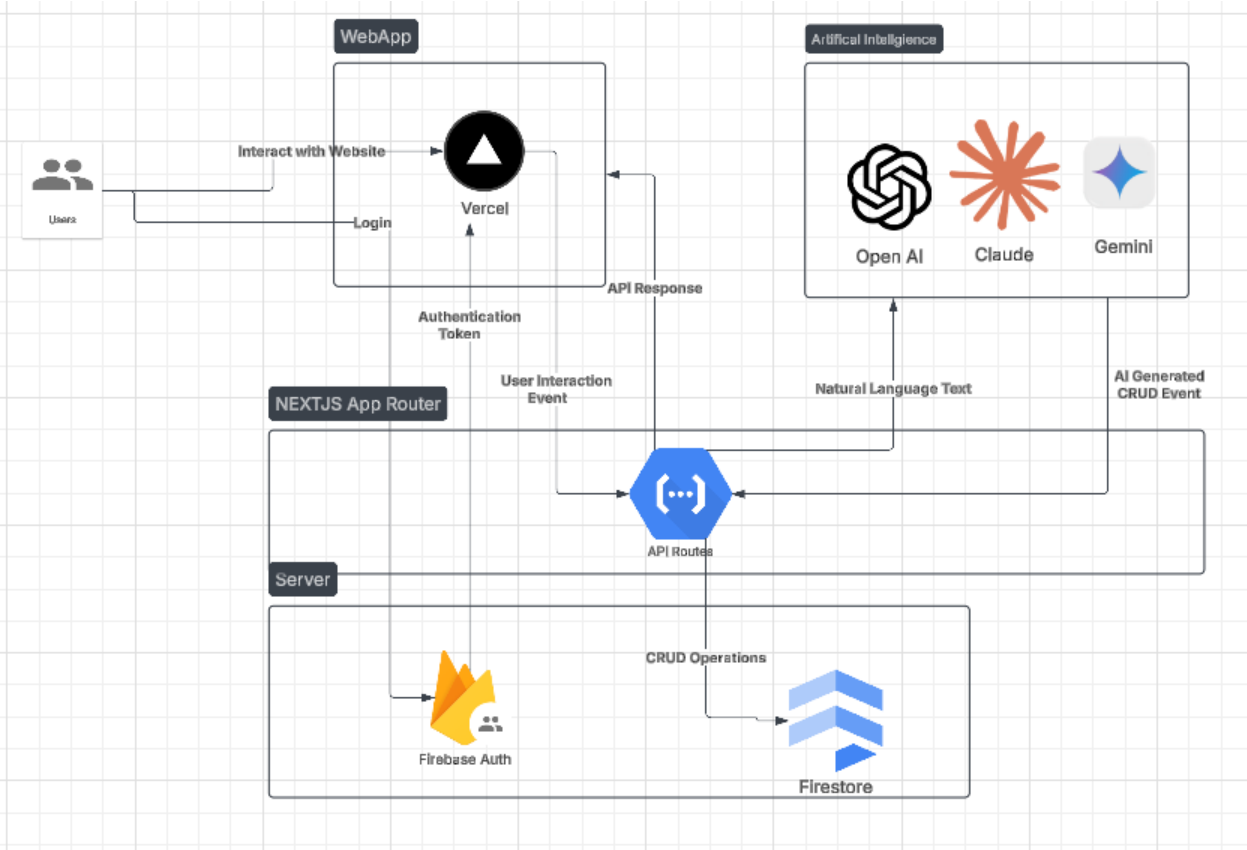


Figure 8: Architecture Diagram

7.1 Why was this Pattern Chosen ?

Layered Architecture

- Separations of Concerns: We structure the Calendar.ai into three distinct layers (Presentation, Application, Data). This simplifies development, testing and maintenance.
- Easier Updates: We can make changes to one layer without affecting another layer. This modularity makes it easy to maintain and update the calendar app. This is particularly beneficial since we are using Scrum for our Calendar App, a Agile Methodology and would need frequent updates.

- Independent Scalability: Layered Architecture allows for different layers for the Calendar.ai to be scaled independently. This allows us for one layer to be optimized without affecting the other layers.

Client Service Pattern

- Centralized Management: By using the client-server pattern, we centralize all data and business logic for Calendar.ai on the server. This makes it easier to manage user events, reminders, and scheduling rules in one place. Clients only handle the presentation and user interaction, ensuring a clean separation that simplifies updates and maintenance.
- Scalability and Performance: The client-server model allows us to scale Calendar.ai's backend infrastructure to handle growing user demands. For example, if more users join and start scheduling events, we can add more servers (horizontal scaling) or upgrade existing ones (vertical scaling) to ensure the app remains fast and responsive, even during peak usage.
- Security: Sensitive data like user events and personal details can be stored securely on the server, not on client devices. This reduces the risk of data breaches and ensures better control over access permissions. For instance, we can implement robust authentication and encryption protocols on the server to protect user information.
- User Experience: The client-server pattern enables Calendar.ai to support multiple platforms—web, mobile, and desktop—while maintaining a consistent experience. Users can access their calendars from any device, and all changes are synchronized in real-time through the server. This ensures seamless usability and accessibility, no matter how users interact with the app.