CS3354 Software Engineering
Final Project Deliverable 2

CalendrAI

Group #: 5

Names of Team Members

Pamela Espinoza Maldonado

Dineshman Bajracharya

Omesh Reddy Sana

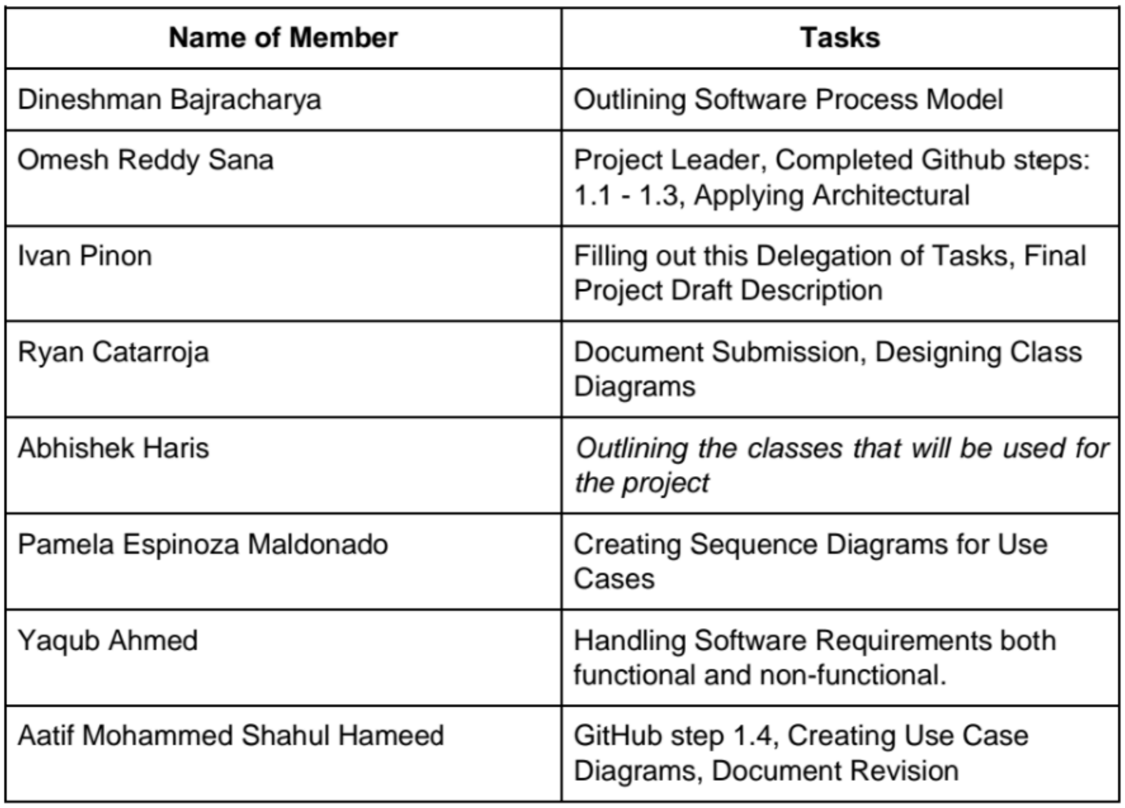
Ivan Pinon

Ryan Catarroja

Abhishek Haris

1. Delegation of Tasks

1st Deliverable:



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

2nd Deliverable:

Name of Member	Tasks
Omesh Reddy Sana	Well described delegation of tasks, Submit the repository
Dinesh Bajracharya	Describe the conclusion of the project
Ivan Pinon	Moving information from deliverable 1 to deliverable 2
Ryan Catarroja	Work on the presentation slides
Abhishek Haris	Deliverable 1 corrections

Pamela Espinoza Maldonado	Create a test plan for the software
Yaqub Ahmed	Compare our work with similar designs
Aatif Mohammed Sahul Hameed	Project Scheduling, Project duration, and staffing. Create supplemental documents. Finish Prototype

2. Project **Deliverable 1 Content: (Note: with all corrections based on feedback)**

- Project Goal/Motivation/ Task Delegation for 1st and 2nd deliverables
    - Our group has decided upon developing a widely accessible calendar application which will include at the very least the basic functions of: alternative view settings (day,week,month,etc.), event scheduling (one-time, reoccurring, etc.), schedule conflict resolution, share event functions, and visual distinction between different event types. But we also plan on implementing various features to further enhance user experience, such as AI-based smart scheduling.
- Software Process Model

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.

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.

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).

· **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

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

- Software Requirements
  - **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.

  - **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

- Diagrams (Use-Case/Sequence/Class/Activity)

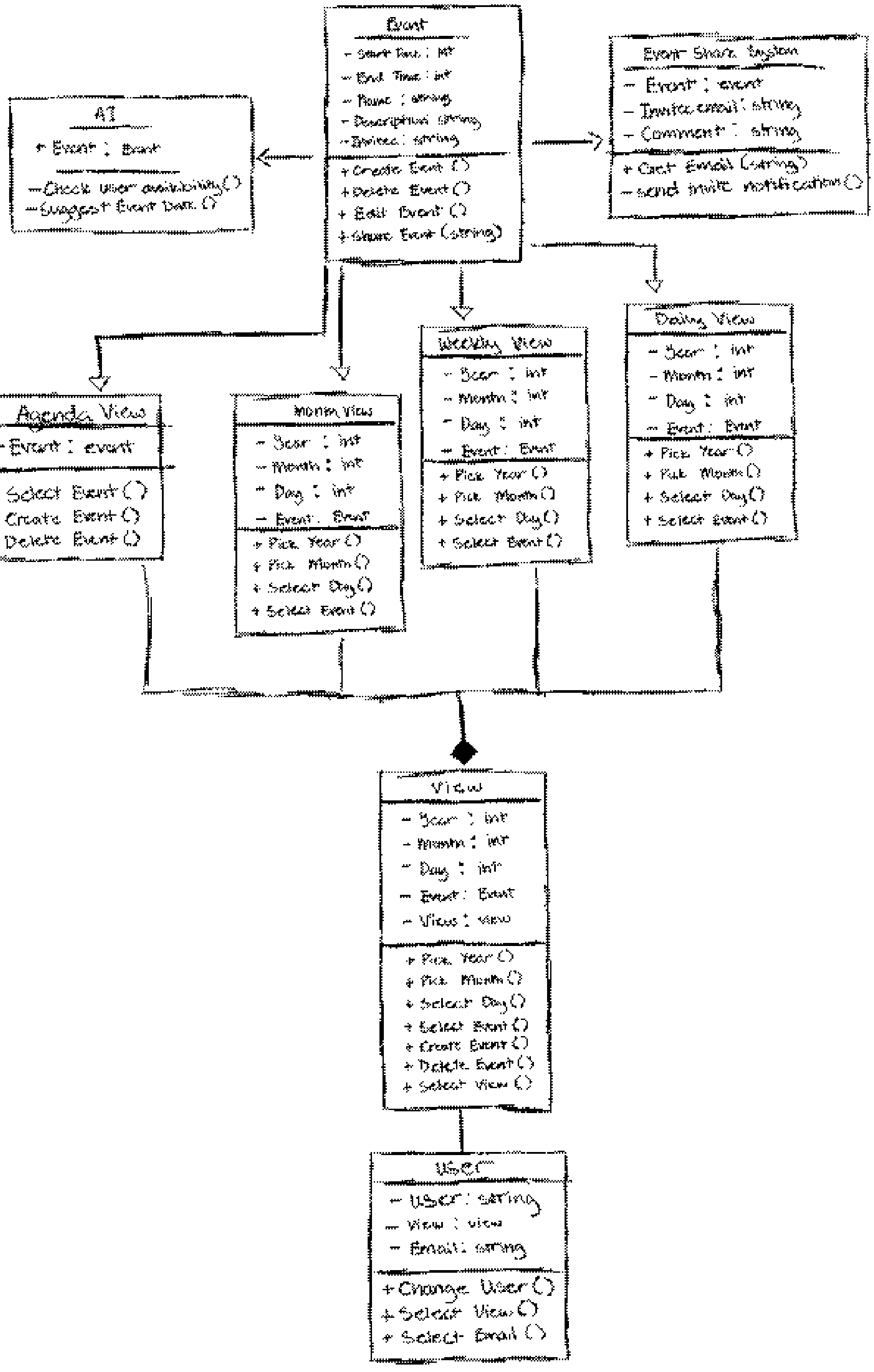
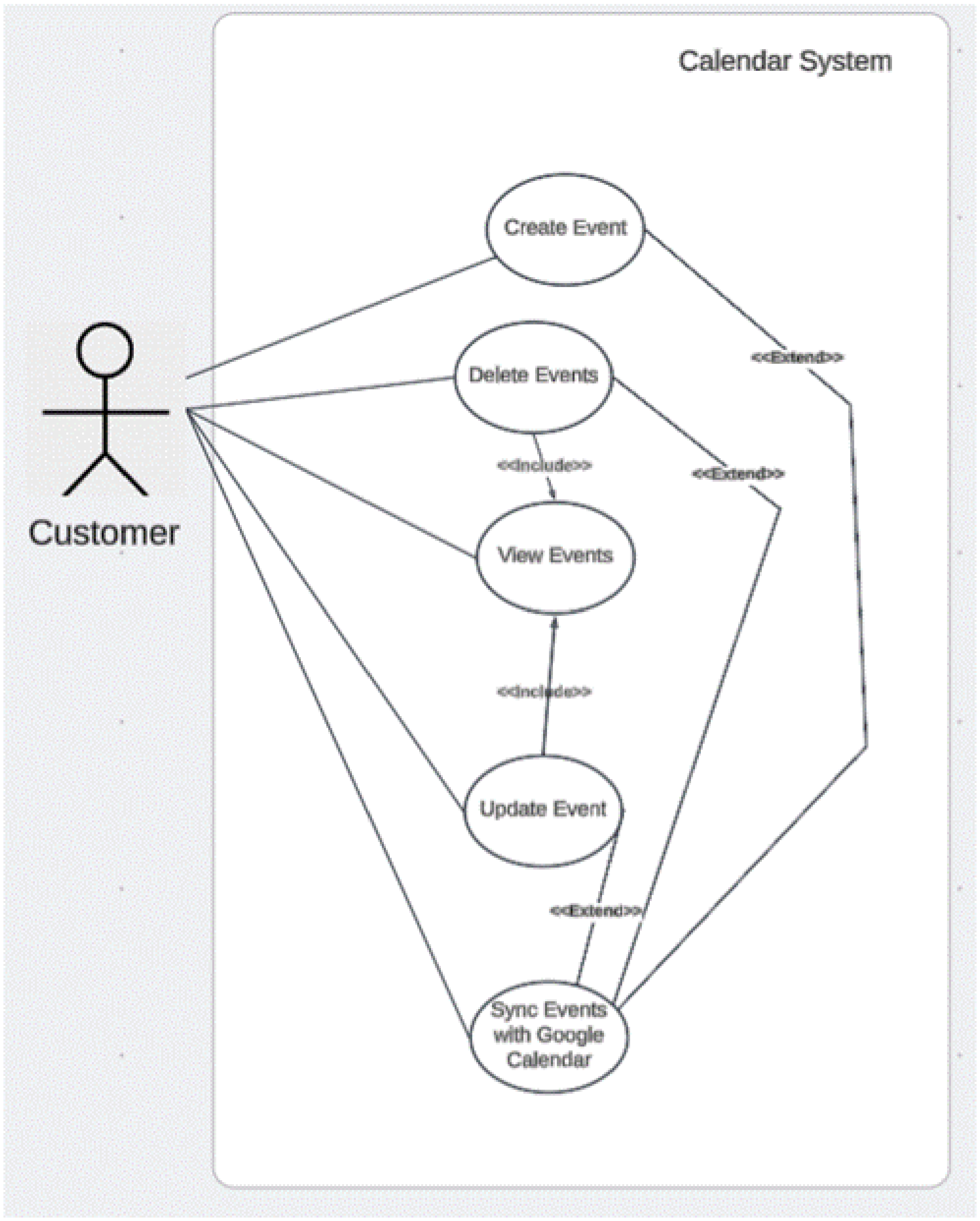


Figure 1: Class Diagram



-

Figure 2: Use Case Diagram

Figure 4-7: Sequence Diagrams

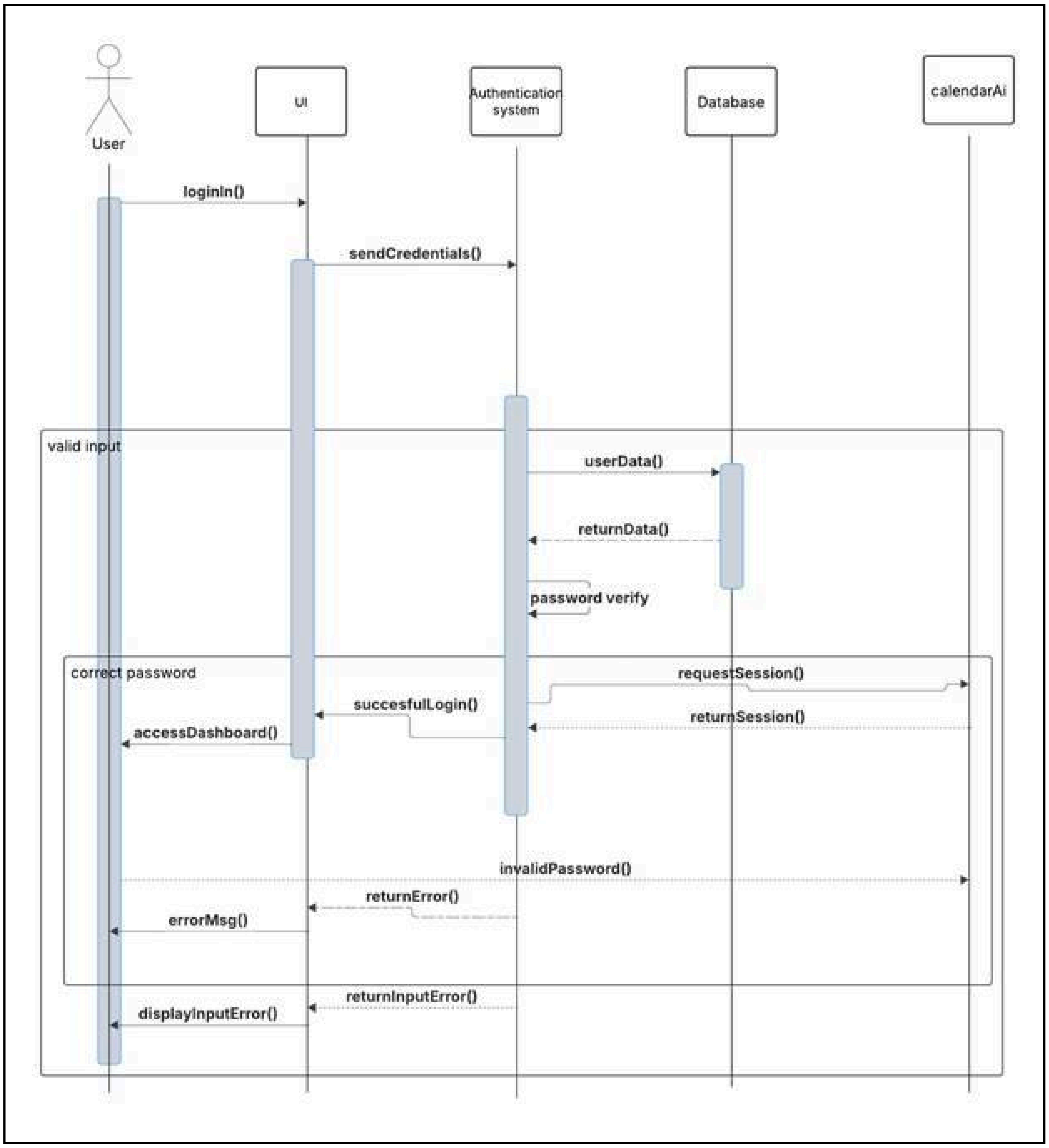


Figure 4: Login

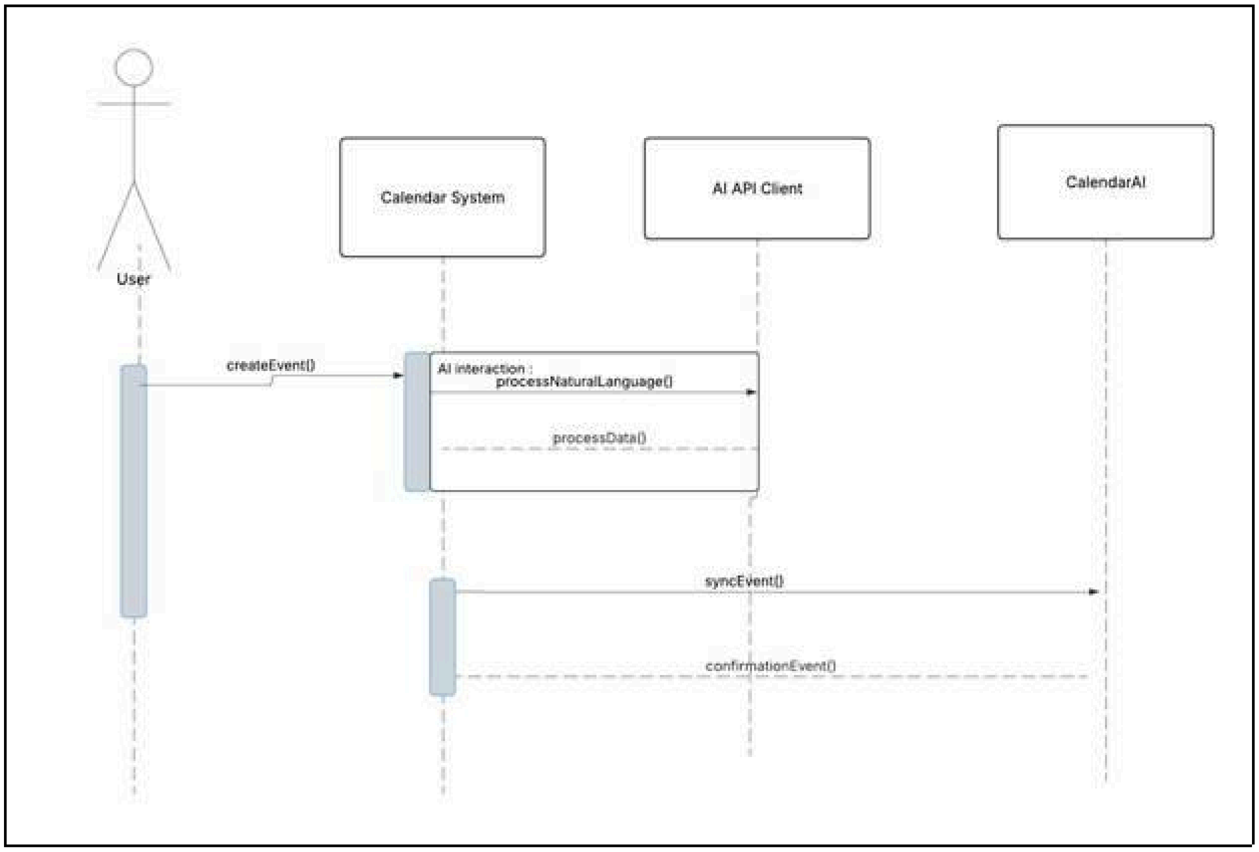


Figure 5: Create Event

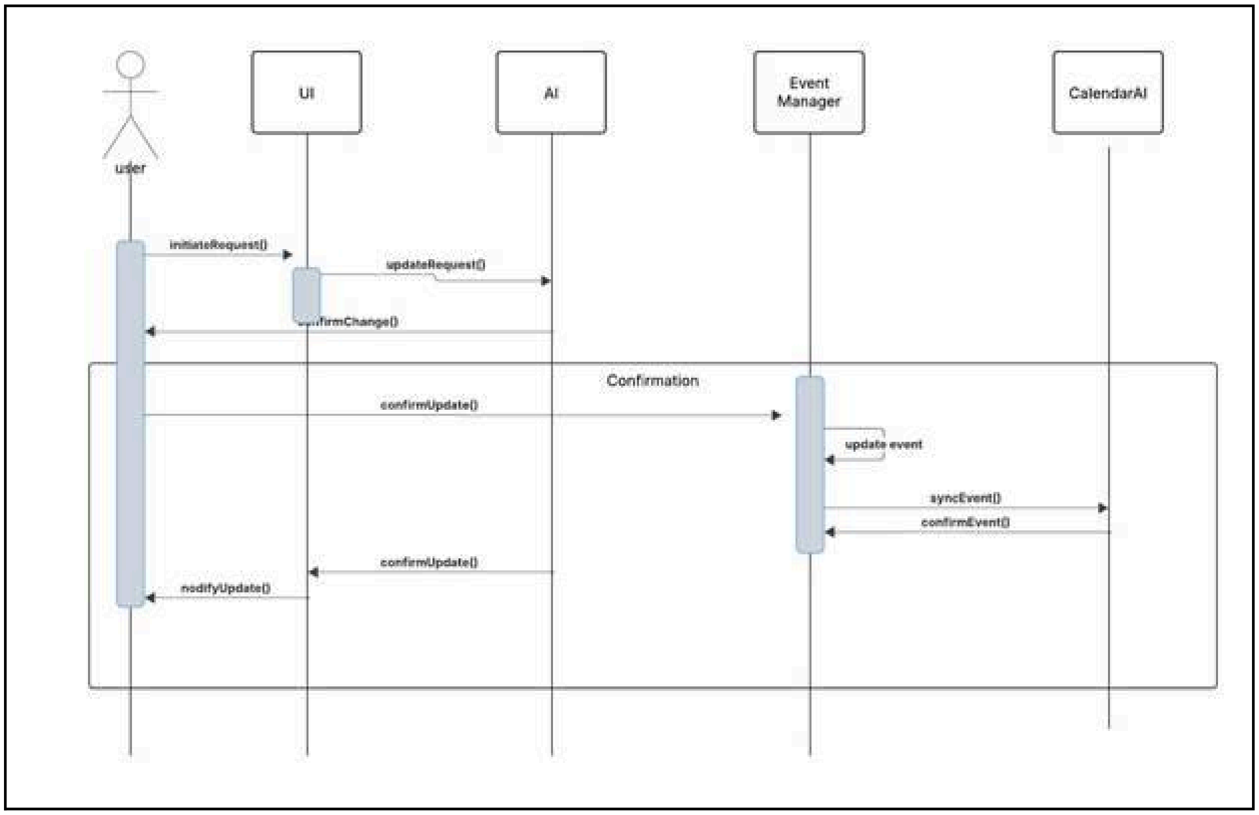


Figure 6: Update event

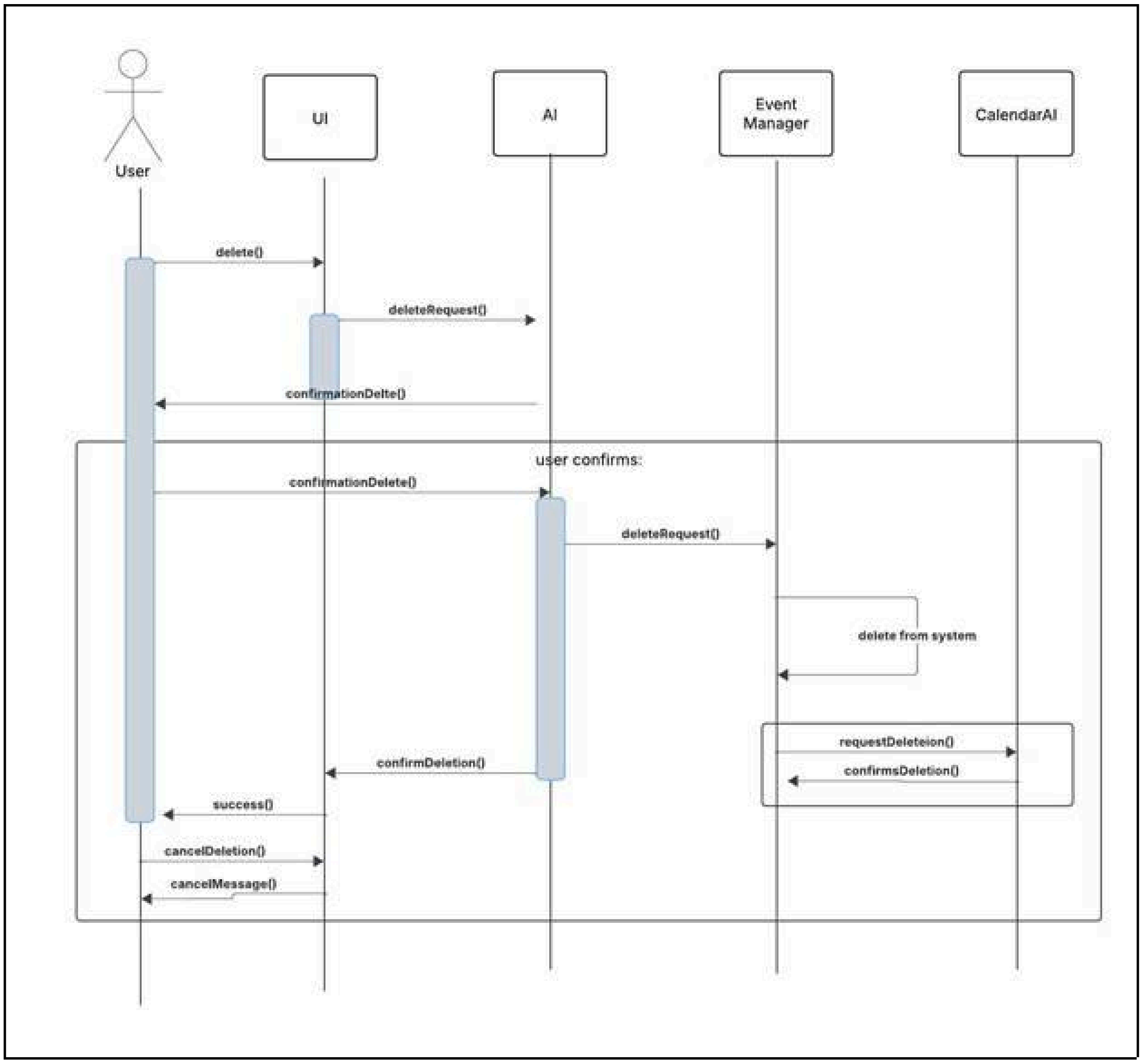


Figure 7: Delete Event

- 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.

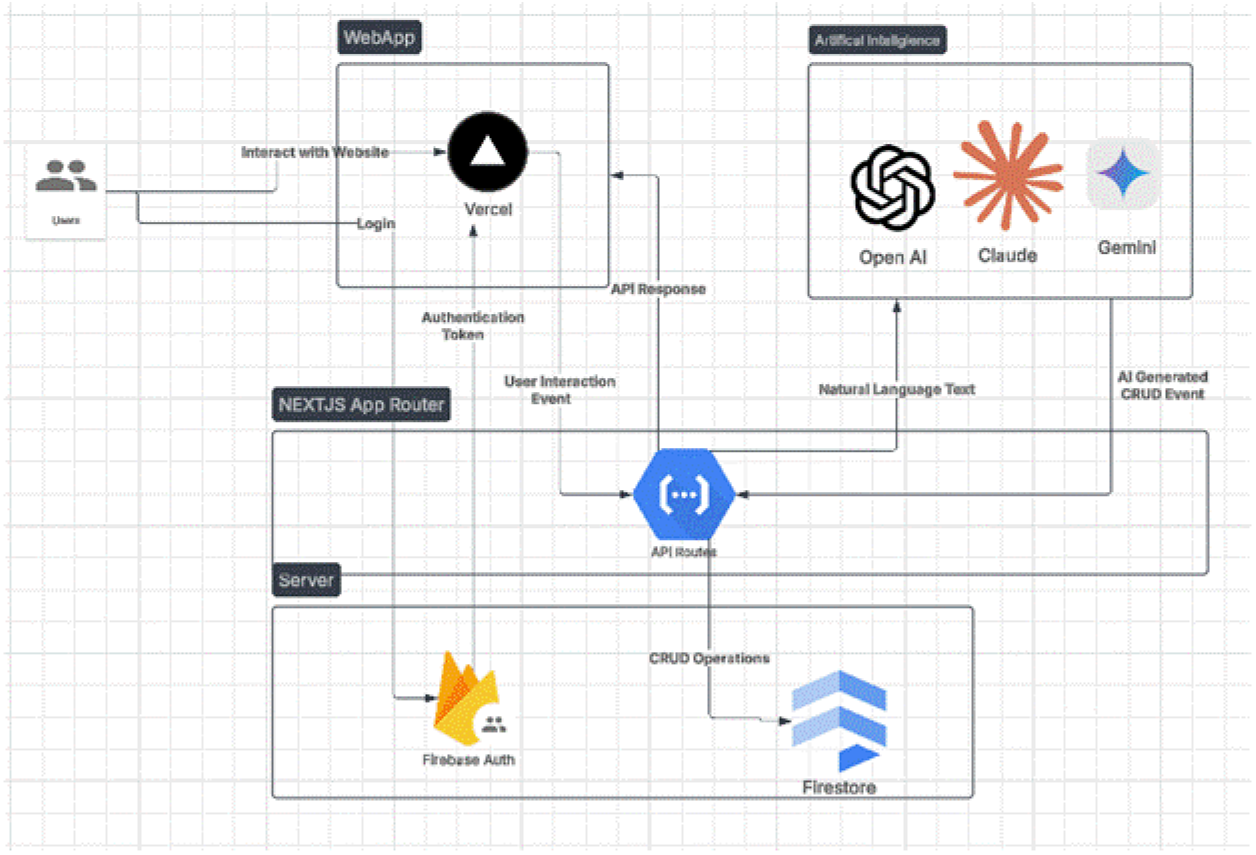


Figure 8: Architecture Diagram

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.

Project Scheduling, Cost, Effort and Pricing Estimation, Project

Project Scheduling

Our Project, CalendrAI is organized using the Scrum methodology and we have divided it into 6 sprints lasting 1 or 2 weeks. This takes into account meetings, designing, testing, developing, refactoring and any unexpected predicaments we may face.

Sprints

Sprint ID	Task Name	Start Date	Duration
S1	Core Calendar Functions	05/20/2025	2 weeks
S2	User Management/Auth + Advanced Functions	06/03/2025	2 weeks
S3	Google Calendar Sync	06/17/2025	1 week
S4	AI Scheduling Phase 1(basic	06/24/2025	2 weeks

	working functionality)		
S5	AI Scheduling Phase 2(testing and enhancements)	07/8/2025	2 weeks
S6	Final Testing and Fixes	07/22/2025	1 week

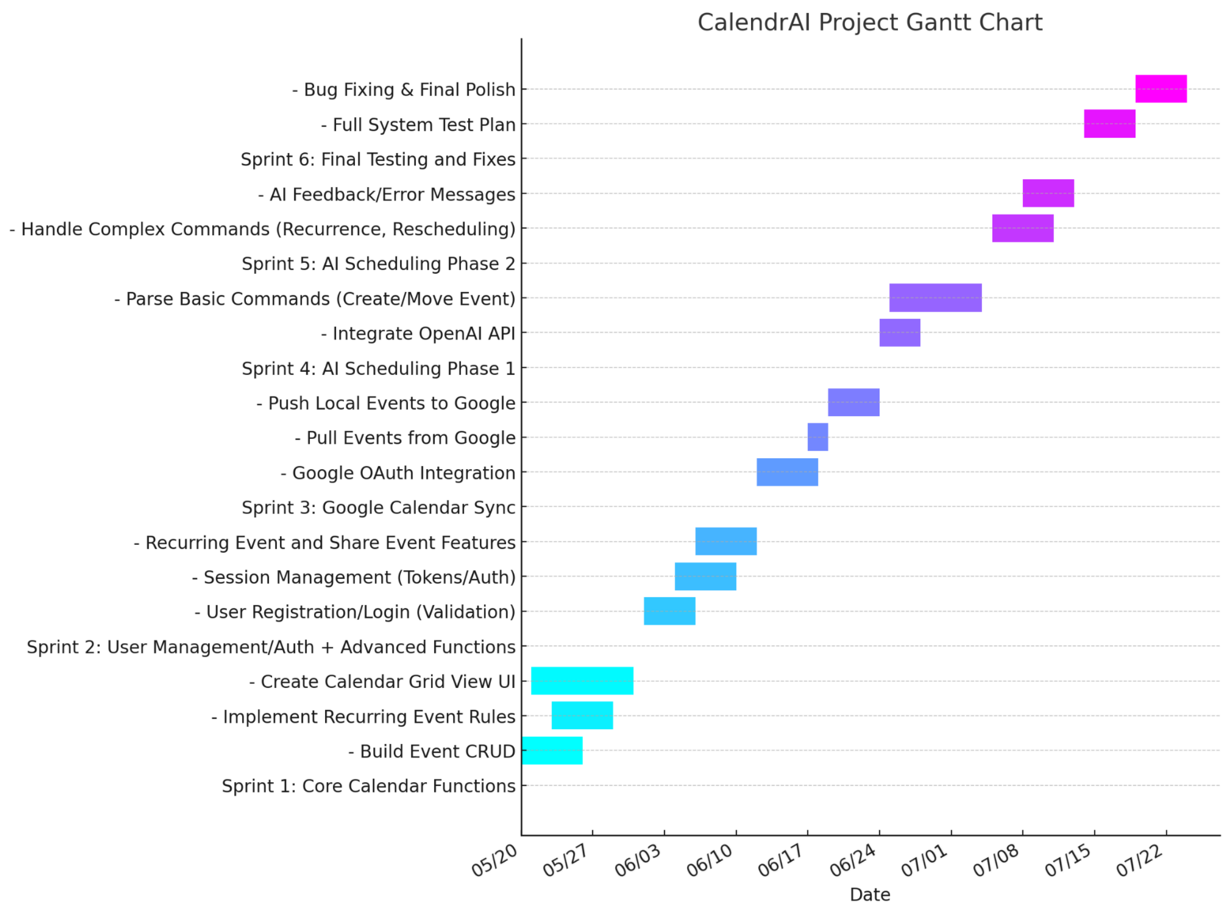
Duration and Staffing

Project Duration:
Start Date: May 20,2025

End Date: July 29, 2025

Gantt Chart

This Gantt Chart illustrates some of the tasks and subtasks in our schedule in order to complete this project.



Staffing

Our personnel will include 5 members assigned across different roles. We decided to just have a few roles since this is not a massive project and having too many will drop our efficiency.

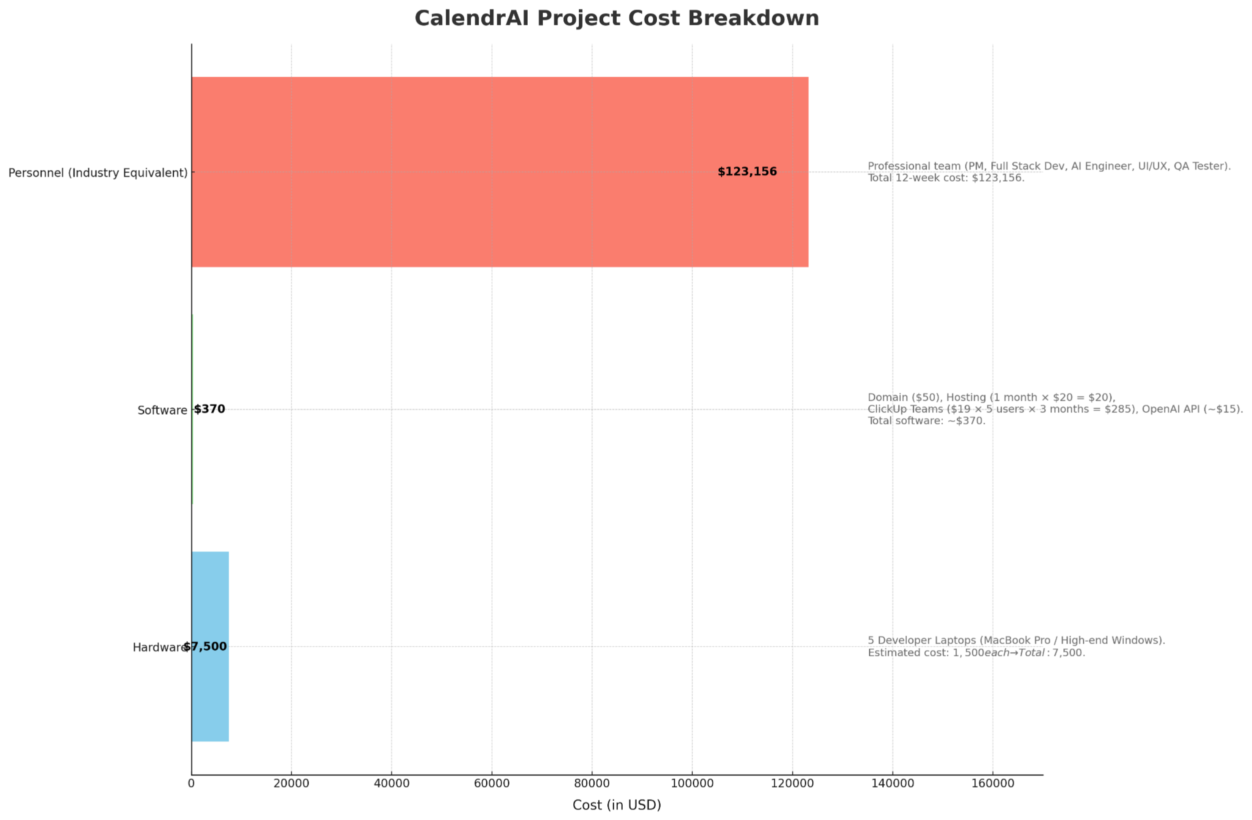
- **Project Manager -** Oversees the project, schedules meetings and make sure the team is on track to delivering on time
- **FullStack Developer** - Implements the frontend/backend, auth, APIs.
- **UX/UI Designer -** Focuses on the app design and user experiences, improving accessibility, usability, to keep users on our app engaged.
- **AI Integration Engineer** - Designs and helps to implement AI features into our app
- **QA Tester -** Verifies functionality, usability, finds bugs and suggests improvements

Cost Estimation

1. For a small startup, we will have modest salaries averaging around $24,600 a week per team member.
   a. **Total Personnel Cost:** $123,146
2. For security and compliance purposes, we will have to purchase quality developer laptops for all of our team members

    a. **Total Hardware Cost:** Macbook Pro/High-end Windows = 1,500*5 = $7,500

3. The software cost will be minimal for development as we will only have clickup to manage our team, OpenAI credits and hosting for deployment and testing.
    a. **Total Software Cost:** OpenAI Credits + ClickupTeams*5 + Any Hosting+Domain = 15+285+20+50 ≈ $370
4. **Total Project Cost:** <u>$131,016</u>



Role	Weekly Salary	10 Weeks Cost
Project Manager	$1,974	$23,688
Full Stack Developer	$2,370	$28,440
AI/Prompt Engineer	$2,045	$24,540
UI/UX Designer	$2,157	$25,884
QA Tester	$1,717	$20,604

Salary data retrieved from ZipRecruiter ("Project Manager Salary in Dallas, TX"; "Full Stack Developer Salary"; "AI Prompt Engineer Salary"; "UI/UX Designer Salary"; "QA Tester Salary in Dallas, TX"). See full citations in the Works Cited section.

Effort Estimation(Function Point)

Category	Simple	Average	Complex	Function Points (FP)	Notes
Inputs	0	7	2	40	Event CRUD, AI Input and processing, recurring events, etc.
Outputs	0	5	3	43	(Multiple Calendar Displays, Forms, AI Chat Recurring events display, Share Event confirmations)
Queries	0	3	0	12	Search, Filter, Categories
Files	2	1	3	69	(Calendar storage, Recurrence rules, Sharing settings, Credits)
External Interfaces	0	0	4	40	(Google Calendar push pull, API sync, OpenAI API)

Total GFP: 204 FP

PCA : 0.65+0.01(4 + 5 + 2 + 4 + 3 + 5 + 2 + 5 + 3 + 3 + 4 + 2 + 2 + 4) = **1.13**

FP: 204*1.2 = 244.8

E: 244.8/40 = 6.12 (7 person weeks for 40 FP per person week since our team is not that experienced)

Project Duration: 7 / 5 = 1.4 weeks.

This function point analysis estimates 1.4 weeks for pure implementation with our full effort of staffing. Our 10 week schedule accounts for a more realistic workflow with multiple iterations, design, testing, refactoring, problems, meetings and the fact that not all work can be done in parallel by all members meaning not our entire team will be working at maximum efficiency.

Pricing Estimation

Although our estimated internal development cost for CalendrAI comes out to be approximately **$131,016**, the final price that we will charge takes into account the potential project risks and uncertainties we may face along the way. This is important especially, given that the project involves third-party API's, and AI integrations that could introduce problems and unexpected costs as we develop and experiment with them. In other words, there is no guarantee that we can easily meet the requirements because of uncertainty.

To address these risks, we are applying a **20% risk adjustment(increased pricing)** over our base costs. This will ensure that we have the flexibility to handle any abrupt delays or problems we encounter while maintaining that we deliver the project with the utmost quality and timely delivery.

The final proposed price for CalendAI is **$157,219**

Name	**Amount**
Base Estimated Project Cost	$131,016
Risk Adjustment (20%)	$26,203
Final Proposed Price to Customer	**$157,219**

Software Testing

CalendrAi underwent multiple testing phases to guarantee user satisfaction, utility and stability.

3.1 Testing goals

- Verify correct functionality of CalendarAi for all features.
- Identify bugs early in development testing and address any software defects.
- Validate CalendarAi functions and performs well before release.
- Gather feedback from users for improvements.

3.2 Testing Methods

- Unit Testing

Unit testing focused on individual functions such as event creation and delete. Made sure each function was tested to ensure it completed its intended function without any issues.

Example:

Made sure the createEvent() function accurately saved an events title, date and time.

- System Testing

  Tested the full app from start to finish to make sure everything works together from a user's point of view.

  Example:

  Developed a process that included launching the app, using AI to schedule an event, and confirming that the event was appropriately saved and displayed.

- User Testing

  Test the user experience and functional utility of the app under real-life conditions.

  Example:

  The user was asked to use the AI features to set up a meeting and report any unpredicted behaviours or unclear actions.

3.3 Test Cases

Test Case ID	Description	Expected Results
TC1	Create new event	Event is added and appears to the calendar
TC2	Delete event	Selected event is removed from the calendar
TC3	Use AI to create meeting	AI suggests a valid time , without conflicts

3.4 Testing Schedule - Sprint based

Sprint	Date Range	Testing Type	Description
S1	May 20 - June 2	Unit Testing	Tested basic create and delete functions
S2	June 3- June 16	Unit Testing	Continued testing new features
S3	June 17 - July 2	System Testing	Tested how features work together
S4	July 3 - July 16	System Testing	Enhanced UI and tested AI time suggestion accuracy
S5	July 17 - July 30	User Testing	Gathered feedback from users to evaluate usability and identify issues
S6	July 31 - Aug 13	Final Testing and Fixes	Regression testing and final improvements

3. **Similar Design Comparisons**

The design of CalendarAI is distinct from other major calendar systems like Google Calendar, Outlook, and Apple Calendar because of its emphasis on AI-driven event and task creation. While traditional calendars such as the aforementioned rely on manual event creation, CalendarAI uses an intelligent, proactive chatbot that you can talk to in order to shape your schedule. This system is provided by a NLP (Natural Language Processing) system in order to understand user input and generate events.

Additionally, CalendarAI can deal with conflict resolution in schedules to dynamically change timings, acting as an assistant to the user. This is different from mainstream calendar systems, which rely on purely manually created events, whether they be scheduled or not.

	CalendarAI	Google Calendar	Outlook
Calendar Views	✅	✅	✅
Event Scheduling	AI-powered	Manual	Manual

Event Modification	AI-powered	Manual	Manual
AI Assistant	✅	❌	❌

Conclusion

First, we started out with a solid plan. We wanted to build a full calendar app with AI integration, Google Calendar syncing, and a clean design. We were able to finish most of what we planned, including all the core calendar features and even a basic AI event creator for the prototype. Throughout the project, we learned a lot about team management, and how to create developer- and client-focused diagrams like class diagrams, use case diagrams, and architecture diagrams. When creating the prototype, we ran into issues properly implementing the Google Calendar sync and OpenAI API integration. Looking back, we wish we had handled error cases better from the beginning instead of trying to brute force fixes later on.

As for project management, we started strong by using Scrum and tools like ClickUp to plan our sprints and delegate tasks. But as deadlines got tighter and schedules clashed, we found ourselves depending more on Discord for quick updates instead of sticking to our original planning system. That shift made things a bit unorganized at times. A major challenge was getting the entire team to work together consistently, since everyone had different class and work schedules. If we were to do it again, we'd try to set clearer deadlines and maybe check in more often as a group.

Overall, while the project didn't go exactly how we imagined, we're proud of what we accomplished. We built a working MVP of a smart calendar system, overcame some tough issues, and learned a lot about software planning, real teamwork, and adapting when things don't go as expected.

3. Presentation Slides (Including Prototype)

4. Repository Submission: https://github.com/omesh-s/CalendrAI

5. References (MLA or APA format)

Somerville, Ian. *Software Engineering.* 10th ed., Pearson, 2016.

Works Cited

"AI Prompt Engineer Salary." *ZipRecruiter*, www.ziprecruiter.com/Salaries/Ai-Prompt-Engineer-Salary. Accessed 29 Apr. 2025.

"Full Stack Developer Salary." *ZipRecruiter*, www.ziprecruiter.com/Salaries/Full-Stack-Developer-Salary. Accessed 29 Apr. 2025.

"Project Manager Salary in Dallas, TX." *ZipRecruiter*, www.ziprecruiter.com/Salaries/Project-Manager-Salary-in-Dallas%2CTX. Accessed 29 Apr. 2025.

"QA Tester Salary in Dallas, TX." *ZipRecruiter*, www.ziprecruiter.com/Salaries/Qa-Tester-Salary-in-Dallas%2CTX. Accessed 29 Apr. 2025.

"UI/UX Designer Salary." *ZipRecruiter*, www.ziprecruiter.com/Salaries/Ui-Ux-Designer-Salary. Accessed 29 Apr. 2025.