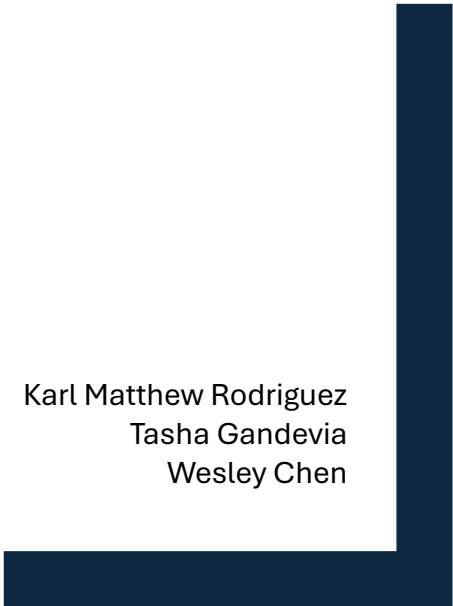




Group 20 - Waves

MILESTONE 1: STUDY BUDDY

Karl Matthew Rodriguez
Tasha Gandevia
Wesley Chen



APIs

The **Google Gemini API** allows users to give prompts and receive responses from the AI chatbot that powers Gemini. We will implement this API into our study app project for allowing the user to get auto-generated study material about their study topic to assist the user with their studies.

The **Google Calendar API** allows you to display your Google Calendar and as well as add or remove events from it. This API will be implemented into our project for allowing the user to use their calendar to keep track of upcoming events like exams, and to determine what to study for based on what's coming up in the calendar.

Features

For the **Google Gemini API**, we will implement features such as creating flash cards, creating quizzes, and asking questions. For the flash cards, the user can pick how many they want and this feature will be used for memorizing terms or vocabulary. Users can click on the flash card to reveal the answer. For the quiz feature, the user will receive a number of questions and after they answer all of them, the AI will give the answers along with the user's quiz score. The flash card and quiz generation features will benefit the user by allowing them to assess their knowledge of their study topic so they can find out how well they know the material. This will also give an indicator to the user if they should study more or if they know the material well. For the third feature that we will implement, we will allow users to ask questions to the study bot to receive answers to help them with studying.

For the **Google Calendar API**, the 3 features that we will implement are displaying events, like exams, to the user from their calendar, sending event reminders, and adding events to their calendar from the study app. For displaying events, the event to be displayed will be clearly labeled and marked on the study app's calendar. For the reminders feature, when an event is nearing, the user will receive a notification. For adding events, the user can input the name of the event, the event type, like midterm or quiz, the time, the course, and then add it to their calendar. These features all benefit the user by helping the user to be aware of upcoming deadlines or events so that they can adjust their study schedule around them.

Software Development Life Cycle

We chose the iterative approach. We chose this because our project's main requirements are well defined and there is not a need to be flexible. The waterfall approach would work, however, if little adjustments are to be added, we would have to restart, so an iterative approach would be more beneficial. Since our requirements are well defined, there shouldn't be a lot of iterations required for the product.

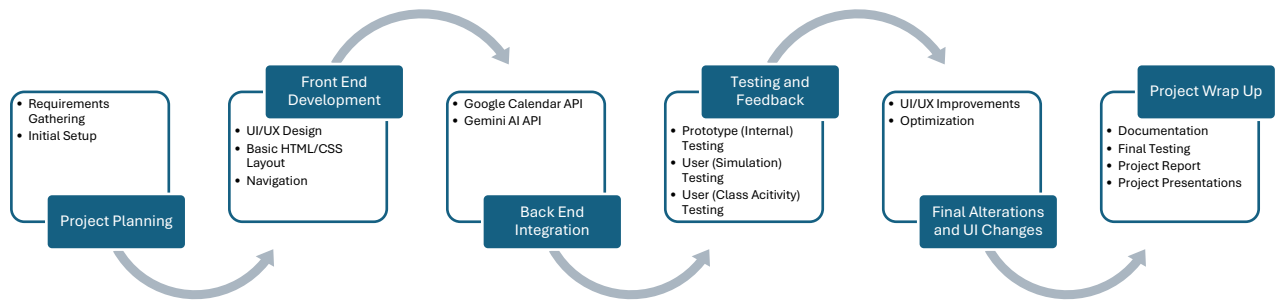
Milestones and Timeline

Milestones

1. Project Setup
 - a. Deliverables:
 - i. Define project requirements in detail
 - ii. Confirm API access and test initial API calls for Google Calendar and Gemini
2. Design and Implement Basic UI
 - a. Deliverables:
 - i. Design wireframes for main screens
 - ii. Implement basic HTML/CSS layout to create placeholders for key features.
 - iii. Set up navigation between pages to create a functional flow.
3. Google Calendar API Integration
 - a. Deliverables:
 - i. Sync user calendar events from Google Calendar.
 - ii. Implement reminders for deadlines and display current time.
 - iii. Enable users to add and edit events through the app.
 - iv. Implement timers (pomodoro, stopwatch, countdown)
4. Gemini API Integration
 - a. Deliverables:
 - i. Implement auto-generated flashcard creation
 - ii. Integrate the quiz generation tool, allowing users to customize topics and difficulty.
 - iii. Add the question-answer and hints feature for real-time homework help.
5. User Testing and Feedback
 - a. Deliverables:
 - i. Test the app in class activity
 - ii. Gather feedback on usability, design, functionality
 - iii. Identify and prioritize all bugs and issues reported.
6. Finalize UI/UX and Optimize Functionality
 - a. Deliverables:
 - i. Apply any changes based on user feedback.
 - ii. Polish UI with additional styling for a more user-friendly experience
 - iii. Ensure each API feature works smoothly
7. Final Testing and Documentation
 - a. Deliverables:
 - i. Perform final testing
 - ii. Write user manual / guide to add to app
 - iii. Document the code

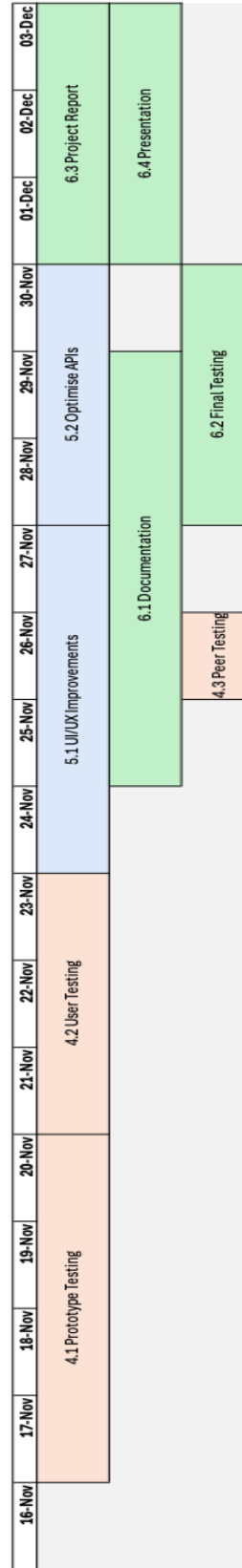
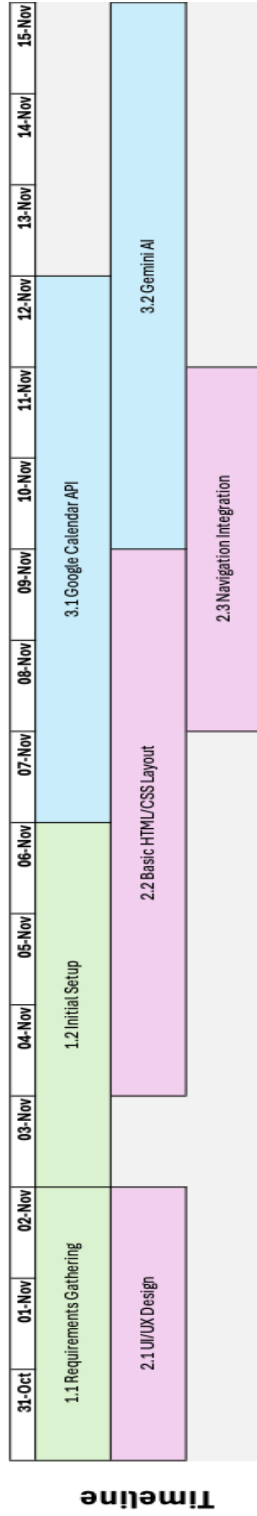
Work Breakdown Structure (WBS)

High Level



Please see appendices for low-level breakdown of above

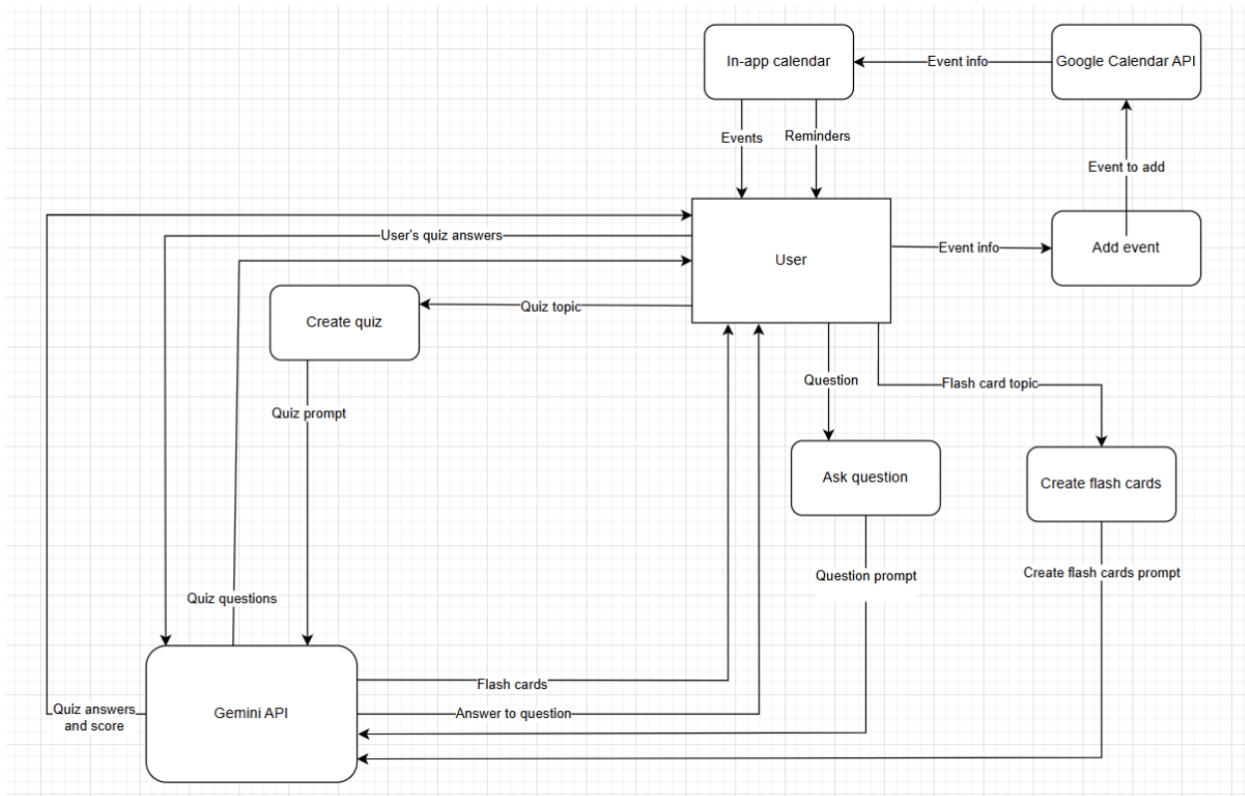
Timeline



Risks and Risk Mitigation

Risk	Risk Rating	Mitigation Strategy
Inconsistent UI on different browsers	Low	Test cross-browser early and adjust CSS as needed.
Minor API latency issues	Low	Implement loading indicators for users and consider caching frequent API calls
Miscommunication within the team	Low	Schedule regular check-ins and use task management systems
Minor bugs in navigation	Low	Conduct thorough testing of navigation during front-end development and ensure logical flow
Minor issues with UI aesthetics	Low	Conduct regular design reviews. Use style guide to ensure uniformity in colours, fonts, button styling
Minor inconsistency in icon sizes	Low	Use design checklist for consistent icon sizes and spacing. Regularly review UI for alignment and visual balance
Compatibility issues with external libraries	Medium	Test libraries in isolation before full integration and check documentation for compatibility
Overlapping user feedback requirements	Medium	Prioritise feedback based on user needs and project goals
API rate limit exceeded	Medium	Use efficient API calls and implement caching for frequent calls
Delays in gathering user feedback	High	Schedule regular feedback sessions within the group, allow time for adjustments
Major bug in calendar syncing	High	Prioritise testing for the Google Calendar integration and set up a local storage solution for temp use if API fails

Data Flow Diagram



For the **Google Calendar API**, the user can add an event by providing the event info which then goes through the calendar API to add the event. The event is then updated in the in-app calendar which is displayed to the user. Reminders about the event will also be delivered to the user based on the calendar's events.

For the **Gemini API**, the user can ask a question or create flash cards, and a prompt will be given to the AI depending on what option the user chooses. The AI will generate a response, and the response will be displayed to the user. For the quiz feature, the user will give the quiz topic, and the AI will create quiz questions about that topic. The user will answer the questions and then submit their answers to the AI. The AI will then display the correct answers along with the user's quiz score.

Model Viewer Controller (MVC)

Home Page

Model	View	Controller
<ul style="list-style-type: none">• Get song• Get AI response• Send AI messages• Get Calendar events• Switch to Calendar• Switch to practice	<ul style="list-style-type: none">• To Do List• Timer• Song • Gemini AI	<ul style="list-style-type: none">• Toggle To Do• Toggle Timer• Toggle Song• Toggle AI• Change to Calendar View• Change to Practice• Message Gemini AI• Pause Timer• Start Timer• Pause/Start Song• Next Song• Previous Song

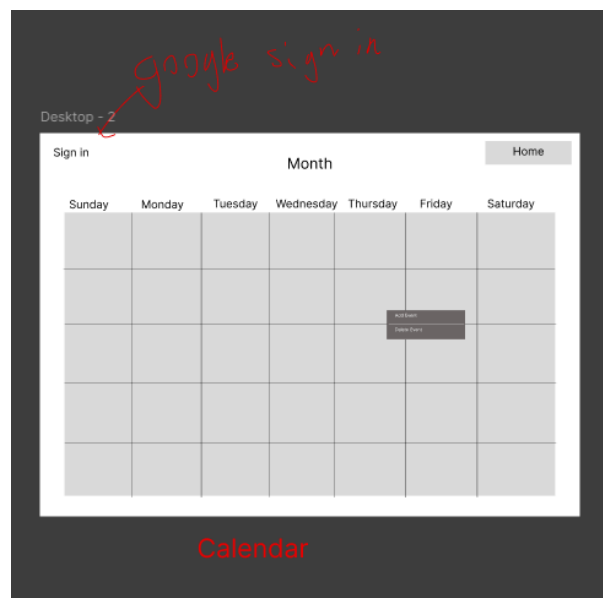
Calendar Page

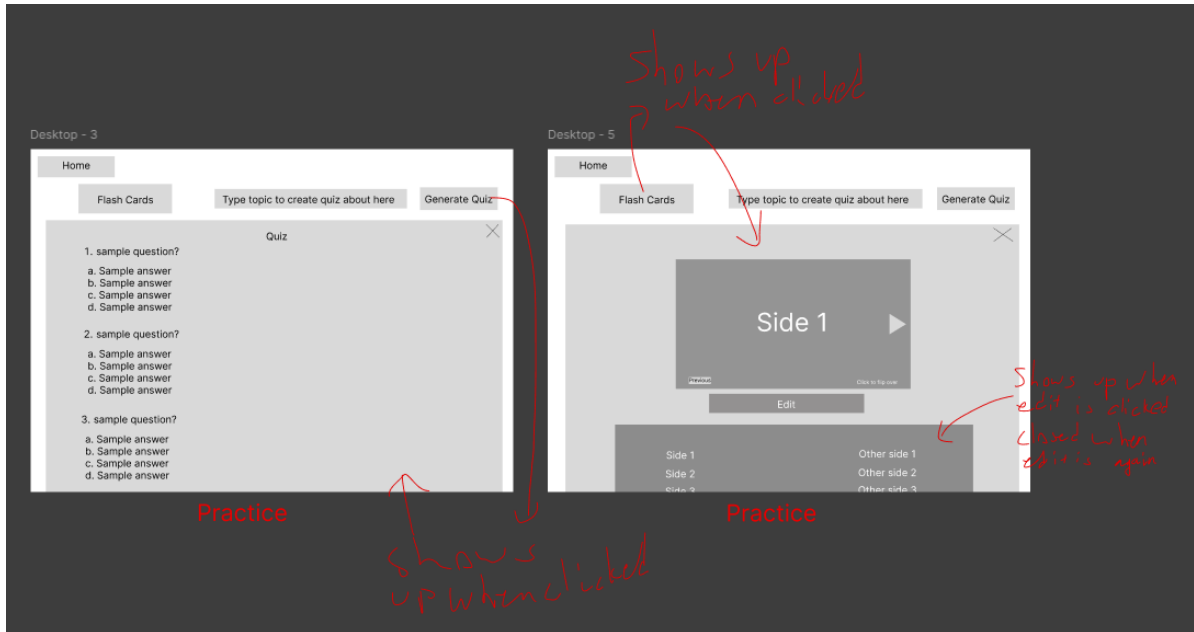
Model	View	Controller
<ul style="list-style-type: none">• Remove Event• Add Event• Get Events	<ul style="list-style-type: none">• Events• Sign In/Out	<ul style="list-style-type: none">• Change Home• Sign In• Add Events• Delete Events

Practice Page

Model	View	Controller
<ul style="list-style-type: none">• Get Flashcards• Make Quiz• Add FlashcardRemove Flashcard	<ul style="list-style-type: none">• Quiz• Flashcards• Quiz Prompt	<ul style="list-style-type: none">• Change Home• Make Quiz• Close Quiz• Show Flashcards• Close Flashcards• Change Text• Next Card• Previous CardFlip Card• Get Hint

Medium Fidelity Prototypes





Appendices

Low Level Work Breakdown Structure (WBS)

1. Project Planning

- 1.1 Requirements Gathering
 - 1.1.1 Identify main goals and objectives of the app
 - 1.1.2 Define features needed from Google Calendar and Gemini API's
 - 1.1.3 Determine user requirements based on personas
- 1.2 Initial Setup
 - 1.2.1 Set up the development environment
 - 1.2.2 Obtain and store API credentials securely
 - 1.2.3 Confirm initial access to Google Calendar and Gemini APIs with test calling

2. Front End Development

- 2.1 UI/UX Design
 - 2.1.1 Plan wireframes for each screen
 - 2.1.2 Design detailed mock-ups for main views
 - 2.1.3 Decide on colour schemes, fonts, button styles, attention grabbers
 - 2.1.4 Team feedback session to finalize designs
- 2.2 Basic HTML/CSS layout
 - 2.2.1 Set up HTML structure and formatting of homepage, calendar, tools, timers
 - 2.2.2 Apply CSS for base layout and interactions
 - 2.2.3 Create headers, footers, navigation bar
 - 2.2.4 Test layout on different screen sizes (desktop, mobile)
- 2.3 Navigation
 - 2.3.1 Implement links and buttons for screen switching
 - 2.3.2 Create feedback mechanism for easy navigation
 - 2.3.3 Test navigation functionality across all pages

3. Backend Integration

- 3.1 Google Calendar API
 - 3.1.1 Event Syncing
 - 3.1.1.1 Set up API action to get events from user's calendar
 - 3.1.1.2 Display events on the Calendar UI
 - 3.1.1.3 Add sorting/filtering options for events
 - 3.1.1.4 Test event accuracy and speed
 - 3.1.2 Reminders
 - 3.1.2.1 Set up reminder scheduling with Google Calendar API
 - 3.1.2.2 Integrate reminders into app notification system
 - 3.1.2.3 Allow users to customize reminder settings on frequency etc
 - 3.1.2.4 Test reminder timing and functionality
 - 3.1.3 Event Creation
 - 3.1.3.1 Implement UI for adding new events
 - 3.1.3.2 Set up API call to save events to Google Calendar

- 3.1.3.3 Event data validation before saving (everything filled out, accurate date format etc)
- 3.1.3.4 Confirm events are added correctly on the calendar
- 3.2 Gemini API Integration
- 3.2.1 Flashcard Generations
- 3.2.1.1 Set up API call to generate flashcards based on input text
- 3.2.1.2 Create UI to display flashcards for review
- 3.2.1.3 Add options for categorizing flashcards
- 3.2.1.4 Test flashcard accuracy, relevancy, speed
- 3.2.2 Quiz Generation
- 3.2.2.1 Implement API call to generate quizzes with specific topics
- 3.2.2.2 Allow user customization for quiz difficulty and number of questions
- 3.2.2.3 Design UI for displaying/interacting with quizzes and tracking scores
- 3.2.2.4 Test quiz generation for quality, relevancy, accuracy
- 3.2.3 Question Functionality
- 3.2.3.1 Implement text input for users to ask questions
- 3.2.3.2 Set up API call to return responses
- 3.2.3.3 Display answers in user friendly formatting
- 3.2.3.4 Test for accuracy and response time

4. User Testing and Feedback

- 4.1. Prototype Testing
- 4.1.1 Test basic app functionality within project group
- 4.1.2 Identify bugs or UX issues
- 4.1.3 Document issues and prioritize fixes
- 4.2 User Testing
- 4.2.1 Mimic user tests based on personas
- 4.2.2 Identify issues
- 4.2.3 Document issues and prioritize fixes
- 4.3 Class Testing
- 4.3.1 External testing within the class activity setting
- 4.3.2 Observe interactions and note possible changes
- 4.3.3 Identify bugs or UX issues
- 4.3.4 Document issues and prioritize fixes
- 4.3.5 Implement fixes

5. Final Alterations, UI Changes

- 5.1 UI/UX Improvements
- 5.1.1 Refine layout and style based on feedback
- 5.1.2 Improve accessibility
- 5.1.3 Add final design elements (icons, images, animations, favicon)
- 5.2 Optimize API's
- 5.2.1 Identify redundancy or inefficient API calls
- 5.2.2 Implement caching where needed for reducing load times
- 5.2.3 Test API calls for timing
- 5.2.4 Document optimizations for referencing

6. Project Wrap Up

- 6.1 Documentation
 - 6.1.1 Write instructions for API setup and usage
 - 6.1.2 Create user manual/quick start guide for application
 - 6.1.3 Document code
 - 6.1.4 Prepare project readme file with setup instructions
- 6.2 Final testing
 - 6.2.1 Full run-through of all features and functionalities from fresh application
 - 6.2.2 Verify all APIs are functioning as intended
 - 6.2.3 Document and fix any remaining bugs
 - 6.2.4 Perform final checks on UI
- 6.3 Project Report
 - 6.3.1 Analysis of project's success to solve problem
 - 6.3.2 Analysis of project's SDLC model
 - 6.3.3 Detailed description of features implemented for each API
 - 6.3.4 Detail description and overview of CI/CD pipeline and use
 - 6.3.5 Detailed diagram of project's architecture
 - 6.3.6 List of known bugs and issues with project and severity
 - 6.3.7 Lessons learned and project takeaways
- 6.4 Presentation
 - 6.4.1 Overview of project and problem
 - 6.4.2 Chosen API's and features implemented
 - 6.4.3 Overview of CI/CD pipeline use
 - 6.4.4 Lessons learned
 - 6.4.5 Project demo video
 - 6.4.6 Potential improvements