



StudySync AI

StudySync AI

Group Members

Uttam Sharma (usa10@sfu.ca)

Simone Motwani (svm4@sfu.ca)

Toma Ozarchevici (toa17@sfu.ca)

Chance Wijewardena (cwijewar@sfu.ca)

[GitHub Repository Link](#)

[StudySync AI Website](#)



1. Analysis of Project Success

Throughout the development of StudySync AI, we incorporated the feedback from users through multiple rounds of feedback we received from milestone reviews, TA, and the peer testing session. This process ensured that the app's features and interface evolved based on real-world usage.

Milestone 1 - Project Planning Feedback

During Milestone 1, peer feedback and TA reviews primarily helped us clarify and narrow the project scope to ensure feasibility and focus. While detailed UI feedback was limited at this stage, the feedback emphasized: defining a clear set of core features, confirming the two main APIs (Google Calendar and Gemini AI) to be used, and detailing how they support these features, prioritizing features within our Work Breakdown Structure. This feedback helped us focus our project goals and avoid overextending.

Peer Testing Feedback

In the peer testing session, testers completed specific tests without additional guidance, providing critical feedback about project usability. Our peers noted that our UI looked visually appealing and well-organized, but provided two key improvement areas: button label clarity and usability guidance. For button clarity, some buttons were not immediately clear about their function; we addressed this by using more descriptive text labels. For usability guidance, testers suggested making it more user-friendly and providing more instructions for users; in response, we added instructions to the users on the calendar page.

Additionally, many testers expressed overall satisfaction with the interface design and smooth animations, noting that the visual flow and transitions made the application feel polished and professional. This positive feedback reinforced that our design choices were effective in creating an engaging user experience.



2. Software Development Life Cycle (SDLC) Analysis

We chose the Agile Kanban model as our SDLC approach and used GitHub Projects' Kanban board to manage our workflow. All tasks were added as GitHub Issues and kept in the backlog. For each sprint, we moved 8 cards into the “Active Sprint” column so everyone knew what to focus on and nobody felt overwhelmed. Usually, each person worked on around 2 cards at a time, and once those were done, we pulled new ones from the backlog. Sometimes a few cards stayed in review longer because people were still testing or making small edits, but that didn't slow us down much. We followed this model throughout the project and didn't make any changes to our process.

Using Kanban gave us a lot of flexibility. It was easy to track who was doing what, what was left to do, and it made dividing tasks simple. The layout helped make sure no one had too much on their plate. If we saw that a card needed more work, we just moved it back to “In Progress.” Overall, this system worked really well for our team; it helped us stay organized, manage our time better, and keep everything running smoothly throughout the project.

3. API Feature Implementation

a. AI-Powered Calendar Event Creation / Conflict Checks / Multi-Schedule Planning

Description

Users can add events by typing natural language requests like “add meeting for 25 August at 3 PM.” The system uses AI to interpret and create structured calendar events. It also checks for conflicts across multiple calendars and offers suggestions to avoid overlap, helping users manage academic, personal, and work commitments in one place.

Implementation Details

- Natural language processing is used to parse and convert user input into calendar events.
- The system cross-references all connected calendars before scheduling.
- Suggests alternate times if conflicts are detected.



- Supports recurring events and color-coded categorization.

API Used

- Google Gemini

Changes Since Planning

- Expanded support for recurring events and more complex natural language commands.
- Enhanced conflict detection logic for faster and more accurate scheduling.

b. Secure Sign-In / Secure Sign-Out

Description

This feature provides users with a secure and seamless authentication process. Users can sign in using their institutional Single Sign-On (SSO) or by creating an account with email credentials. All sign-in data is encrypted and managed securely to ensure the privacy and protection of user data, especially sensitive calendar events and study information.

Implementation Details

- Users choose between SSO and standard email/password sign-up.
- Session tokens are securely stored using industry-standard methods.
- Auto sign-out is triggered after periods of inactivity.
- Ensures protection of synced data with backend Firebase authentication.

API Used

- Google Identity API from Google Calendar

Changes Since Planning

- Strengthened authentication by enabling email verification and 2FA compatibility.
- Improved UI flow for smoother sign-in/sign-out experiences.



c. Import Schedules from Canvas and SFU

Description

This feature pulls academic data from external educational platforms like Canvas and SFU. It automatically imports critical academic dates such as assignment deadlines, midterms, finals, and class schedules. This helps reduce manual entry and ensures users always have an up-to-date academic calendar.

Implementation Details

- Connects to the user's Canvas and SFU accounts using API tokens.
- Extracts iCal data and maps it to the StudySync calendar.
- Automatically updates when new events are added on the connected platforms.
- Events are categorized and tagged for clarity.

APIs Used

- Canvas LMS API

Changes Since Planning

- Added automatic syncing to check for new course events weekly.
- Introduced a visual tagging system (assignments, exams, lectures) to distinguish imported events.

d. Progress Analytics Dashboard

Description

This dashboard gives a visual summary of upcoming tasks, including goals and events. The dashboard also gives access to schedule, goals, notes, and suggestions.

Implementation Details

- Includes a dynamic checklist for goal tracking.
- Tracks both user-initiated and AI-generated study sessions.
- Syncs with the timer and notifications to show completed vs. missed sessions.



API Used

- Google Calendar

Changes Since Planning

- Smoother UI animations
- Widgets connected to real data from context files
- Button for notes widget that takes you to the notes page
- Streamlined icons from lucide-react library

e. Smart Notification System / Task and Reminder Creation

Description

This system will deliver intelligent, personalized study reminders by analyzing calendar activity, time of day, and user behavior. Currently, the notification system is functional with the Google Calendar API; however, the following features still need to be fleshed out: Instead of fixed alerts, users can input tasks in natural language (e.g., “review notes before lab tomorrow”). The system schedules them at optimal times for productivity. A study reminder card appears on the Settings page, prompting users to start study sessions. Once activated, a timer is displayed in the navigation bar to track the session in real time.

Implementation Details (current features)

- Notifications are controlled through email toggles, giving users full autonomy.
- The reminder card serves as both a prompt and a control panel for starting the timer.
- The timer is embedded in the navigation bar and visible across all app pages.

API Used

- Google Calendar

Changes Since Planning

- Introduced manual control for email notification toggles.



- Added a study timer that allows users to select a duration, start the timer from the reminder card, and view it across all tabs.

4. CI/CD Pipeline Overview

Overview of CI/CD Pipeline

Our project used a dual-pipeline CI/CD system with GitHub Actions, setting up separate automated workflows for the backend and frontend. The frontend pipeline was triggered by pushes to the main branch and pull requests targeting main. For the backend, we set up a smarter system where the pipeline only ran if there were changes inside the backend folder. This helped save resources and avoided unnecessary runs. We also made sure to keep things secure by using GitHub Secrets to store sensitive info like JWT secrets, Google OAuth tokens, and API keys, so everything stayed protected during build and deployment.

The frontend workflow started by setting up the environment using Ubuntu runners with Node.js 18. Then it installed dependencies and ran ESLint for code quality. We also added test suites that included health checks and navigation validation. After that, it built the frontend with secure environment variables and automatically deployed it to Netlify. The backend pipeline was more complex. It started by running the actual backend server and doing live health checks on localhost. Then it used Docker to build and test the full containers to make sure everything was deployment-ready.

Testing, Deploying, and Monitoring

Our CI/CD Pipeline was designed to streamline deployment and ensure continuous delivery of both the frontend and backend components of StudySync AI. The frontend was deployed on Netlify with automatic GitHub workflows that built and deployed our project using Vite, allowing every push to the main branch to trigger a build and deployment process without manual intervention.

For the Backend, we implemented a Dockerized environment using Docker Compose, enabling consistent deployment across environments and simplifying service management. The backend services were hosted on Uttam's home



servers, providing full control and freedom over runtime configurations and resource allocation.

We used Proxmox hypervisor for operating system deployment and monitoring, and Portainer for Docker container management and monitoring, allowing the team to track container health and logs through a user-friendly dashboard. Nginx was configured to process and route network traffic for the backend efficiently, as well as to provide an additional monitoring layer for incoming requests and service performance. From the beginning, our group prioritized scalability and the ability to give full visibility on the application's operational state.

5. Testing Strategy and Implementation

From early stages of development, we established a comprehensive testing strategy to ensure functionality. Our approach combined edge case handling, user feedback, and performance validation.

On the front end, we implemented both static analysis and functional testing. ESLint was integrated into the CI/CD pipeline to enforce code quality and catch early issues. During development, we manually tested navigation flows, UI responsiveness, and form validation, which covered common edge cases such as invalid inputs, empty submissions, and conflicting user actions. These tests ensured that users received helpful feedback and were prevented from making invalid choices, especially in sensitive features like the calendar scheduling using Gemini.

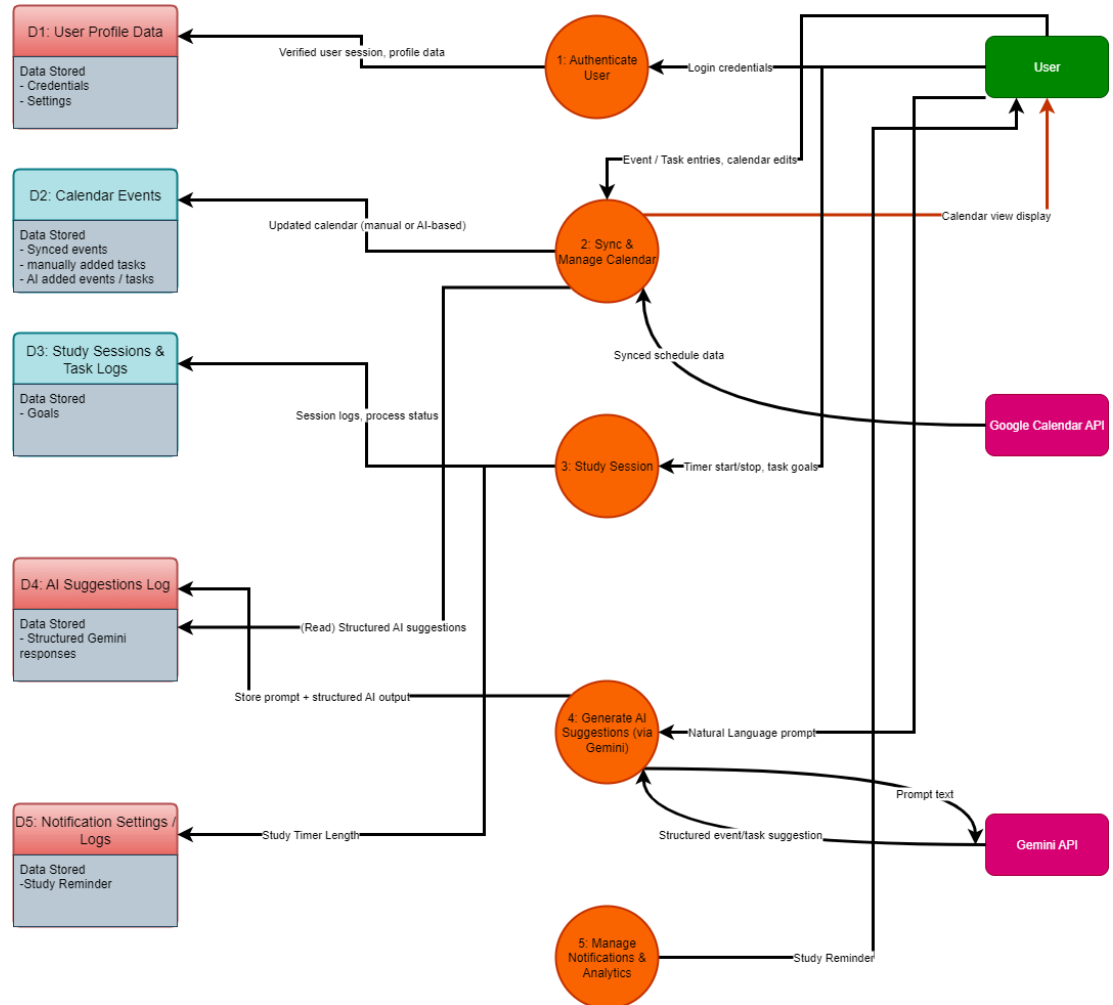
For the backend, we conducted targeted testing of our API endpoints using local scripts and live health checks without our GitHub Actions pipeline. This included verifying data validation logic, JWT-based authentication, and error handling under both normal and incorrect usage conditions. We also simulated multiple connections and requests to the remote server by monitoring nginx logs and docker container health using Portainer.

Testing outcomes were reviewed by the team during our development sprints and translated into actionable improvements logged as GitHub issues. During feature development, we relied heavily on local testing in our development environments; this included using console logging and error tracing to verify functionality, track down bugs, and confirm correct data flow between components. This way of development allowed us to address edge cases early, before code was



committed. Once changes were pushed to GitHub, our CI/CD pipeline ran automated workflows to build and test the application. This provided another checkpoint to detect broken builds or integration issues before deployment. This iterative process helped us catch bugs early, refine user experience, and maintain deployment stability as we pushed new features.

6. Updated Level 1 Data Flow Diagram (DFD)



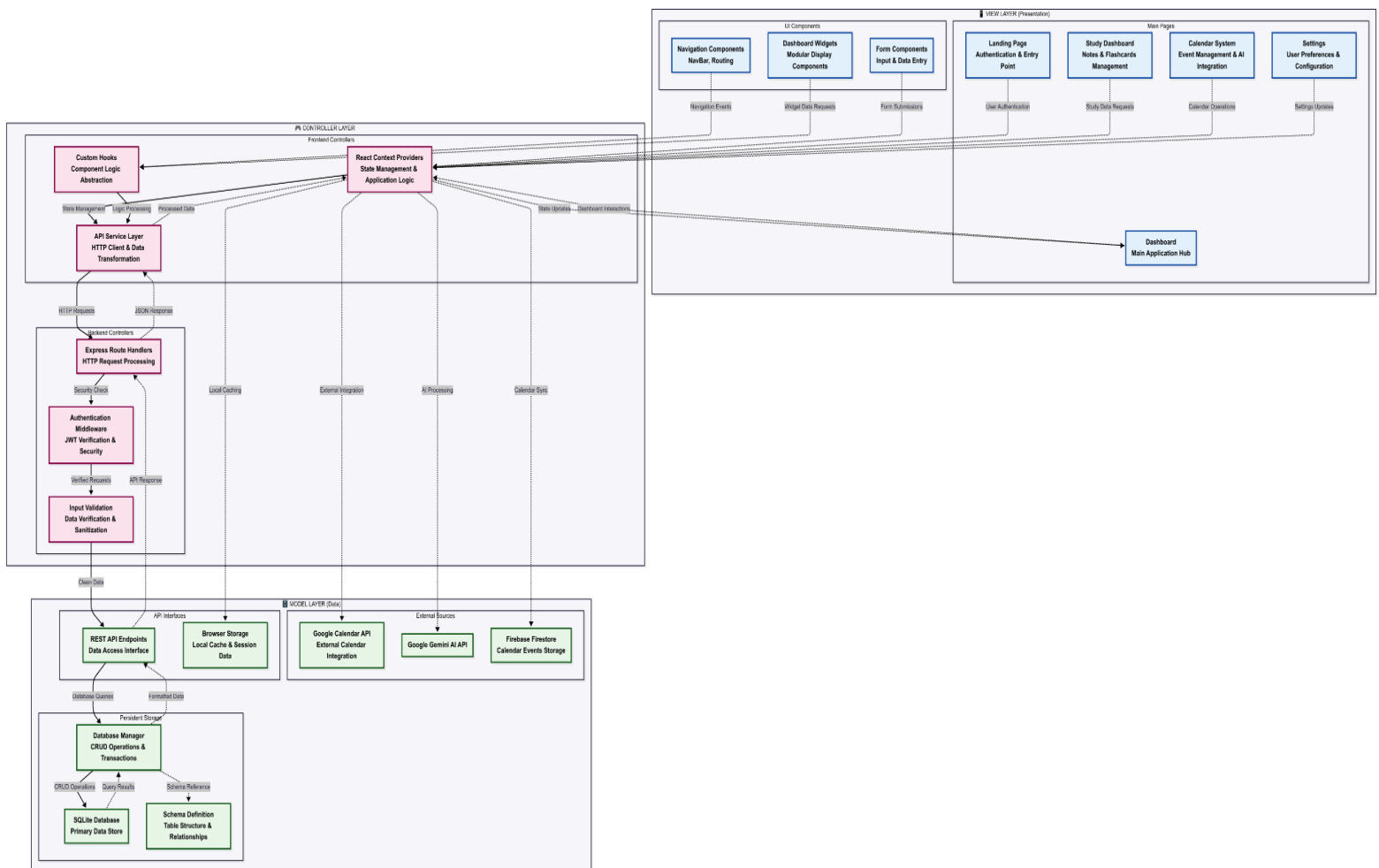
Component Breakdown:

- Right
 - External Entity
- Middle
 - Processes
- Left
 - Data Store
 - Red = Backend store



- Blue = Front-end store
- Changes made from the previous milestone
 - Removed connection from process Manage Notification & Analytics to Notification Settings / Log Data store
 - Connection was not implemented
 - Removed some data stores as some were not implemented

7. Updated MVC Model



Component Breakdown:

Model

- SQL Database for persistent storage

- Database Manager providing CRUD operations
- Schema definitions maintain data structures
- Google Calendar API for external integration
- Google Gemini API for AI-powered calendar actions
- Rest API endpoints providing standardized data access
- Browser storage for local caching

View

- React pages: Landing, Dashboard, Study Dashboard, Calendar, Settings
- Reusable UI components: Navigation, Dashboard Widgets, Forms

Controller

- React context providers manage state and application logic
- Custom hooks for component logic
- API services handling HTTP communication
- Express route handlers
- JWT authentication middleware

8. Bug Report Table

BUG	STEPS TO REPRODUCE	SEVERITY	GITHUB ISSUE
Inconsistent UI sizing for the website pages across different devices	Loading the website on other devices results in needing to scroll to see the rest of the page	Moderate	https://github.com/C-MPT-276-SUMMER-2025/final-project-24-horizons/issues/62
If a reply by Gemini is very, very big, then it goes out of Gemini's dedicated chat bubble	In the case that Gemini gets overloaded, it replies with a long error code, which goes out of the Gemini chat bubble	Moderate	https://github.com/C-MPT-276-SUMMER-2025/final-project-24-horizons/issues/64
Gemini recognizes tomorrow but adds an event for the day after	Try prompting to add an event tomorrow at 2 pm	Moderate	https://github.com/C-MPT-276-SUMMER-2025/final-project-24-horizons/issues/67



Settings page cards don't fit	Resize the cards in settings	Minor	https://github.com/C-MPT-276-SUMMER-2025/final-project-24-horizons/issues/60
Notification Toggle buttons have no functionality	Implement the notification functions	Severe	https://github.com/C-MPT-276-SUMMER-2025/final-project-24-horizons/issues/61
AI suggestions are not being pulled from Gemini	On the main dashboard, Gemini is not autopopulating the suggestions.	Severe	https://github.com/C-MPT-276-SUMMER-2025/final-project-24-horizons/issues/68
When a user refreshes, they get an error for "Page not found" from netlify	Go to any page, for example the dashboard, and refresh the page.	Severe	https://github.com/C-MPT-276-SUMMER-2025/final-project-24-horizons/issues/69

9. Future Work and Potential Improvements

Core AI Features

- **AI schedule suggestions widget:** Implement intelligent scheduling recommendations in the dashboard using Google Gemini API
- **AI study plan generator:** Create personalized study plans with adaptive scheduling and progress tracking

Bug Fixes

- **UI sizing issues:** Fix responsive design problems and component sizing inconsistencies across browsers
- **AI study plan generator:** Create personalized study plans with adaptive scheduling and progress tracking
- **Google Gemini API:** Resolve API integration issues, including user-friendly error handling and rate limiting users.
- **Missing button functionality:** Complete implementation of non-functional UI elements

User Experience Enhancements



- **Calendar page opening animation:** Add smooth transition and loading animations for better visual flow
- **Keyboard shortcuts:** Implement navigation and action shortcuts for power users
- **Calendar data export:** Enable export to iCal, CSV, and PDF formats with selective date range options
- **Modular study timer:** allows the user to change the duration of the study timer

10. Lessons Learned and Projects Takeaways

One big challenge for our group was minimizing dependencies between tasks. Since everyone worked on their own schedule, relying on someone else's work before starting your own could slow things down. To solve this, we made sure to break up the work so tasks stayed as independent as possible. This worked really well, and no one had to wait for someone else to get started.

Another major challenge we faced was integrating all the code together. Everyone was working on their own features and pushing code, but we needed a way to bring it all together and make sure everything worked smoothly. One of our team members stepped up and took on the role of combining everyone's work, making sure the project was cohesive and well-integrated.

Another thing was communication. Poor communication meant features weren't done, tasks went unnoticed, and ultimately, the project was poorly executed. So we made sure to communicate everything and ask for help from the group when needed. Everyone tried to be there for one another, so nobody felt overwhelmed with work.

We realized, toward the end, that some of our features were just extras and added more burden on the team members. So, we dialed some things down to keep the work more balanced and focused on getting the required tasks done properly, instead of trying to do everything badly.

Lessons learned were that clear communication is key, and it's better to focus on doing fewer things well than trying to do everything at once. Keeping tasks independent helps avoid delays, and having someone take



charge of integration makes the project run smoother. Also, being flexible and adjusting the workload when things get overwhelming keeps the team balanced and productive.

11. Appendix


- **Simone Motwani:** Made the Calendar page and the Calendar-AI page. On the Calendar page, I added the import functionality so users can import their calendars through Google or an .ics link. On the Calendar-AI page, I integrated Gemini with the calendar so users can organize their calendar using Gemini. I also connected both with Firebase, so the events created are stored properly. For the Milestone 2 report, I did the SDLC analysis, CI/CD overview, and the lessons learned section.
- **Toma Ozarchevici:** Made UI improvements to the main dashboard and created the new Study Dashboard page with two tabs for notes and flashcards. I connected the dashboard to actual data using context files. I set up an SQL database for persistent storage of notes, flashcards, and goals. I also added unit testing for notes and flashcards and their CRUD operations, along with integration testing. For the milestone 2 report, I created the MVC diagram and explanation, wrote the future work and potential improvements section, and identified a bug in our table.
- **Uttam Sharma:** Made a working backend server that manages Google sign-in with timed authentication tokens and a working database. Worked on GitHub workflow actions for CI/CD. Also worked on merging each person's branches into main and making sure the coding format and functionality worked across files. I worked on "Analysis of Project Success," "Project Testing Strategy," and wrote the README in the project documentation. I hosted the backend on my home server, so I dockerized and managed uptime for the backend; I monitored it through Proxmox and Portainer.
- **Chance Wijewardena:** Made the settings page that manages the theme, displays user info, sets timer duration, and allows the timer to display across all tabs in the navigation bar. Made a working Light/Dark theme toggle. Made the updated data flow diagram for milestone 2.



Change Log

- Version 1.0: Initial Report (Date: June 6, 2025)
- Version 1.1: Updated Report with API feature description (Date: July 4, 2025)
- Version 2.0: Milestone 1 Report
- Version 3.0: Milestone 2 Report

Filled User Feedback Forms



Bidisha Ray, Group 11

StudySync AI Usability Testing Questionnaire

Thank you for participating in our usability testing session. Your feedback is valuable in helping us improve the StudySync AI application. Please complete this questionnaire after your testing session.

1. General Impressions

What are your general thoughts and impressions of the application?

Easy to navigate, clean design

2. Task-Specific Feedback

1. Task 1: Import your current schedule.

Was the task easy or difficult to complete? Why?

Very easy, clear buttons that are easy to navigate to complete task

Did you encounter any issues or confusion during this task?

No

2. Task 2: Use Gemini to create an event on the calendar.

Was the task easy or difficult to complete? Why?

Easy, gemini chatbox easy to locate and use

Did you encounter any issues or confusion during this task?

Gemini was overloaded



StudySync AI

3. Task 3: Sign in and sign out of the application.

Was the task easy or difficult to complete? Why?

easy, sign in and out buttons clear and easy to use

Did you encounter any issues or confusion during this task?

No

3. Usability Suggestions

Do you have any suggestions to improve navigation, design, or clarity of the application?

N/A



StudySync AI

Sadab

StudySync AI Usability Testing Questionnaire

Thank you for participating in our usability testing session. Your feedback is valuable in helping us improve the StudySync AI application. Please complete this questionnaire after your testing session.

1. General Impressions

What are your general thoughts and impressions of the application?

Pretty good & smooth

2. Task-Specific Feedback

1. Task 1: Import your current schedule.

Was the task easy or difficult to complete? Why?

Easy

Did you encounter any issues or confusion during this task?

Nope, can be a little more learnable

2. Task 2: Use Gemini to create an event on the calendar.

Was the task easy or difficult to complete? Why?

Easy

Did you encounter any issues or confusion during this task?

I was stuck in the canvas page



StudySync AI

3. Task 3: Sign in and sign out of the application.

Was the task easy or difficult to complete? Why?

Yes it was due to google sign in

Did you encounter any issues or confusion during this task?

Nope

3. Usability Suggestions

Do you have any suggestions to improve navigation, design, or clarity of the application?

A little more color



StudySync AI

StudySync AI Usability Testing Questionnaire

Thank you for participating in our usability testing session. Your feedback is valuable in helping us improve the StudySync AI application. Please complete this questionnaire after your testing session.

1. General Impressions

What are your general thoughts and impressions of the application?

Honestly amazed! Everything is so nice and intuitive. This is an app I would genuinely use.

2. Task-Specific Feedback

1. Task 1: Import your current schedule.

Was the task easy or difficult to complete? Why?

The task was easy to complete as the UI made it easy to follow.

Did you encounter any issues or confusion during this task?

No issues or confusion.

2. Task 2: Use Gemini to create an event on the calendar.

Was the task easy or difficult to complete? Why?

Once again, the task was very easy to complete as the group did a great job.

Did you encounter any issues or confusion during this task?

A issue I encountered was that the Gemini API was down but that is not the group's fault. Other than that it was great.



StudySync AI

3. Task 3: Sign in and sign out of the application.

Was the task easy or difficult to complete? Why?

Once again it was easy to complete, same reasons as before.

Did you encounter any issues or confusion during this task?

Nope.

3. Usability Suggestions

Do you have any suggestions to improve navigation, design, or clarity of the application?

I honestly don't think it can get any better, you guys did a great job. keep it up!