NAGai: Final Report

Wooyoung Jung, Jiwoo Kim, Joohyoung Jun
CSE 416, Spring 2025

Links:

- Production site (on Vercel)
- API Docs (on Render)
- GitHub Repo

Executive Summary:

- **nagAI** is a web-based focus-tracking assistant that combines a Pomodoro timer with real-time webcam gesture & distraction detection.
- The application lets any visitor sign up (Google OAuth) and immediately begin structured study sessions while receiving gentle "nags" when off-task. The project fully implements every use-case and requirement, and is now production-ready.

Problem & Goal:

- Students often lose concentration when studying alone; existing timers do not intervene when the user looks away, slouches, or checks their phone.
- Our goal was to deliver a usable and useful product that:
    - Detects distraction with ≥ 90 % precision in typical indoor lighting.
    - Logs focus time and distraction events for later review.
    - Fits seamlessly into a browser-only workflow—no extensions or local executables.
    - Demonstrates solid, extensible full-stack engineering suitable for a public launch.

Target Users:

- Our product is mainly aimed at students (middle school, high school, undergraduate, graduate, etc) who want to improve their productivity.
    - There was also some demand for monitoring features, so that parents, teachers, or supervisors can view and monitor the focus data of their children or students.

Definitions:

1. Study session
   - A 25-minute session when users are supposed to be doing their work.
   - The AI API will be given snapshots of the user's webcam to analyze whether the user is focusing on their work or not.
   - If distractions are detected in the image, the image will be saved in the DB. The frontend will pause the Focus timer and display a warning message.

2. Break session
   - A 5-minute session when users do not have to focus on their work.
     i. The AI API will not analyze user snapshots for distractions, and only watch for 'stop' gestures.

3. Distractions
   - User is not focusing on their work.
     i. Using their phone
     ii. User is not in their seat (absence)
     iii. Sleeping
     iv. Talking to other people

4. Focus Time
   - Accumulation of time when the user was being focused for the day.
     i. It is considered a new day starting from 12:00 a.m.
     ii. But, if the user is in their study session from the day before to past midnight, the focus time will include until the study session finishes for the previous day (when the study session started).
   - "Focus Time Log" shows the calendar with the focus time for each day.

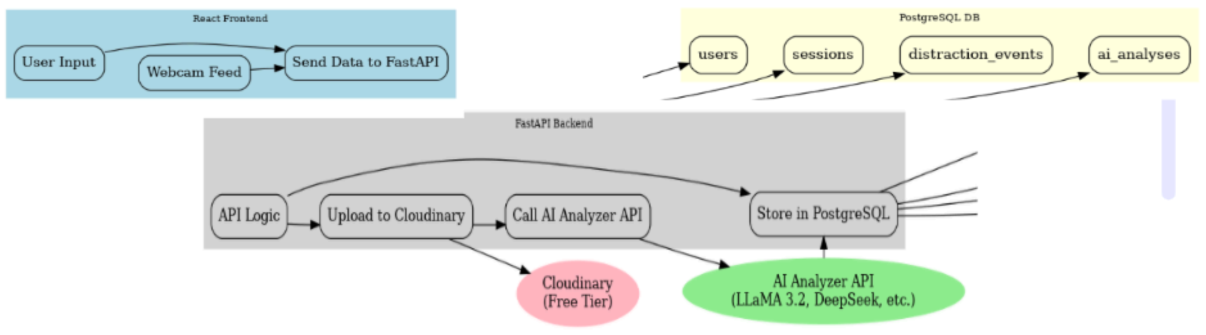
Core Requirements

Requirement	Status	Notes
Real-time distraction detection	Working	Gemini 1.5 Flash Vision API via `@google/generative-ai`
Pomodoro timer with break cycles	Working	25 min for Focus, 5 min for Breaks
CRUD session data via FastAPI + PostgreSQL	Working	For Distraction events, webcam snapshots are saved to a Supabase Storage Bucket.
Focus & Distraction Log pages	Working	Calendar & table views
Desktop push notifications	Working	Sends system notifications to client device

System Overview

Layer	Stack / Key packages	Rationale
Frontend	React 18, Vite, Typescript, CSS Modules, Framer-Motion	Fast dev cycle. Lightweight styling
AI Module	Gemini 1.5 Flash Vision API via Google AI JS SDK	Handles gesture classification
Backend API	FastAPI 0.100, Uvicorn, Pydantic V2	Async Python, Easy OpenAPI
Auth / DB	Supabase Postgres 15	Managed, JWT auth out of box
Hosting	Render (backend) + Vercel (frontend)	Student-friendly deployment
CI/CD	Github-Actions -> Vercel / Render	Auto deploy on ‘main’

Architecture Diagram



Key Implementation Details

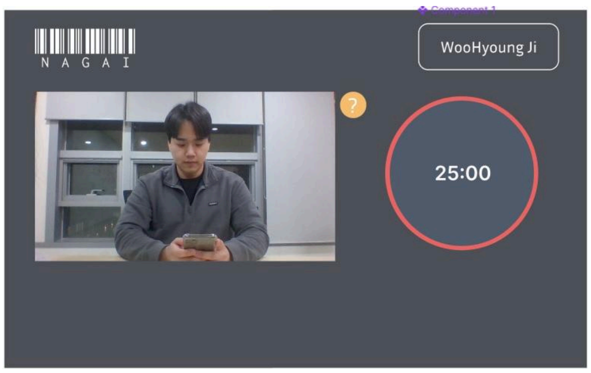
- `useBehaviorDetection.ts` – Captures a snapshot of the user's webcam every 3-15 seconds, sends images to Gemini 1.5 Flash, and pauses whenever `response.isFocused === false.`
- Finite-State Timer (`Timer.tsx`)** – States `IDLE → RUNNING → PAUSED → BREAK`; focus seconds accrue **only** in RUNNING, eliminating earlier off-by-one bugs.

User Guide (for graders)

#	Action	Expected Result
1	Visit 'https://nagai-pi.vercel.app'	Landing page loads.
2	Click 'Continue with Google'	OAuth pops up; account created
3	Show your palm to webcam for three seconds	25:00 timer starts; browser prompts for webcam
4	Look away/ pick up phone	'Distraction detected!' modal plus alert chime
5	Open 'Focus Log'	Calendar shows the total focus time, average focus score, and session count of each day.
6	Open 'Distraction Log'	Distraction Log shows the list of distractions occurred within the day. Default is recent(top) to old(bottom), but user can change the order by clicking the 1st row. By clicking on a specific distraction row, user can see the detailed Distraction Log.

7	Click on a specific row from 'Distraction Log' (opening detailed distraction log)	The detailed distraction log shows the information about the distraction including the snapshot image captured as the distraction.

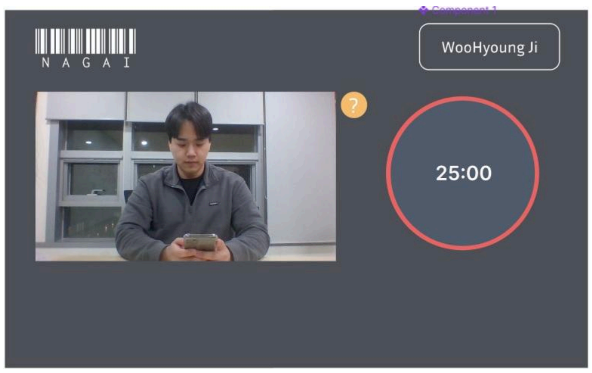
<u>User feedback:</u>



i. Buttons for the Distraction Log and Focus Log may distract users during a focus session.
    - Accordingly, we implemented a 'Widescreen' mode that shows only the webcam feed and hides all other elements from the page.

ii. Notifications to warn users of distractions when they are on other tabs
    - We added this feature before our Final Presentation.

- Final Design:



i. Eliminated Start/Stop buttons.
    - Start/stop sessions through AI-based hand gesture recognition.

Supplementary Documents

- Product Requirements
- Data Design
- Progress Reports (Milestones 1-4)
- Final Presentation Slides

Deployment

- Frontend Deployment
  - Our frontend is built using React with TypeScript and uses React Router for client-side routing.
  - The frontend is deployed on ‘Vercel,’ a platform that supports continuous deployment and automatic build pipelines from GitHub repositories.
  - Every push to the ‘main’ branch will trigger a redeployment.
- Backend Deployment
  - Our API server is hosted on Render.
  - We handled all other backend operations with Supabase.
    - PostgreSQL-based Supabase DB
    - Supabase Auth as a provider for OAuth requests,
    - Supabase Storage Buckets to store webcam snapshots.
- Deployment Workflow
  - Connect the GitHub repository to Vercel.
  - Vercel will detect the React project and automatically use ‘npm run build’ as the build command.
  - The output folder is at ‘/dist/.’
  - The app is hosted at a public URL: https://nagai-pi.vercel.app.
  - During Milestone 4 (Beta Release) and the Final Presentation, this deployed link was used to allow user testing and demonstration.

Code Conventions

- PascalCase for class names, camelCase for method names
- Capitalization must be consistent!
  - Example:
    - src/components/MainPage/MainPage.tsx
    - src/components/focusLog/focus-log.css
- CSS class names: .main-page-outer-wrap
- Must be descriptive
- Modularized code into Components, Pages, Contexts, etc.
- Components: Name should mention the type of component, e.g.: DistractionsButtons, not just Distractions.
- Pages: should have ‘Page’ appended to its name.

Schedule

- Jira link (shared with yoonseok.yang@stonybrook.edu)

Testing

- Testing was done visually, with each team member navigating through the pages, interacting with buttons, and observing the flow or output to see if they match our expectations.
- We established test cases and use cases to document our tests.

Known Issues

- Occasional false positives when the user drinks from a bottle/cup and puts their hand on their face.
- Site becomes unresponsive and displays a blank white screen after inactivity; limitation due to Render Free Tier constraints.
    - This issue seems more stable now; the site loads correctly after 1-3 minutes of latency.

Outcomes

- All core requirements were completed.
- New features added after Milestone 4 / Beta Release:
    - Desktop push notifications for distractions
    - 'Widescreen' mode to hide timer & buttons for less distractions
    - Display 'Gesture Help' overlay for first-time login users
    - Blink 'Gesture Help' button on > 15 seconds of inactivity
    - 'Loading spinner' to guide users on how long they should hold their hand up for

Lessons Learned

- Isolating AI logic early prevented later state-sync headaches.
- Deploying a product to an online service can reveal previously unknown bugs.
- Designing API contracts first saved multiple front-end rewrites.

Future Work

- On-device model (e.g., TensorFlow Lite) to cut latency below 50 ms per frame.
- Adaptive break timing informed by real-time fatigue signals.
- Social leaderboard to encourage accountability.

Contributions

- Wooyoung:
  - Backend:
    - Wrote API endpoints to fetch, PATCH, and POST data between the frontend and the database.
    - Debugged endpoints to return data in correct formats and/or time ranges.
    - Synced frontend timer state updates with session records in database.
    - Resolved time zone normalization issue in frontend
      - Timestamps for Focus sessions and Distraction events are stored in UTC time, but the frontend recognizes the date strings as local (KST) time. I appended a 'T' to the end to force TypeScript to add 9 hours to the timestamps.
  - Frontend:
    - Blink GestureHelpButton after >15 seconds of inactivity
    - Loading spinner to help users time their hand gestures
    - Display Gesture Help overlay for first-time login users
    - Added Average Score metrics in Focus Log
  - Set up Subabase Storage Bucket to store webcam snapshots.
  - Set up Gemini API on Google Cloud Dashboard.
  - Set up & debugged deployment environments on Vercel and Render.
  - Refactored code to follow Code Conventions.
- Jiwoo:
  - Modified the requirements
  - Created the schedule & assigned tasks to members
  - Located class diagram & SQL schema
  - Set up Google OAuth provider on Supabase
  - Enabled notification and chime effect when distraction is detected
  - Frontend
    - FocusLog button
    - FocusLog modal
    - DistractionLog button
    - DistractionLog modal
    - MyPage
- Joohyoung:
  - Modified the requirements.
  - Wrote the Deployment section.
  - Frontend:
    - LoginPage

        - CreateAccountPage
        - TermsPage
        - LoginFailPage
    - Added Widescreen mode in MainPage.
    - Enabled webcam streaming in MainPage.
    - Added motion animations when navigating to other pages.

References:

- https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/
- https://www.theverge.com/2024/9/25/24253774/meta-ai-vision-model-llama-3-2-announced
- Figma
- ChatGPT
- Focus To-Do
- Google Meet