Project: Ivy, Student Success Platform
Course: Engineering Design 4OI6
Team: Group 28 (Ali Raza, Khushi Akbari, Burhanuddin Qadir, Zuhaib Quraishi, Raul Sidhu)

1. Statement of Need and Requirements

Students typically manage academics and career preparation using multiple disconnected tools, such as calendars, note apps, AI chat tools, and job platforms. The result is a fragmented workflow where deadlines, tasks, study, and career prep steps are scattered across different systems. Ivy is intended to reduce that fragmentation by consolidating core student workflows into a single platform. The system centres on course and deadline tracking, AI-assisted study support that can generate actionable learning artefacts (such as summaries and practice content), and early career preparation tools like resume feedback and cover letter drafting.

The project scope is being developed in tiers. A baseline implementation focuses on a central dashboard and study tools, with later tiers expanding into resume and interview support.

Functional requirements are focused on end-to-end student usability. Ivy must support course creation and editing, task and deadline management, and a dashboard that makes upcoming work visible and easy to act on. Ivy also needs a study assistant workflow that can take real course materials and produce structured outputs that populate the system. Supporting features include a Pomodoro timer to help with study sessions and early career pages that establish the intended user experience for resume and cover letter support.

Non-functional requirements include a responsive UI, modular and maintainable components to support parallel development, predictable typed data structures for reliability, and safe handling of user documents during ingestion. At this stage, the system is intentionally optimized for rapid iteration and demonstration rather than full production hardening.

2. High-Level System Design

Ivy is currently implemented as a client-heavy web application with a single AI service boundary exposed as an API route. The architecture has been intentionally kept simple in Semester 1 to reduce integration overhead while the team validated the product workflow.

On the front end, the application is organized as pages (dashboard, study assistant, Pomodoro, career tools) built from reusable components. Shared UI components provide consistent navigation, course displays, and common layout elements. State management is currently local to the client. User data such as courses, tasks, and deadlines is stored in the browser using local storage, which enables persistence across refreshes without requiring authentication during early development.

The AI functionality is accessed through one serverless API endpoint. The study assistant sends chat messages plus extracted file text to this endpoint. The endpoint calls a language model and requests a structured response that conforms to a defined schema. The system returns two useful outputs: a readable assistant response and a structured extraction payload that includes course-related items (for example, tasks and deadlines). The dashboard maps that structured payload into the application's data model so students can go from "course outline PDF" to "populated tasks and deadlines" without manual entry.

Document ingestion is handled through client-side parsing for PDF content, with size control strategies (including truncation for large documents) to keep requests bounded and reliable. This term's design deliberately avoids external dependencies such as databases and auth systems until core flows were functioning.

3. Design Modules

The system is divided into modules with clear responsibilities and integration points.

The Dashboard and Navigation module provides the main entry point, showing courses, high-level progress, and quick access to tools. The Course, Task, and Deadline Management module handles the core data model, CRUD interactions, progress calculations, and persistence through local storage. The Study Assistant and AI Ingestion module covers chat UI, file upload, document parsing, the OpenAI API call, schema extraction, and mapping extracted outputs into the dashboard. The Productivity module implements the Pomodoro timer and associated user controls. The Career Tools module provides initial resume and cover letter workflows to validate user experience and establish future integration points. Finally, the Utilities and Theming module contains shared helper functions, deadline/status logic, and consistent styling patterns across pages.

This modular split allowed the team to develop core student workflows first while prototyping career features in parallel without blocking integration.

4. Progress in Design, Prototyping, and Testing

Dashboard and Navigation.
The dashboard UI has been completed and behaves as a functional control centre for the app. Courses display consistently in the grid layout, and the navigation experience is stable across pages. The system supports realistic demo flows including creating courses, viewing upcoming items, and moving between study and productivity tools.

Validation has primarily been manual. The team repeatedly tested the dashboard flow during development by adding and updating courses, verifying that progress indicators and summary cards updated correctly, and checking that state persisted after refresh.

Course, Task, and Deadline Management.

Core CRUD workflows are implemented so students can manage their academic workload in the system. A key milestone was making tasks and deadlines populate correctly from real course materials rather than relying on mock content. Persistence is implemented through local storage, and the UI supports the intended creation and update flows without requiring a backend in the first term.

A known challenge in this module was time handling. The system initially struggled to consistently interpret "where we are in time," which affected upcoming deadline calculations and the ordering of tasks. This surfaced as incorrect "next deadline" displays and inconsistent schedule behaviour. The team treated this as a reliability issue rather than a cosmetic problem and adjusted the time handling approach so deadlines are interpreted and displayed consistently relative to the user's context. This was one of the most important fixes of the term because it directly impacted trust in the dashboard's "what's next" logic.

Two concrete manual tests that were used repeatedly during iteration were generating tasks and deadlines from a course outline upload and verifying that the created items appear in the correct course context, and refreshing the page to confirm that stored courses and upcoming deadlines remain accurate and in the right order.

Study Assistant and AI Ingestion.
This module reached an end-to-end prototype that works on real documents. The assistant supports user messaging and file upload. PDFs are parsed into text, and the extracted text is sent to the AI endpoint. The endpoint returns a structured extraction response, which is then mapped into new tasks and deadlines in the dashboard. This "outline to action items" workflow is a major indicator of progress because it demonstrates the intended value of Ivy: reducing manual planning overhead for students.

Manual validation focused on functional outcomes rather than individual unit tests. The team tested the ingestion pipeline using representative course outlines, confirming that: uploads succeed, extracted items match the expected schema, generated tasks and deadlines appear in the dashboard, and edge cases such as large documents do not crash the interface.

Current limitations include basic error handling and lack of rate limiting. These are acceptable for prototyping right now but will require strengthening as the system moves toward final stages.

Productivity (Pomodoro).
The Pomodoro timer is implemented as a functional tool and has been used in practice during the term, not only as a demo component. The timer supports mode switching and normal session behaviour. Manual testing was straightforward and based on real use, including starting sessions, switching modes, and validating that the timer behaves correctly during typical study scenarios.

Career Tools.
The resume analyzer and cover letter generator pages exist as UI-first prototypes. They demonstrate the intended workflows and interaction design but are not yet wired to real parsing or AI scoring logic. This approach was intentional. Our team prioritized building the core student workflow first while keeping career features moving forward in parallel at a prototype level.

Across modules, this term's output aligns with an approximately 35-40% implementation state: the UI is complete, core CRUD is present, AI ingestion is functional for creating tasks and deadlines from course outlines, and the Pomodoro feature works end-to-end.

5. Plan for Completion, Integration, and Testing

The second semester will focus on moving from a strong prototype to a robust system that can be used by real students.

1) The team will finish any remaining depth in task and deadline workflows, including polishing edge cases, improving editing flows, and ensuring that course progress and "next deadline" logic remain stable under realistic student usage.
2) The AI pipeline will be extended from extraction into study generation. This includes completing the flashcard and quiz generation flow so the study assistant can produce usable practice content from uploaded materials, not only tasks and deadlines.
3) Career tools will move from prototype to functional features by implementing real resume text extraction and meaningful feedback generation, along with cover letter drafting through an API route similar to the existing study assistant pattern.
4) For integration, the team is leaning toward Supabase in Semester 2 because it supports rapid iteration with authentication and persistent storage. This will allow Ivy to move beyond local storage and support multi-device use.
6) Testing will be expanded beyond manual validation. The team will add focused unit tests for helper logic (deadline calculations and status), integration tests for the AI endpoint's schema compliance, and end to end tests for the highest value flows, course creation, ingestion of a course outline into tasks and deadlines, and the dashboard's "upcoming work" display.

Key risks include model availability, rate limits, and cost for AI features, as well as reliability risks in document ingestion. These will be addressed through stricter request limits, improved error messaging, and a controlled demo mode if needed. Data persistence risk will be reduced by completing the backend migration early in the second semester.

Conclusion

By the end of Semester 1, Ivy has progressed from concept to a working prototype with a complete UI, functional course and workload management, a working AI ingestion pipeline that can generate tasks and deadlines from course outlines, and a Pomodoro timer that is usable in real study sessions. The most significant engineering issue addressed this term was reliable time handling for deadline logic, since it directly affects the dashboard's ability to surface what a student should do next. Semester 2 will focus on integrating Supabase for authentication and persistence, extending the AI pipeline into study and career generation, and strengthening testing and reliability in preparation for final evaluation.