Bilkent University

Department of Computer Engineering

Senior Design Project II – CS 492

T2416

Edux

Final Report

Bilginer Oral	22103163	bilginer.oral@ug.bilkent.edu.tr
Cahit Ediz Civan	22003206	ediz.civan@ug.bilkent.edu.tr
Efe Kaan Fidancı	22102589	kaan.fidanci@ug.bilkent.edu.tr
Görkem Kadir Solun	22003214	kadir.solun@ug.bilkent.edu.tr
Murat Çağrı Kara	22102505	cagri.kara@ug.bilkent.edu.tr

Supervisor: Doruk Öner

Course Instructors: Atakan Erdem, Mert Bıçakçı

May 2nd, 2025

1. Introduction

In today's fast-paced world, learning and staying ahead require more than just access to resources. It demands efficiency, adaptability, and focus. While the internet and digital platforms have made educational content more accessible than ever, they've also created an ocean of information that often overwhelms learners. Juggling multiple resources, managing time, and ensuring progress can be intimidating, but many find it hard to keep their knowledge and reach their goals.

This is where Edux steps in as a game-changer. Edux changes the learning process by utilising the power of large language models (LLMs) and personalised tools to make learning more effective and interactive. With features like detailed explanations powered by LLMs, personalised study schedules, and interactive learning tools, Edux transforms scattered study efforts into structured, goal-oriented journeys [1]. By tailoring study schedules to support aids for near deadlines and generating interactive tools such as flashcards, quizzes, and skill trees from user-uploaded content, Edux simplifies complex learning processes while boosting understanding and retention. With plans for future expansion into mobile and tablets, Edux is poised to meet the evolving demands of modern education, though these developments fall outside the current scope of the senior design project.

This final report presents the complete architecture, development, and evaluation of Edux, marking the culmination of the senior design project. After the introduction, the final report describes detailed and updated system requirements. The final system design is documented through architectural diagrams and component breakdowns, while the development section outlines implementation processes, tools used, and challenges addressed. Testing procedures and results validate the system's functionality and reliability. A maintenance plan is also provided to ensure continued usability and support.

Key engineering considerations such as constraints, standards, risks, and alternatives are discussed, along with reflections on ethical and professional responsibilities observed throughout the project. Teamwork details highlight individual contributions, collaboration strategies, leadership roles, and progress against initial objectives. The report also outlines the new knowledge acquired and applied during the project, supported by the learning methods used. The software system is delivered alongside a user manual with installation instructions and usage guidance. Final code, executables, and documentation are accessible via the project repository and website.

This document serves as a comprehensive summary of the Edux project, capturing its full technical and collaborative journey.

2. Requirements Details

2.1. Functional Requirements

The functional requirements define the core features and capabilities of the Edux platform, focusing on user interaction, educational content management, intelligent learning assistance, and secure, personalised experiences. These requirements are categorised by system modules and user objectives.

2.1.1. User Management and Authentication

Objective: To facilitate secure and role-specific access to the Edux platform.

FR-1.1: The system shall support a primary user role:

- Learner.

FR-1.2: Users shall be uniquely identified and stored with a User ID in the system database.

FR-1.3: The system shall provide secure registration and login mechanisms via:

- Email and password.
- Third-party authentication (e.g., Google OAuth).

FR-1.4: Users shall be able to:

- Update their profile information.
- Reset their password.
- Configure personal preferences.

FR-1.5: Authentication and authorisation processes shall be modularised into microservices.

FR-1.6: Inter-service communication shall implement RBAC (Role-Based Access Control) using unique authentication tokens.

2.1.2. Resource Upload and Processing

Objective: To enable users to upload and interact with diverse educational resources.

FR-2.1: The system shall support uploads in the following formats:

- PDF, images, PowerPoint presentations, and text documents.

FR-2.2: The system shall parse uploaded documents and extract textual data for analysis.

FR-2.3: Users shall receive explanations:

- Page-by-page.
- Across the entire document.

2.1.3. Interactive Learning Tools

Objective: To assist learners in content retention through interactive mechanisms.

FR-3.1: The system shall generate flashcards highlighting key concepts from uploads and chat history when users click generate flashcard with their unique ID.

FR-3.2: A quiz module shall support:

- Multiple-choice questions based on uploaded content, highlighting the percentage of correct answers after completion.

FR-3.3: The platform shall support:

- Auto-generated skill trees from uploaded user material and chat history.
- Quizzes are embedded within skill trees to assess comprehension.

2.1.4. Personalised Study Schedules

Objective: To optimise the learner's time and productivity.

FR-4.1: Users shall be able to:

- Upload/Update course syllabi.
- Manually input syllabi.

FR-4.2: The platform shall generate a card-based study plan for each week linked to the user’s unique ID.

FR-4.3: Notifications shall be triggered based on:

- Proximity to assignment deadlines.

2.1.5. Learning Assistance with LLMs

Objective: To deliver context-aware content explanation and natural language support [2].

FR-5.1: The system shall use large language models to:

- Explain uploaded content with contextual accuracy
- Respond to natural language queries from users

FR-5.2: Responses shall align with the educational content and user history for personalised learning.

2.1.6. Data Security and Privacy

Objective: To uphold user privacy and system integrity.

FR-6.1: All user data (including uploads and personal details) shall be encrypted at rest and in transit.

FR-6.2: Access to sensitive data shall be controlled through:

- Role-based access controls
- User-specific permissions

FR-6.3: The system shall reject material access requests that do not match the user's unique ID.

FR-6.4: The system must comply with international privacy regulations, including the GDPR

2.1.7. Scalability and Performance

Objective: To ensure reliability and responsiveness under varying loads.

FR-7.1: The platform shall scale horizontally to accommodate user and data growth.

FR-7.2: It shall maintain high availability and responsiveness during peak usage (e.g., exam seasons).

FR-7.3: The system shall implement:

- Content caching

- Optimised database queries
- Load-balanced microservice orchestration

2.2. Non-Functional Requirements

2.2.1. Usability

Edux is designed for learners requiring a user-friendly interface that is intuitive, accessible, and easy to navigate. The platform must cater to the diverse needs and preferences of its users. Key usability features should include clearly labelled buttons accompanied by text or universally recognised icons to ensure clarity. The user interface (UI) should be compact and responsive, allowing students to efficiently access tools and resources, especially during busy study periods. The system shall ensure that 95% of users can complete common tasks, such as uploading content or generating flashcards, within three clicks or fewer [3].

2.2.2. Reliability

Ensuring no loss of data integrity in case of system failures is crucial, as these losses can be real-time data entries that have not been committed to the database and could be at risk during system failures. The system should maintain a data consistency rate of at least 99.9%, ensuring that data remains consistent and accurate even during system failures. Scheduled maintenance should be conducted during periods of minimal user activity to minimise disruption. Ensuring that user data is protected during interactions with API gateways and safeguarded against cyberattacks is crucial. The system should achieve an uptime of at least 99.9% annually, which translates to no more than 8.76 hours of downtime per year.

2.2.3. Performance

Database queries, such as loading individual study materials in the chat screen and generating skill trees, flashcards, and quizzes, as well as reloading skill trees, should return results within milliseconds to a few seconds, depending on the complexity and size of the data, to ensure an optimal user experience. For instance, a query handling a few hundred records should ideally be completed in under one second, while more complex queries involving millions of records might take a few seconds. This order of magnitude ensures that users experience minimal delay, maintaining a responsive and efficient system. Moreover, Edux will utilise visual indicators to notify users during longer loading times and manage expectations. The average response time should not exceed two seconds under normal conditions and should not exceed 3 seconds during peak loads.

2.2.4. Supportability

The application must be accessible and function seamlessly across various devices and operating systems, including desktops, mobile phones, and tablets, with integration in the future

scope of the project. The application should be compatible with at least 95% of the most commonly used devices [4]. Support processes must align with industry standards and regulatory requirements, ensuring data security and user trust [5].

2.2.5. Scalability

Edux should seamlessly scale to accommodate increasing user traffic and a growing database of study materials. The system should be designed to handle higher loads without compromising performance or usability. An example metric to track Edux's ability to handle higher loads without compromising performance or usability is system throughput, measured in transactions per second (TPS). This metric captures the volume of transactions the system can process within a given timeframe, reflecting performance and scalability under varying loads. This approach can provide early detection of issues and proactive scaling. Moreover, the system shall handle at least 1000 simultaneous users without performance degradation.

3. Final Architecture and Design Details

3.1. Architectural Overview

Edux adopts a modular, multi‑layered architecture designed to support scalability, maintainability, and seamless integration of advanced AI capabilities. At its core, the system is divided into three primary layers Client, Backend, and Data each encapsulating distinct responsibilities and interacting through a centralized API Gateway. This layered approach allows the platform to cleanly separate concerns: the Client Layer focuses on user interaction, the Backend Layer, that has a microservices architecture, provides business logic and AI services, and the Data Layer ensures reliable storage and retrieval of information. By compartmentalizing functionality and leveraging service boundaries, Edux achieves a high degree of flexibility, making it straightforward to introduce new features or scale existing ones without disrupting the overall system.

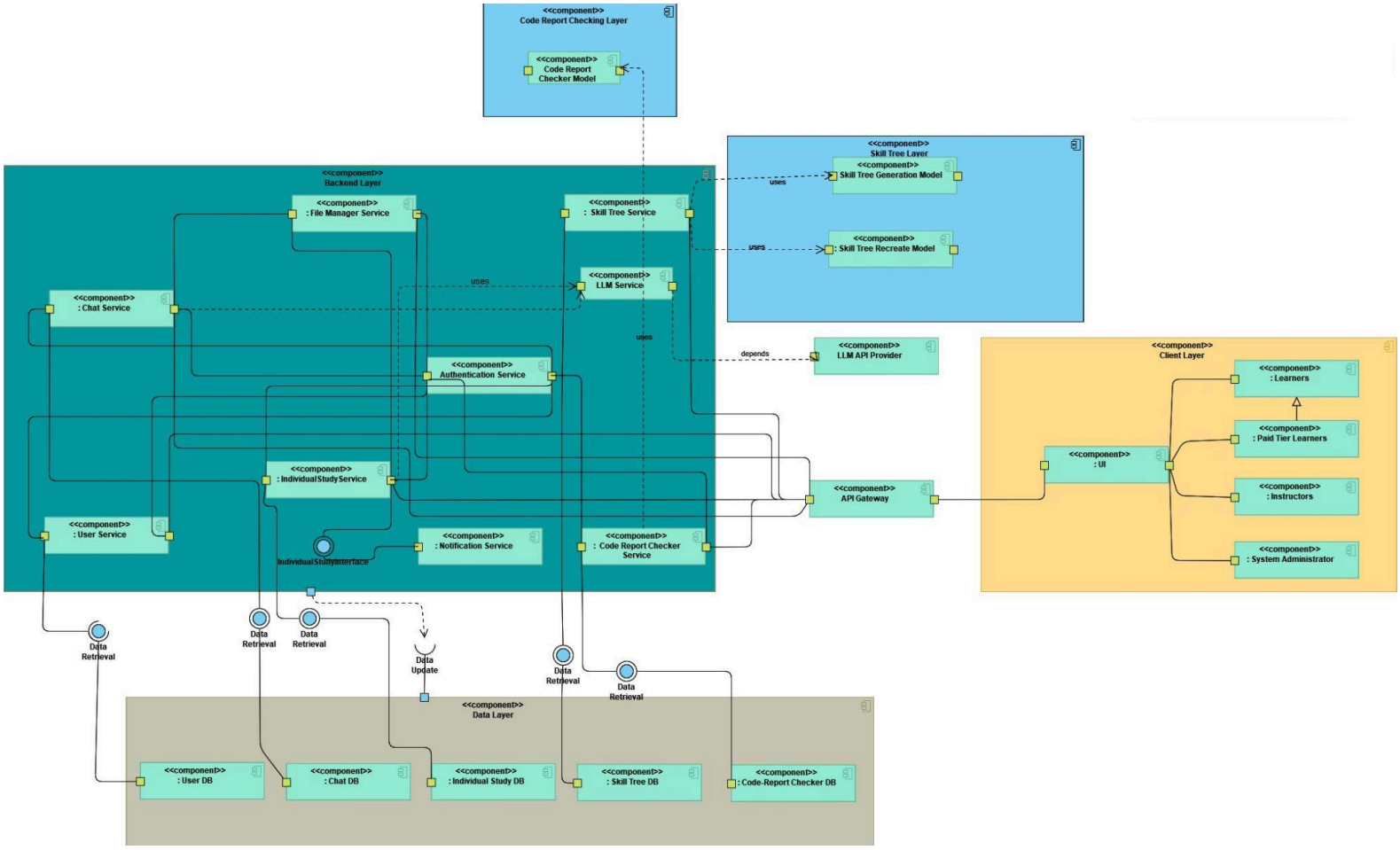


Figure 1: Edux Component Diagram

3.2. Layered Structure

The Client Layer presents a responsive web interface built with modern frontend technologies, allowing learners, paid subscribers access the platform's features. All interactions originate here, routed through the API Gateway to ensure secure and authenticated communication. Beneath this, the Backend Layer orchestrates a suite of microservices that collectively implement the system's functionality. These services include Authentication for user access control, User Management, Course Management, Chat and GenAI Services for interactive study tools, Skill Tree service that generates and stores skill trees and user progression and the File Manager for handling uploads. Each service exposes a well-defined API, with public endpoints secured via JSON Web Tokens and private inter-service channels protected by API keys. At the foundation, the Data Layer comprises multiple databases such as User DB, Course DB, Chat DB, Skill Tree DB, and specialized storage in Azure Blob for persisting structured records and unstructured files. This stratification promotes clear interfaces and fault isolation, ensuring that issues in one layer, such as heavy AI processing in the GenAI Service, do not cascade across the entire platform.

3.3. Subsystem Decomposition and Services

Within the Backend Layer, Edux is decomposed into focused microservices, each responsible for a specific domain. The Authentication Service issues and validates JWTs, handling secure login flows and enforcing role‑based access. The User Service manages profile creation, updates, and retrieval, while the Course Service oversees syllabus uploads, weekly study plan generation, and course metadata. Chat Service enables real‑time AI‑driven discussions and slide‑based learning, leveraging the GenAI Service's integration with external LLM APIs like Gemini or OpenAI. The File Manager Service abstracts storage concerns, offering endpoints to store and retrieve user uploads ranging from lecture slides to generated flashcards and quizzes. Notification and User Analytics services round out the ecosystem, delivering timely alerts and insight dashboards. This decomposition enables independent development, deployment, and scaling of each service, reducing interdependencies and accelerating feature rollout.

3.4. Data Management and Security

Edux's Data Layer employs MySQL databases for structured data within containerized microservices managed by Docker, ensuring high availability and transaction integrity [6]. Unstructured assets such as PDFs, images, quizzes, and flashcards are persisted to cloud object storage (Azure Blob) [7], with the File Manager Service maintaining reference IDs for efficient access. To safeguard sensitive educational content and personal information, Edux enforces strict role‑based access controls validated by the Authentication Service. Inter‑service communication is further secured via mutual API key handshakes, preventing unauthorized requests within the internal network. Compliance with regulations such as GDPR [8] and regional data protection laws is baked into the architecture, ensuring that user materials and metrics remain private and are never served to unintended recipients. With automated backups, failover strategies, and continuous security monitoring, the design prioritizes data integrity and user trust, laying a robust foundation for future growth and feature expansion.

4. Development/Implementation Details

4.1. Frontend

The Edux frontend is developed with Next.js (App Router) and TypeScript, using a modular architecture optimized for scalability and developer efficiency. All UI components are built using the `shadcn/ui` component library, layered on top of Radix UI primitives and styled

with Tailwind CSS, providing a cohesive, accessible, and flexible design system [9, 10, 11, 12, 13].

At the highest level, all routes reside under the `app/` directory. The root `layout.tsx` wraps every page in essential global providers, and the `ClientLayout` component manages client-specific behaviors. This includes injecting a shared Navbar, which is excluded on authentication-related pages like `/sign-in`, `/sign-up`, and `/forgot-password` to provide a distraction-free user experience. `ClientLayout` also handles global theming with a `ThemeProvider`, disabling transitions during theme changes to avoid flickering. Additionally, a global `Toaster` component (via Sonner) is mounted here to deliver notifications, and a `useExitTracker` hook monitors exit events.

Global styles are defined in `globals.css`, with Tailwind CSS and PostCSS driving a utility-first styling system. Configuration via `tailwind.config.ts` ensures theme consistency and responsive behaviour.

The routing structure reflects the platform's feature set. Public routes (sign-in, sign-up, password reset, etc.) are isolated within their own directories. Once authenticated, users are directed to a centralised dashboard (`user-dashboard.tsx`), which links to key learning features—Courses, Flashcards, Quizzes, Skill Trees, and Chat. Each feature is accessible through nested routes under the course, such as `app/course/[course_id]/flashcards` or `app/course/[course_id]/quizzes`, enabling structured navigation.

The `components/` folder houses reusable elements, including `navbar.tsx`, `sidebar.tsx`, `mode-toggle.tsx`, dialogues like `confirmation-dialog.tsx`, and general UI primitives (buttons, separators, scroll areas, etc.) built using `shadcn/ui`. Notifications and toasts are wrapped in `toast.tsx`, abstracting the Sonner library. Moreover, `components/` folder also houses the reusable elements for each service component, such as Courses, Flashcards, Quizzes, Skill Trees, Search, Notifications and Chat, where inside create-edit dialogues, microservice-specific components are placed.

Data communication between the frontend and backend is handled via service-layer abstractions. Each service—such as the user, chat, file, or analytics service—exposes its own API. The frontend interacts with these services using dedicated modules (e.g., `userService`, `chatService`, etc.). These services issue authenticated HTTP requests using bearer tokens. During creation screens, if there is multiple data to be inserted using formdata we use multipart/form-data and pass the formdata to the request.

An example for how communication occurs from backend to frontend:

```
const coursesResponse = await userService.get("/user", {
  headers: {
    "Content-Type": "application/json",
    Authorization: `Bearer ${token}`,
  },
});
setCourses(coursesResponse.data?.courses || []);
```

Each of these services communicates with a corresponding backend microservice, and the route resolution on the backend is managed through a shared `public.py` entry point. This file defines route handlers for actions such as retrieving user data, history messages, file assets, and other context-specific content requested by the frontend. Whether the service relates to user progress, chat history, flashcard content, or skill tree metadata, `public.py` acts as the centralised gateway that maps incoming requests to the appropriate handler logic.

Advanced UI interactions are central to Edux's learning experience. Flashcards feature animated card-flip mechanics; quizzes offer multi-choice interaction with feedback and scoring logic; and skill trees use React Flow to visualise a graph-based learning path. Skill nodes are locked until their related quizzes are passed (with an 80% threshold). Skill dialogues manage quiz initiation and post-quiz progression [14].

Charts and analytics throughout the app are rendered with Recharts, while React Markdown (augmented with `rehype-katex` and `remark-math`) enables support for formatted text and LaTeX-style mathematical notation, used frequently in AI-generated explanations. Subtle UI transitions are powered by React Spring, enhancing visual flow without overwhelming users [15].

To maintain high code quality, the codebase is linted with ESLint and formatted with Prettier, including the `eslint-plugin-tailwindcss` for class ordering. These tools are enforced as pre-deployment checks. Jest and React Testing Library (configured via `jest.config.js` and `setupTests.ts`) ensure component correctness through unit tests. All configurations—including `next.config.mjs`, `postcss.config.mjs`, and `tailwind.config.ts`—are standardised across the team via version control

Overall, the Edux frontend emphasizes performance, extensibility, and developer clarity. The use of `shadcn/ui`, modular routing via the App Router, and service-driven backend communication (via `public.py`) allows the platform to support new features—such as AI-enhanced tools, real-time collaboration, or expanded analytics—without compromising user experience or code structure.

4.2. Backend

The backend of the Edux platform is organized as a suite of independently deployable microservices, each responsible for a distinct domain of functionality. The codebase is laid out under a single repository containing separate directories for seven core services Authentication, User, Course, Chat, Skill Tree, GenAI, and FileManager alongside a Docker Compose manifest and shared scripts. Each service follows a consistent structure: a `main.py` that instantiates a FastAPI application; subpackages for routing (api), data models and persistence (database or models), security checks, utility functions, and inter-service clients. This division enforces a clear separation of concerns and makes it straightforward to develop, test, and scale each service in isolation [6, 16, 17].

Containerization and local orchestration are handled via Docker and Docker Compose. The root docker-compose.yml defines a shared bridge network and dedicates named volumes to each service's database storage (for example, `db_user`, `db_chat`, etc.). Each microservice directory includes a Dockerfile that installs its Python dependencies, copies the application code, and specifies environment variables via an `.env` file. In development, docker-compose up will bring up all seven containers simultaneously, wiring them together so that the Authentication Service issues tokens consumed by the User and Course Services, while the FileManager and GenAI services provide specialized functionality to the Course and Chat Services over HTTP [6, 16].

The Authentication Service is the linchpin for security. It exposes a public `/login` endpoint implementing OAuth2 password flow to verify credentials (email and password) against a user datastore and issues JWTs with expiration. A private router, protected by an API-key check, offers token validation endpoints so that downstream services can verify both service-to-service credentials and user tokens. Passwords are hashed using industry-standard libraries, and tokens are signed with a shared secret. This service encapsulates all logic for identity verification and access token lifecycle management [18].

The User Service manages user records, profiles, and preferences. Through its public API, clients can create accounts, retrieve profile details, and list associated resources. Internally, it uses SQLAlchemy to map User entities to a MySQL database. Protected routes guarded by the same JWT bearer scheme allow administrators or the Authentication Service to update or delete user records. This service also provides client code used by other services (e.g., the Course Service) to fetch user information without duplicating authentication logic [19].

The Course Service orchestrates course definitions, syllabi, and study plan generation. Users can create or update courses by supplying metadata and uploading files (such as syllabi or icons). When a syllabus is uploaded or modified, the service invokes the GenAI Service via its client library to produce a weekly study plan, persists that plan through the `FileManager`

`Service`, and stores references in its own database. Course entities, including linked files and generated content, are exposed through public routes (requiring JWTs) and internally updated via private routes secured by API keys.

The Chat Service underpins interactive learning conversations. It maintains chat sessions scoped to individual courses, allowing learners to send text messages or upload documents (PDFs, slide decks). Uploaded content is processed, indexed, and stored via the FileManager Service, then incorporated into prompts sent to the GenAI Service for contextual explanations. Learners can also trigger generation of quizzes and flashcards directly from chat history. All chat data is persisted in a dedicated database, and endpoints are protected with both JWT and API-key verification to secure user access and interservice calls.

The Skill Tree Service orchestrates the creation and management of directed-acyclic "skill graphs" where each node represents a quiz and edges encode prerequisite relationships. When learners interact with the chat and upload new files, the service calls the GenAI Service to synthesize a tree of quizzes tailored to the student's history and the course's learning objectives. It then persists the resulting node and edge models in its own MySQL database via SQLAlchemy, using adjacency-list tables to represent parent/child relationships. Quiz content and any generated assets are stored and retrieved through the FileManager Service, with each node's quiz id referencing that storage. Public endpoints—guarded by JWTs—allow clients to fetch a user's current skill tree, report node completions, and query which quizzes are now unlocked. Private routes, secured by service API keys, enable other backend components to update progress and regenerate subtrees when content changes. This separation of structure (tree topology) from state (user progress) ensures both referential integrity and efficient in-memory traversal for unlocking new quizzes.

The GenAI Service encapsulates all integration with external large language models. It provides a private API offering multiple AI-powered operations, weekly study plan generation, flashcard creation, quiz synthesis by routing requests to different providers (Google Gemini, for example) through a unified client abstraction. Incoming requests include chat histories or document content; responses are validated against expected JSON schemas before being returned. Access to these endpoints requires a valid service API key, ensuring only authorized backend components can leverage potentially costly AI calls.

Finally, the `FileManager Service` handles all user-uploaded assets and generated files. It exposes public endpoints to upload, download, and delete files. Metadata about each file (original filename, storage path, owner) is recorded in a MySQL database, while the files themselves reside on Azure Blob Storage. Private routes allow other microservices to manage files programmatically, with access gated by API-key checks.

Cross-cutting concerns are uniformly addressed across services. CORS middleware is enabled by default (with plans to lock it down in production), environment variables configure

database connections and secrets, and Pydantic schemas enforce request/response structure consistency. By combining FastAPI's async performance, Docker‑based isolation, and clear API‑key plus JWT‑based security, this microservices backend delivers a scalable, maintainable foundation that aligns with the detailed design requirements of Edux [20].

4.3. Deployment

We leverage Docker to provide a consistent, isolated development environment across all platforms. By packaging each application component's frontend, backend, and any auxiliary services along with its dependencies into its own container, we eliminate *it works on my machine* issues and ensure every developer operates in an identical runtime. This containerized approach not only boosts reproducibility but also makes our workflows portable, allowing team members to start coding without wrestling with local setup discrepancies [6, 16].

To reflect the distinct needs of our tech stack, we maintain separate Dockerfiles for FastAPI, React, and any other services. Each Dockerfile precisely outlines the environment, libraries, and build steps required for its component, enabling independent builds and focused updates. We then orchestrate these containers with a single docker-compose configuration, which binds them together into a unified, multi-service environment [6, 13, 16, 17].

This architecture greatly simplifies service management: spinning up the entire system becomes as easy as running one command, and scaling individual components for example, adding more API workers or front-end instances can be handled seamlessly [16]. By encapsulating each service and coordinating them with Docker Compose, we achieve a streamlined deployment process that adapts effortlessly to development, staging, and production environments.

5. Test Cases

The procedures are written for test software engineers.

5.1. Functionality Test Cases

Test ID	TCF0	Category	Functional	Severity	Medium
Objective	Logout				
Steps	• Login to Edux with any account. • After logging in, log out using the navigation bar on the right. • Click the "Logout" button.				

Expected	The user will log out and be directed to the login page.
Tester	Murat Çağrı Kara
Date/Result	13.08.2024 / Success.

Test ID	TCF1	Category	Functional	Severity	Low
Objective	Show Terms of Service and Privacy Policy				
Steps	● Open Edux on the browser. ● Navigate to the “Sign in” page. ● At the bottom, click “Terms of Service” and “Privacy Policy,” and click one of them.				
Expected	Terms of Service or Privacy Policy are shown.				
Tester	Murat Çağrı Kara				
Date/Result	13.08.2024 / Success.				

Test ID	TCF2	Category	Functional	Severity	Low
Objective	Show About Page				
Steps	● Login to Edux on the browser. ● Navigate to the “About” page from the navigation bar. ● Click “Team” or “Github Page.”				
Expected	Team or Github Page are shown.				
Tester	Murat Çağrı Kara				
Date/Result	17.08.2024 / Success.				

Test ID	TCF3	Category	Functional	Severity	Medium
Objective	Show Subscription Service				
Steps	• Login to Edux on the browser. • Navigate to the “Services” dropdown from the navigation bar. • Select the “Subscription Service.”				
Expected	Subscriptions page will be shown.				
Tester	Murat Çağrı Kara				
Date/Result	15.02.2025 / Success.				

Test ID	TCF4	Category	Functional	Severity	Medium
Objective	Show Notifications				
Steps	• Login to Edux on the browser. • Navigate to the “Services” dropdown from the navigation bar. • Select the “Notifications.”				
Expected	Notifications page will be shown.				
Tester	Görkem Kadir Solun				
Date/Result	22.04.2025 / Success.				

Test ID	TCF5	Category	Functional	Severity	Medium
Objective	Search in Edux				

Steps	• Login to Edux on the browser. • Navigate to the “Search in Edux…” dropdown from the navigation bar. • Write the feature that you want to access. • Select the feature shown below.
Expected	The selected feature will be opened.
Tester	Murat Çağrı Kara
Date/Result	24.08.2024 / Success.

Test ID	TCF6	Category	Functional	Severity	Medium
Objective	Switch between dark and light mode				
Steps	• Open Edux on the browser. • Navigate to the moon icon on the right of the navigation bar. • Click the moon icon. • Select the theme you want: dark, light, or system.				
Expected	The user interface will follow the wanted theme.				
Tester	Murat Çağrı Kara				
Date/Result	27.08.2024 / Success.				

Test ID	TCF7	Category	Functional	Severity	Critical
Objective	Open Skill Tree				
Steps	• Login to Edux on the browser. • Click the “Skill Tree” card in the main menu.				

Expected	The Skill Tree page will be shown.
Tester	Görkem Kadir Solun
Date/Result	25.10.2024 / Success.

Test ID	TCF8	Category	Functional	Severity	Critical
Objective	Creating a Skill Tree				
Steps	• Login to Edux on the browser. • Click the “Skill Tree” card in the main menu. • Open the sidebar on the Skill Tree page. • Click the Create icon on the top of the sidebar. • Fill in the required fields in the creation modal.				
Expected	A new Skill Tree will be created.				
Tester	Görkem Kadir Solun				
Date/Result	17.02.2025 / Success.				

Test ID	TCF9	Category	Functional	Severity	Critical
Objective	Creating a Skill Tree: Unsuccessful Creation				
Steps	• Login to Edux on the browser. • Click the “Skill Tree” card in the main menu. • Open the sidebar on the Skill Tree page. • Click the Create icon on the top of the sidebar. • Fill in the required fields in the creation modal. • Skill Tree creation failed.				
Expected	Skill Tree creation fails with an error message: “Unable to create Skill Tree, please try again later.” Error logs indicate server-side validation failure. Issue requires immediate backend service review.				

Tester	Görkem Kadir Solun
Date/Result	17.02.2025 / Fail.

Test ID	TCF10	Category	Functional	Severity	Critical
Objective	Editing a Skill Tree				
Steps	● Login to Edux on the browser. ● Click the “Skill Tree” card in the main menu. ● Open the sidebar on the Skill Tree page. ● Click the edit icon on the Skill Tree that you want to edit. ● Change the fields in the edit modal.				
Expected	The selected Skill Tree will be edited.				
Tester	Görkem Kadir Solun				
Date/Result	17.02.2025 / Success.				

Test ID	TCF11	Category	Functional	Severity	Critical
Objective	Editing a Skill Tree: Unsuccessful Edit				
Steps	● Login to Edux on the browser. ● Click the “Skill Tree” card in the main menu. ● Open the sidebar on the Skill Tree page. ● Click the edit icon on the Skill Tree that you want to edit. ● Change the fields in the edit modal.				
Expected	The selected Skill Tree could not be edited; an error occurred stating “Unable to save changes, please retry.”				
Tester	Görkem Kadir Solun				

Date/Result	17.02.2025 / Fail.

Test ID	TCF12	Category	Functional	Severity	Critical
Objective	Deleting a Skill Tree				
Steps	• Login to Edux on the browser. • Click the “Skill Tree” card in the main menu. • Open the sidebar on the Skill Tree page. • Click the delete icon on the Skill Tree that you want to delete. • Click confirm on the deletion modal.				
Expected	The selected Skill Tree will be deleted.				
Tester	Görkem Kadir Solun				
Date/Result	17.02.2025 / Success.				

Test ID	TCF13	Category	Functional	Severity	Critical
Objective	Viewing a Skill Tree				
Steps	• Login to Edux on the browser. • Click the “Skill Tree” card in the main menu. • Open the sidebar on the Skill Tree page. • Click the view icon on the Skill Tree that you want to view.				
Expected	The selected Skill Tree will be shown, and the graph will be rendered.				
Tester	Görkem Kadir Solun				
Date/Result	17.02.2025 / Success.				

Test ID	TCF14	Category	Functional	Severity	Critical

Objective	Creating an Individual Study
Steps	● Login to Edux on the browser. ● Click the Create icon on the right of the “Your Studies” in the main menu. ● Fill in the required sections on the modal. ● Click the Save button.
Expected	A new Individual Study will be created.
Tester	Görkem Kadir Solun
Date/Result	14.09.2024 / Success.

Test ID	TCF15	Category	Functional	Severity	Critical
Objective	Creating an Individual Study: Creation Fail				
Steps	● Login to Edux on the browser. ● Click the Create icon on the right of the “Your Studies” in the main menu. ● Fill in the required sections on the modal. ● Click the Save button.				
Expected	Error message displayed: “Unable to create Individual Study due to incomplete fields.” despite all required fields being filled. Validation or backend issue suspected.				
Tester	Görkem Kadir Solun				
Date/Result	14.09.2024 / Fail.				

Test ID	TCF16	Category	Functional	Severity	Critical

Objective	Editing an Individual Study
Steps	● Login to Edux on the browser. ● Click the Edit icon on the right of the “View Course” in the Your Studies section on the Individual Study you want to edit. ● Change the sections on the modal. ● Click the Save button.
Expected	The selected Individual Study will be updated.
Tester	Görkem Kadir Solun
Date/Result	15.09.2024 / Success.

Test ID	TCF17	Category	Functional	Severity	Critical
Objective	Editing an Individual Study: Edit Fail				
Steps	● Login to Edux on the browser. ● Click the Edit icon on the right of the “View Course” in the Your Studies section on the Individual Study you want to edit. ● Change the sections on the modal. ● Click the Save button.				
Expected	The edit action fails, displaying an error message: “Update failed, please retry.”				
Tester	Görkem Kadir Solun				
Date/Result	15.09.2024 / Fail.				

Test ID	TCF18	Category	Functional	Severity	Critical
Objective	Deleting an Individual Study				

Steps	• Login to Edux on the browser. • Click the Delete icon on the right of the “View Course” in the Your Studies section on the Individual Study you want to delete. • Click the Confirm button.
Expected	The selected Individual Study will be deleted.
Tester	Murat Çağrı Kara
Date/Result	15.09.2024 / Success.

Test ID	TCF19	Category	Functional	Severity	Critical
Objective	Open an Individual Course				
Steps	• Login to Edux on the browser. • Click the “View Course” button on the Individual Study you want to open in the Your Studies section.				
Expected	The user will be navigated to the selected Individual Study.				
Tester	Murat Çağrı Kara				
Date/Result	14.09.2024 / Success.				

Test ID	TCF20	Category	Functional	Severity	Critical
Objective	Open Flashcards				
Steps	• Login to Edux on the browser. • Click the “View Course” button on the Individual Study you want to open in the Your Studies section. • Click the Flashcards card on the Individual Study page.				

Expected	Flashcards page will be opened.
Tester	Efe Kaan Fidancı
Date/Result	23.01.2025 / Success.

Test ID	TCF21	Category	Functional	Severity	Critical
Objective	Open Quizzes				
Steps	● Login to Edux on the browser. ● Click the “View Course” button on the Individual Study you want to open in the Your Studies section. ● Click the Quizzes card on the Individual Study page.				
Expected	Quizzes page will be opened.				
Tester	Efe Kaan Fidancı				
Date/Result	23.01.2025 / Success.				

Test ID	TCF22	Category	Functional	Severity	Critical
Objective	Open Instructor/Chat				
Steps	● Login to Edux on the browser. ● Click the “View Course” button on the Individual Study you want to open in the Your Studies section. ● Click the Instructor/Chat card on the Individual Study page.				
Expected	Instructor/Chat page will be opened.				
Tester	Murat Çağrı Kara				
Date/Result	23.10.2024 / Success.				

Test ID	TCF23	Category	Functional	Severity	Critical
Objective	Open Weekly Study Plan				
Steps	• Login to Edux on the browser. • Click the “View Course” button on the Individual Study you want to open in the Your Studies section. • Click the Weekly Study Plan card on the Individual Study page.				
Expected	Weekly Study Plan page will be opened.				
Tester	Efe Kaan Fidancı				
Date/Result	19.11.2025 / Success.				

Test ID	TCF24	Category	Functional	Severity	Critical
Objective	Upload Syllabus				
Steps	• Login to Edux on the browser. • Click the “View Course” button on the Individual Study you want to open in the Your Studies section. • Click the Upload Syllabus card on the Individual Study page. • Upload the Syllabus to the opened modal. • Click the Upload button.				
Expected	The syllabus will be uploaded/updated.				
Tester	Efe Kaan Fidancı				
Date/Result	19.11.2025 / Success.				

Test ID	TCF25	Category	Functional	Severity	Critical
Objective	Create Chat				

Steps	• Login to Edux on the browser. • Click the “View Course” button on the Individual Study you want to open in the Your Studies section. • Click the Instructor/Chat card on the Individual Study page. • Click the Create chat button on the top left of the page. • Fill in the opened sidebar. • Click the Create chat button.
Expected	A new chat will be created.
Tester	Murat Çağrı Kara
Date/Result	23.10.2024 / Success.

Test ID	TCF26	Category	Functional	Severity	Critical
Objective	Create Chat: Creation Fail				
Steps	• Login to Edux on the browser. • Click the “View Course” button on the Individual Study you want to open in the Your Studies section. • Click the Instructor/Chat card on the Individual Study page. • Click the Create chat button on the top left of the page. • Fill in the opened sidebar. • Click the Create chat button.				
Expected	Chat creation failed with error: “Chat creation unsuccessful, please check input fields and retry.”				
Tester	Murat Çağrı Kara				
Date/Result	23.10.2024 / Fail.				

Test ID	TCF27	Category	Functional	Severity	Critical
Objective	Send a Message to a Chat				
Steps	• Login to Edux on the browser.				

	• Click the “View Course” button on the Individual Study you want to open in the Your Studies section. • Click the Instructor/Chat card on the Individual Study page. • Select the chat from which you want to send the message from the sidebar. • Type your message at the bottom of the page. • Click the Send Message button.
Expected	A new message will be sent to the chat.
Tester	Murat Çağrı Kara
Date/Result	23.10.2024 / Success.

Test ID	TCF28	Category	Functional	Severity	Critical
Objective	Rename Chat				
Steps	• Login to Edux. • Click the “View Course” button on the Individual Study you want to open in the Your Studies section. • Click the Instructor/Chat card on the Individual Study page. • Select the chat from which you want to rename from the sidebar. • Click the three dots on the right of the selected chat and press Edit. • A Dialog will pop up and for the Name field write the new name. • Click the Confirm button.				
Expected	The selected chat will be renamed.				
Tester	Murat Çağrı Kara				
Date/Result	23.10.2024 / Success.				

Test ID	TCF29	Category	Functional	Severity	Critical
Objective	Create a Quiz from a Chat				
Steps	• Login to Edux. • Click the “View Course” button on the Individual Study you want to open in the Your Studies section. • Click the Instructor/Chat card on the Individual Study page.				

	● Select the chat from which you want to create a quiz from the sidebar. ● Click the three dots on the right of the selected chat. ● Click the Create Quiz button. ● Click the Confirm button.
Expected	A new Quiz will be created.
Tester	Murat Çağrı Kara
Date/Result	23.01.2025 / Success.

Test ID	TCF30	Category	Functional	Severity	Critical
Objective	Attaching a file to a Chat				
Steps	● Login to Edux. ● Click the “View Course” button on the Individual Study you want to open in the Your Studies section. ● Click the Instructor/Chat card on the Individual Study page. ● Select the chat from which you want to attach a file from the sidebar. ● Click the file attach button at the bottom of the page. ● Select the files and click the Confirm button.				
Expected	A new file will be attached.				
Tester	Murat Çağrı Kara				
Date/Result	25.10.2024 / Success.				

Test ID	TCF31	Category	Functional	Severity	Critical
Objective	Deleting a Chat				
Steps	● Login to Edux. ● Click the “View Course” button on the Individual Study you want to open in the Your Studies section. ● Click the Instructor/Chat card on the Individual Study page. ● Select the chat that you want to delete from the sidebar. ● Click the three dots on the right of the selected chat. ● Click the Delete button. ● Click the Confirm button.				

Expected	Selected chat will be deleted.
Tester	Murat Çağrı Kara
Date/Result	25.10.2024 / Success.

Test ID	TCF32	Category	Functional	Severity	Critical
Objective	Create Flashcards from a Chat				
Steps	● Login to Edux. ● Click the “View Course” button on the Individual Study you want to open in the Your Studies section. ● Click the Instructor/Chat card on the Individual Study page. ● Select the chat from which you want to rename from the sidebar. ● Chat for some time ● Click the three dots on the right of the selected chat. ● Click Create Flashcard button ● Confirm a Flashcard has appeared in Flashcards				
Expected	A Flashcard must appear in the individual study’s flashcard section.				
Tester	Murat Çağrı Kara				
Date/Result	23.01.2025 / Success.				

Test ID	TCF33	Category	Functional	Severity	Low
Objective	Ensure that users can update their profile information.				
Steps	● Login to Edux. ● Navigate to the “Profile” section. ● Edit profile details (e.g., name, bio, avatar). ● Click “Save.”				
Expected	The updated information should be saved and displayed correctly.				
Tester	Görkem Kadir Solun				

Date/Result	14.04.2025 / Success.

Test ID	TCF34	Category	Functional	Severity	Medium
Objective	Ensure that users can insert another slide to chat.				
Steps	● Login to Edux. ● Click the “View Course” button on the Individual Study you want to open in the Your Studies section. ● Click the Instructor/Chat card on the Individual Study page. ● Select the chat from which you want to rename from the sidebar. ● Click the three dots on the right of the selected chat and press Edit. ● A Dialog will pop up and for the Upload New Slides (Optional) upload the new slides. ● Click Confirm.				
Expected	A new slide will be attached.				
Tester	Bilginer Oral				
Date/Result	14.04.2025 / Success.				

5.1.1. Integration Test Cases

Test ID	TCI0	Category	Integration	Severity	Critical
Objective	Retrieving courses from the course database				
Steps	● Sign in as one of the users ● Call the course service's endpoint for retrieving the courses that belong to our user				
Expected	● The courses are expected to be retrieved from the course service's database successfully.				

Tester	Bilginer Oral
Date/Result	19.02.2024 / Success.

Test ID	TCI1	Category	Integration	Severity	Critical
Objective	Retrieving chats from the chats database				
Steps	• Sign in as one of the users • Call the course service's endpoint for retrieving the chat id's associated with it. • Call the chat service's endpoint for retrieving the chats with the given id's from the chat database				
Expected	• The chats are expected to be retrieved from the chat service's database successfully.				
Tester	Bilginer Oral				
Date/Result	17.02.2024 / Success.				

Test ID	TCI2	Category	Integration	Severity	Critical
Objective	Getting the current user from the user database				
Steps	• Sign in as one of the users • Call any public endpoint in any service, that endpoint depends on the current user so it will call the user service to fetch the current user. • The current user will get the current user using the given Json web token				
Expected	• The current user is expected to be retrieved from user service's database				
Tester	Bilginer Oral				
Date/Result	16.02.2024 / Success.				

Test ID	TCI3	Category	Integration	Severity	Critical
Objective	Uploading files to Azure Blob Storage				
Steps	• Sign in to Edux • Open up one of the existing course • Open up one of the existing chats • Upload a file • Make a fetch query for the uploaded file to Azure Blob Storage				
Expected	• The uploaded file is expected to be retrieved successfully from Azure Blob Storage				
Tester	Cahit Ediz Civan				
Date/Result	15.03.2024 / Success.				

Test ID	TCI4	Category	Integration	Severity	Critical
Objective	Uploading slides page by page as images to Azure Blob Storage				
Steps	• Sign in to Edux • Open up one of the course • Create a chat by providing some slides (pdf or pptx). The course will be created in slides mode. • The slide pages will be uploaded page by page to Azure Blob Storage and their id's will be returned. • In a loop, make a fetch query for the page images to Azure Blob Storage.				
Expected	• The slide pages are expected to be retrieved from the Azure Blob Storage successfully.				
Tester	Cahit Ediz Civan				
Date/Result	15.03.2024 / Success.				

Test ID	TCI5	Category	Integration	Severity	Major
Objective	When a chat is deleted, the files associated with it should also be deleted from Azure Blob Storage				
Steps	● Sign in to Edux ● Open up a course, find a chat that has files uploaded to it. ● Get the id's of the files that are associated with that chat. ● Delete the chat. ● Make a query to Azure Blob Storage for those files with the given id's.				
Expected	● No files are expected to be retrieved.				
Tester	Cahit Ediz Civan				
Date/Result	15.03.2024 / Success.				

Test ID	TCI6	Category	Integration	Severity	Medium
Objective	When a chat is deleted the slide histories associated with it should also be deleted from Azure Blob Storage				
Steps	● Sign in to Edux ● Open up a course, find a chat that has files uploaded to it. ● Get the id's of the slide histories of the slides that are associated with that chat. ● Delete the chat. ● Make a query to Azure Blob Storage for those slide histories with the given id's.				
Expected	● No files are expected to be retrieved.				
Tester	Cahit Ediz Civan				
Date/Result	15.03.2024 / Success.				

Test ID	TCI7	Category	Integration	Severity	Major
Objective	When a chat is deleted, the chat history associated with it should also be deleted from Azure Blob Storage				
Steps	• Sign in to Edux • Open up a course, find a chat with messages. • Get the id of the chat history. • Delete the chat. • Make a query to Azure Blob Storage for the chat history with the given id.				
Expected	• No file is expected to be retrieved.				
Tester	Cahit Ediz Civan				
Date/Result	15.03.2024 / Success.				

Test ID	TCI8	Category	Integration	Severity	Critical
Objective	When an explanation for the current slide is asked, the response should be inserted into the slide history.				
Steps	• Sign in to Edux • Open up a course. • Create a chat in slides mode (upload slides during creation). • Call the necessary endpoint for explanations on the current slide page. • Get the id of the slide history of the current slide page. • Make a query to Azure Blob Storage for the slide history with the given id. • Search for the last response (explanations) from the LLM service inside the slide history.				
Expected	• The last response from the LLM service is expected to be present in the file.				
Tester	Bilginer Oral				
Date/Result	15.03.2024 / Success.				

5.2. Non-Functional Test Cases

5.2.1. Performance

Test ID	TCNF0	Category	Performance	Severity	Low
Objective	Measure the time taken to create a new chat.				
Steps	• Login to Edux and navigate to the “Instructor/Chat” section. • Click the “Create Chat” button. • Enter a chat name and select participants. • Click the “Confirm” button to create the chat. • Measure the time taken for the system to create the chat.				
Expected	• The chat should be created within an acceptable time frame (must not exceed 5 seconds). • The chat should appear in the list of active chats. • A confirmation message should indicate successful chat creation.				
Tester	Cahit Ediz Civan				
Date/Result	24.04.2025 / Success.				

Test ID	TCNF1	Category	Performance	Severity	Medium
Objective	Measure the time taken to upload files to a chat.				
Steps	• Login to Edux and navigate to the “Instructor/Chat” section. • Select an active chat conversation. • Click the “Attach File” button. • Select a file (e.g., PDF, image, document) and confirm upload. • Measure the time taken for the system to upload the file and make it accessible.				

Expected	• The file should be uploaded within an acceptable time frame (must not exceed 8 seconds for files up to 5MB). • The file should be accessible to all participants in the chat. • A confirmation message should indicate successful file upload.
Tester	Cahit Ediz Civan
Date/Result	23.04.2025 / Success.

Test ID	TCNF2	Category	Performance	Severity	Medium
Objective	Measure the response time of the instructor chat to ensure timely communication.				
Steps	• Login to Edux with an instructor account. • Navigate to the “Instructor/Chat” section from the main menu. • Select an ongoing chat conversation or create a new one. • A student sends a message to the instructor. • Measure the time taken for the instructor to receive the message and respond.				
Expected	• The instructor's response should be retrieved within an acceptable timeframe (must not exceed 5 seconds).				
Tester	Efe Kaan Fidancı				
Date/Result	23.04.2025 / Success.				

Test ID	TCNF3	Category	Performance	Severity	Medium
Objective	Measure the time taken to create a quiz from chat interactions.				
Steps	• Login to Edux and navigate to the “Instructor/Chat” section. • Select a chat conversation containing relevant study discussions. • Click the “Create Quiz” button from the chat options menu. • Confirm the quiz generation request.				

	• Measure the time taken for the system to generate the quiz.
Expected	• The quiz should be generated within an acceptable time frame (must not exceed 10 seconds). • The quiz should contain relevant questions based on the chat discussion. • A confirmation message should appear indicating successful quiz creation.
Tester	Efe Kaan Fidancı
Date/Result	22.04.2025 / Success.

Test ID	TCNF4	Category	Performance	Severity	Medium
Objective	Measure the time taken to create a quiz from chat interactions.				
Steps	• Login to Edux and navigate to the “Instructor/Chat” section. • Select a chat conversation containing study-related content. • Click the “Create Flashcards” button from the chat options menu. • Confirm the flashcard generation request. • Measure the time taken for the system to generate the flashcards.				
Expected	• The flashcards should be generated within an acceptable time frame (must not exceed 10 seconds). • The flashcards should be relevant to the content discussed in the chat. • A confirmation message should appear indicating successful flashcard creation.				
Tester	Bilginer Oral				
Date/Result	22.04.2025 / Success.				

5.2.2. Security Test Cases

Test ID	TCS0	Category	Security	Severity	Critical
Objective	Sign up for an account with registered mail				
Steps	● Open Edux. ● Direct the page to the Sign Up page using the button Sign Up. ● Enter a mail that is already registered in the system. ● Click on the Sign Up button. ● Pop up a toaster that displays the error message “The mail is already registered for another user.”.				
Expected	The error toaster should display the message “The mail is already registered for another user.”, and the user shall not be able to sign up for an account until they write a mail that is not in the User database.				
Tester	Murat Çağrı Kara				
Date/Result	22.10.2024 / Success.				

Test ID	TCS1	Category	Security	Severity	Critical
Objective	Sign up for an account with a registered username				
Steps	● Open Edux. ● Direct the page to the Sign Up page using the button Sign Up. ● Enter a username that is already registered in the system. ● Click on the Sign Up button. ● Pop up a toaster that displays the error message “The username is already registered for another user.”.				
Expected	The error toaster should display the message “The username is already registered for another user.”, and the user shall not be able to sign up for an account until they write a username that is not in the User database.				
Tester	Murat Çağrı Kara				
Date/Result	22.10.2024 / Success.				

Test ID	TCS2	Category	Security	Severity	Critical
Objective	Sign up for an account with an invalid password				
Steps	● Open Edux. ● Direct the page to the Sign Up page using the button Sign Up. ● Enter a password not in the format of a valid password with at least eight characters up to twenty-four characters, and include at least one unique, numeric, and uppercase character in the password or check password fields. ● Click on the Sign Up button. ● Pop up a toaster that displays the error message “This is not a valid password; a valid password should contain at least eight characters up to twenty-four characters and include at least one special, numeric and uppercase character.”.				
Expected	The error toaster should display the message “This is not a valid password; a valid password should contain at least eight characters up to twenty-four characters and include at least one special, numeric, and uppercase character.” The user shall not be able to sign up for an account until they write a valid password.				
Tester	Murat Çağrı Kara				
Date/Result	22.10.2024 / Success.				

Test ID	TCS3	Category	Security	Severity	Critical
Objective	Sign up for an account with Check Password and Password Fields are not equivalent.				
Steps	● Open Edux. ● Direct the page to the Sign Up page using the button Sign Up. ● Enter a valid password in the password field. ● Enter a valid password in the check password field, which differs from the one entered in the password field. ● Click on the Sign Up button. ● Pop up a toaster that displays the error message “Two passwords do not match.”.				
Expected	The error toaster should display the message “Two passwords do not match.”, the user shall not be able to sign up for an account until two password fields have the same input.				
Tester	Murat Çağrı Kara				

Date/Result	22.10.2024 / Success.

Test ID	TCS4	Category	Security	Severity	Critical
Objective	Sign in with an invalid mail				
Steps	• Open Edux. • Enter an invalid mail that is not in the format of a mail or not a mail in the User database in the mail field. • Enter a valid password in the password field. • Click on the Sign In button. • Pop up a toaster that displays the error message “Invalid credentials, cannot find a user that has this mail in the Edux system.”				
Expected	The error toaster should display the message “Invalid credentials, cannot find a user that has this mail in the Edux system.”, the user shall not be able to sign in for an account until the user enters a valid mail belonging to a user, provided that they enter a valid password belonging to that user.				
Tester	Murat Çağrı Kara				
Date/Result	22.10.2024 / Success.				

Test ID	TCS5	Category	Security	Severity	Critical
Objective	Sign in with an invalid password				
Steps	• Open Edux. • Enter a valid mail in the mail field. • Enter an invalid password, either not in the User database or the wrong format in the password field. • Click on the Sign In button. • Pop up a toaster that displays the error message “Invalid credentials, for the given user, a wrong password has been entered.”				
Expected	The error toaster should display the message “Invalid credentials, for the given user, a wrong password has been entered.”, the user shall not be able to sign in for an account until the user enters a valid mail belonging to a user, provided that they enter a valid password belonging to that user.				
Tester	Murat Çağrı Kara				

Date/Result	22.10.2024 / Success.

Test ID	TCS6	Category	Security	Severity	Critical
Objective	Free plan users cannot use paid plan endpoints.				
Steps	• Sign in to Edux with a free plan user account. • Create a course, create a chat. • Send a message using a paid plan API endpoint (paid LLM APIs).				
Expected	The endpoint should refuse serving the user and return HTTP code “403 Forbidden”.				
Tester	Görkem Kadir Solun				
Date/Result	16.04.2025 / Success.				

Test ID	TCS7	Category	Security	Severity	Critical
Objective	Unauthorized parties cannot send API requests to private interservice communication endpoints.				
Steps	• Make an API call to one of the private endpoints of any service and do not provide an API key.				
Expected	The endpoint should refuse serving the caller and return HTTP code “403 Forbidden”.				
Tester	Bilginer Oral				
Date/Result	04.04.2025 / Success.				

Test ID	TCS8	Category	Security	Severity	Critical

Objective	Unauthorized parties cannot send API requests to private interservice communication endpoints.
Steps	● Make an API call to one of the private endpoints of any service and provide an invalid API key (the API key should not be in the accepted API keys list of the service).
Expected	The endpoint should refuse serving the caller and return HTTP code “403 Forbidden”.
Tester	Bilginer Oral
Date/Result	04.04.2025 / Success.

Test ID	TCS9	Category	Security	Severity	Critical
Objective	Private interservice communication endpoints should serve authorized services.				
Steps	● Make an API call to one of the private endpoints of any service and provide a valid API key (the API key should be in the accepted API keys list of the service).				
Expected	The endpoint should return HTTP code “200 OK”.				
Tester	Bilginer Oral				
Date/Result	04.04.2025 / Success.				

Test ID	TCS10	Category	Security	Severity	Low
Objective	Entering Invalid Email While Changing Password				
Steps	● Open Edux on the browser. ● Navigate to the “Login” page. ● At the bottom of the password section, there is “Forgot Password” click it.				

	• Fill in an invalid email address and click the “Reset” button.
Expected	The error toaster should display the message “Invalid mail address.”
Tester	Murat Çağrı Kara
Date/Result	25.04.2025 / Success.

5.2.3. Usability Test Cases

Test ID	TCU0	Category	Usability	Severity	Low
Objective	Evaluate if a new user can easily sign up for an account on Edux.				
Steps	• Recruit a new user. Provide them only with a brief overview of the platform. • Instruct the user to locate the sign-up option and create an account using valid input data. • Ask the user to verbalize their thoughts as they navigate through the Sign In and Sign Up pages (i.e., a "think aloud" protocol). • Record any difficulties or confusion regarding navigation and responsiveness during the account creation process. • After the task, request feedback on the overall experience, including ease of navigation and app responsiveness.				
Expected	The user is able to locate the sign-up option and successfully create an account with minimal or no assistance. The overall experience should be reported as intuitive, with the user reporting a positive feedback regarding the app’s responsiveness.				
Tester	Cahit Ediz Civan				
Date/Result	25.04.2025 / Success.				

Test ID	TCU1	Category	Usability	Severity	Low
Objective	Evaluate if a new user can easily sign in for an account on Edux.				
Steps	• Recruit a new user. Provide them only with a brief overview of the platform. • Instruct the user to locate the sign-in option for a created account. • Ask the user to verbalize their thoughts as they try to enter the				

	credentials(i.e., a "think aloud" protocol). ● Record any difficulties or confusion regarding navigation and responsiveness during the account creation process. ● After the task, request feedback on the overall experience, including ease of use and app responsiveness.
Expected	The user is able to locate the sign-in option and successfully sign in with minimal or no assistance. The overall experience should be reported as intuitive, with the user reporting a positive feedback regarding the app's responsiveness.
Tester	Cahit Ediz Civan
Date/Result	25.04.2025 / Success.

Test ID	TCU2	Category	Usability	Severity	Low
Objective	Assess how intuitively a user can interpret the weekly time-spent graph.				
Steps	● Provide a user who has already spent some time in Edux, so they have analytics data. ● Instruct user to sign in to their account ● Note if they understand the bar chart without extra explanation. ● Ask them if the labeling of days/hours is clear. ● After the task, request feedback on the overall experience.				
Expected	The user should be able to read and interpret the data (e.g., quickly see how many hours they studied on a given day). No or minimal confusion about the meaning of chart.				
Tester	Efe Kaan Fidancı				
Date/Result	25.04.2025 / Success.				

Test ID	TCU3	Category	Usability	Severity	Low
Objective	Assess how intuitively a user can resume their course.				
Steps	● Ask the user to find and open one of their existing courses from the dashboard. ● Instruct user to route to the Course Dashboard through Your Individual				

	Study container after user signed in via View Course button. ● Record any difficulties or confusion regarding navigation process. ● After the task, request feedback on the overall experience, including ease of use and app responsiveness.
Expected	The user should be able to find the container within **5 seconds** without needing help. The user should recognize View Course as a clickable element and not be confused by the terminology. User shall successfully go to the course dashboard without assistance.
Tester	Efe Kaan Fidancı
Date/Result	25.04.2025 / Success.

Test ID	TCU4	Category	Usability	Severity	Low
Objective	Assess how intuitively a user can access the chat feature from the dashboard.				
Steps	● Provide a user who has already spent some time in Edux and have chat data for a course. ● Ask the user to locate the Chat redirection card on the dashboard. ● Instruct the user to click on the Chat card to open the chat interface. ● Record any difficulties or confusion regarding the navigation process. ● After the task, request feedback on the overall experience, including ease of use and app responsiveness.				
Expected	The user should be able to find the Chat card within **5 seconds** without assistance. The user should recognize the Chat card as a clickable element without confusion regarding its purpose. The chat interface should open successfully, allowing the user to start a conversation without additional guidance.				
Tester	Efe Kaan Fidancı				
Date/Result	25.04.2025 / Success.				

Test ID	TCU5	Category	Usability	Severity	Low
Objective	Assess how intuitively a user can navigate to the Skill Tree from the dashboard.				
Steps	● Provide a user who has already spent some time in Edux and have chat data for a course. ● Ask the user to locate the Skill Tree redirection card on the dashboard. ● Instruct the user to click the Skill Tree card to navigate to the Skill Tree section. ● Record any difficulties or confusion regarding the navigation process. ● After the task, request feedback on the overall experience, including ease of use and app responsiveness.				
Expected	The user should be able to find the Skill Tree card within **5 seconds** without needing help. The card should be clearly identifiable as a clickable element that directs the user to view their skill progression. The Skill Tree section should load correctly, enabling the user to view and interact with their learning progress effortlessly.				
Tester	Cahit Ediz Civan				
Date/Result	25.04.2025 / Success.				

Test ID	TCU6	Category	Usability	Severity	Medium
Objective	Assess how intuitively a user can start and interact with the Chat feature.				
Steps	● Recruit a new user. Provide them only with a brief overview of the platform. ● Instruct the user to locate to the chat screen for a pre-created account with a pre-created existing Individual Study. ● Open a chat with pre-uploaded material for that Individual Study. ● Ask user to navigate in the slides inside of chat mode. ● Observe if the user understands how to type and send a message without assistance. ● Record any difficulties or confusion regarding message input, chatbot responses, or chat functionality. ● After the task, request feedback on the overall experience, including ease of use, clarity of responses, and app responsiveness.				
Expected	The message input field and send button should be easily recognizable and usable. The chatbot should respond appropriately to the user's question. The user should find the interaction smooth and intuitive, with minimal				

	confusion regarding how to ask a question or interpret the chatbot's responses.
Tester	Bilginer Oral
Date/Result	25.04.2025 / Success.

5.2.4. Document Test Cases

Test ID	TCD1	Category	Documentation	Severity	Medium
Objective	Verify the availability and correctness of the User Manual.				
Steps	● Navigate to the "Help" or "Documentation" section in Edux. ● Click the link for the User Manual. ● Download or open the User Manual. ● Verify that the document includes accurate descriptions of features, workflows, and images. ● Confirm that contact/support information is included.				
Expected	● The User Manual opens or downloads successfully. ● Content matches the latest system version. ● No broken links, grammar, or formatting issues.				
Tester	Efe Kaan Fidancı				
Date/Result	22.04.2025 / Success.				

Test ID	TCD2	Category	Documentation	Severity	Low
Objective	Ensure the Terms of Service document is accessible from the Sign-Up page.				
Steps	● Navigate to the Edux Sign-Up page. ● Locate the "Terms of Service" link. ● Click the link. ● Verify that the document opens and is readable.				
Expected	● Terms of Service opens without errors in a new window or tab. ● The content is readable and up-to-date.				
Tester	Efe Kaan Fidancı				

Date/Result	21.04.2025 / Success.

Test ID	TCD3	Category	Documentation	Severity	Low
Objective	Confirm that the Privacy Policy document is available and complies with GDPR standards.				
Steps	● Navigate to the “Privacy Policy” link in the footer of the application. ● Click the link. ● Review if GDPR compliance statements are included. ● Confirm sections on data handling, user rights, and contact information.				
Expected	● Privacy Policy loads successfully. ● GDPR compliance statements are present. ● Content is accurate and clear.				
Tester	Efe Kaan Fidancı				
Date/Result	23.04.2025 / Success.				

Test ID	TCD4	Category	Documentation	Severity	High
Objective	Verify the installation guide covers all deployment methods (Docker, Kubernetes, Azure) [This is for developers only].				
Steps	● Access the Edux API documentation ● Locate and download the installation guide. ● Review sections for Docker setup. ● Review sections for Kubernetes deployment via AKS. ● Verify cloud storage (Azure Blob) instructions are clear.				
Expected	● The guide provides step-by-step instructions. ● Commands are correct and reproducible. ● Screenshots or diagrams are present.				
Tester	Efe Kaan Fidancı				

Date/Result	21.04.2025 / Success.

Test ID	TCD5	Category	Documentation	Severity	Medium
Objective	Confirm the API documentation includes complete and accurate endpoint details [This is for developers only].				
Steps	● Access the Edux API documentation ● Check authentication endpoints. ● Verify endpoint parameters, request/response formats, and error codes are correct. ● Look for versioning information.				
Expected	● API documentation is complete, accurate, and up-to-date. ● Example requests/responses are available. ● Includes authentication/token instructions.				
Tester	Cahit Ediz Civan				
Date/Result	16.04.2025 / Success.				

6. Maintenance Plan and Details

This maintenance plan outlines the strategies and procedures for ensuring the long-term stability, performance, and scalability of the Edux platform. The plan covers both corrective and preventive maintenance approaches, including bug fixes, updates, performance optimizations, and security enhancements. It also details the roles and responsibilities of the maintenance team, the tools and processes to be used, and the schedule for regular maintenance activities.

The maintenance plan centers around four key objectives.

- Corrective maintenance focuses on swiftly addressing and resolving any bugs, errors, or issues that users report or that system monitoring identifies.
- Preventive maintenance aims to proactively update and optimize the system, working to prevent potential problems and ensure its continuous smooth operation.
- Adaptive maintenance involves modifying the system to effectively accommodate new requirements, integrate with emerging technologies, or adapt to changes in the operational landscape.

- Perfective maintenance is dedicated to enhancing the system's performance, improving its usability, and adding new functionalities based on user feedback and evolving needs through user experience improvements and ongoing performance enhancements.

The maintenance process leverages a suite of essential tools to ensure efficiency and effectiveness. Version control is managed through GitHub, facilitating code management and seamless team collaboration. Issue tracking is handled by Jira, which is used for logging, prioritizing, and meticulously tracking both bugs and development tasks [21]. CI/CD pipelines are automated using GitHub Actions, streamlining testing and deployment processes [22]. Finally, Communication within the team and for incident reporting relies on platforms like Slack and Microsoft Teams, ensuring clear and timely information exchange [23, 24].

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

Our team considered multiple engineering, social, and ethical factors in the design of Edux:

- Public Health and Welfare: Edux helps reduce cognitive overload by organizing study efforts, supporting students' mental well-being.
- Global and Cultural Considerations: With plans for multilingual support and accessibility features, Edux aims to serve a diverse, global audience.
- Environmental Considerations: By using cloud-native and scalable infrastructure, we minimize physical hardware waste and energy usage.
- Economic Considerations: Our pricing and resource usage strategies (e.g., free Gemini API tier) ensure affordability and operational sustainability.

7.1.1. Constraints

The implementation of LLM-based features necessitated careful strategies for efficient memory and API usage due to their computational demands. Economically, costs were effectively managed by leveraging free services such as MySQL, Docker, and the Gemini API. From a regulatory standpoint, compliance with GDPR and KVKK [8, 25] was ensured through the encryption of sensitive data and the secure implementation of HTTPS for all user communication. Finally, the system's architecture prioritizes scalability, enabling it to handle peak academic season loads and accommodate future feature expansions without requiring substantial modifications.

7.1.2. Standards

Adherence to established software engineering standards was a priority throughout the development process. For Coding Practices, we followed PEP8 guidelines for our Python services and utilized ESLint for maintaining consistency in the frontend codebase. Regarding Security, we implemented JWT authentication, API key validation, and adhered to OAuth2 standards to ensure robust protection. Design Documentation was created using UML diagrams

for comprehensive system modeling and OpenAPI specifications for clear service documentation. Finally, in the realm of Data Handling, we adopted GDPR and KVKK-compliant storage and access practices to safeguard user information.

7.2. Ethics and Professional Responsibilities

7.2.1. Ethical Considerations

Public Health and Safety

Edux is designed with learners' mental and intellectual well-being in mind, recognizing that effective study tools can alleviate cognitive stress. By providing adaptive study schedules, interactive flashcards, and quizzes tailored to individual needs, the platform supports students' mental health while encouraging efficient time management. Beyond cognitive benefits, Edux implements encrypted storage to protect sensitive user data such as course materials and personal credentials. These measures align with international data-protection laws like GDPR and KVKK, ensuring that both privacy and safety are upheld.

Welfare and Security

Although Edux does not directly deliver healthcare services, it contributes to social welfare by democratizing access to high-quality educational resources. By reducing barriers to personalized learning, the platform helps bridge inequalities in educational opportunity. Simultaneously, Edux's security architecture featuring transactional data handling, automated backups, and failover mechanisms safeguards user progress and ensures system reliability, reinforcing trust that learners' efforts and records will remain intact even in the face of technical disruptions.

Global and Cultural Impact

Edux is built to serve an international audience: the default interface is in English, with plans for multilingual support in future releases. Careful attention to cultural diversity underpins the platform's AI-generated content, with explicit efforts to identify and remove biases so that explanations, quizzes, and recommendations remain fair and inclusive. Free plan for underserved communities further extend the platform's reach and reinforce its commitment to equal learning opportunities worldwide.

Social, Environmental, and Economic Responsibility

Handling sensitive educational data obliges Edux to uphold the highest ethical standards in data collection and usage. The system never employs hidden data-mining techniques; any datasets used for model training such as medical or scholarly corpora are obtained transparently from reputable institutions with proper consent. Environmentally, Edux encourages efficient use of

computational and physical resources, promoting sustainable practices in affiliated institutions to minimize waste. Economically, the platform adopts a tiered subscription model, balancing affordability for regular student users with revenue streams for advanced features.

7.2.2. Professional Responsibilities

Engineering Best Practices and Compliance

Upholding professional standards in software engineering is central to Edux's long-term viability. The codebase is organized into modular microservices, each accompanied by clear documentation and consistent naming conventions. Version control through GitHub and continuous integration pipelines ensure that new features and bug fixes are deployed systematically and safely. These practices not only facilitate swift onboarding of new team members but also minimize the risk of regressions, ensuring that the platform can evolve without compromising existing functionality. Adherence to data-protection regulations and the strength of its engineering disciplines are central to Edux's professional integrity.

Collaborative Teamwork and Ethical Leadership

Professional responsibility extends beyond code to the people who build it. The Edux team employs Jira for transparent task tracking and holds weekly meetings led by a rotating Scrum Master to distribute leadership opportunities. By recording notes in shared documents, discussing challenges openly, and pairing members on complex issues, the team fosters an inclusive environment where all voices are heard and respected. This structure not only drives accountability but also cultivates a culture of mutual support, ensuring that every decision from technical design to user-experience considerations is made with ethical deliberation and collective ownership.

7.3. Teamwork Details

7.3.1. Contributing and functioning effectively on the team to establish goals, plan tasks, and meet objectives

Our team maintained a consistent rhythm of communication through weekly meetings, strategically focused on several key objectives. These meetings served as a platform to set clear milestones and define sprint goals, effectively managed through Jira. Responsibilities were carefully divided across the various components of the project, including microservices, the frontend, and cloud deployment strategies. Crucially, these sessions also ensured that every team member actively contributed their expertise and effort to all phases of the project, encompassing design, implementation, and rigorous testing.

Bilginer Oral

Bilginer Oral was an important contributor to the Edux platform from the beginning. He worked on many parts of the system, including backend services, system design and testing. His work helped the platform become stable, scalable, and easy to maintain during the whole project. From the early stages, he gave advice on architecture, especially about microservices, database structure, and CI/CD pipelines. He also helped create rules for API design, service communication, and good documentation.

On the backend side, Bilginer led the development of main services like authentication, notifications, and file handling. He used FastAPI to build secure token-based login systems and managed user permissions across services. His notification system supported both real-time and scheduled messages. His file service allowed users to upload files safely and access them. He also worked on setting up service discovery, managing settings for different environments, and using Docker for containers. This made deployments easier for the team. He improved monitoring and debugging by adding logging and alerting tools to make sure everything was ready for production.

Bilginer also worked a lot on testing and DevOps. This helped find bugs earlier and improve code quality. He wrote scripts for database seeding, migrations, and test setups, which made it easier for new developers to join and test the system. He joined code reviews and debugging sessions often, giving helpful feedback and making sure the code followed shared standards. His documentation about services and setup steps helped the team work better and made the final product more professional.

Cahit Ediz Civan

Cahit Ediz Civan was a key contributor to the platform's backend and DevOps infrastructure, playing a pivotal role in the transition from a monolithic architecture to a scalable microservices-based system. His work laid the foundation for long-term maintainability and developer efficiency across the project.

Cahit Ediz Civan began by containerizing the initial monolithic application, setting the stage for future modularization. As the system evolved, he co-led the architectural design of the microservices backend with Bilginer, ensuring a clean separation of concerns and improved scalability. After the full migration to microservices, Cahit Ediz Civan took charge of dockerizing the entire backend ecosystem, including multiple services and their associated databases, dramatically reducing the complexity of local and remote deployments. Deployed the project on Azure Kubernetes Services after creating the Kubernetes configurations. Cahit Ediz Civan efforts were instrumental in streamlining the developer workflow and improving team adoption of modern DevOps practices.

In addition to infrastructure work, Cahit Ediz Civan designed and implemented the backend service for the skill tree feature, enabling dynamic representation of user progress and learning dependencies. Cahit Ediz Civan contributions extended beyond code, as he frequently supported team members through large pull request reviews and deep dives into complex backend flows. He was recognized as a go-to problem solver within the team, often assisting with debugging, architecture questions, and technical blockers throughout the project.

Cahit Ediz Civan technical leadership, commitment to maintainability, and willingness to tackle challenging backend tasks made him a core asset to the team during a period of significant architectural transition.

Efe Kaan Fidancı

Efe Kaan Fidancı was responsible for a wide range of contributions across the frontend, backend, prompt engineering, project management, and reporting components of the final project. Notably, he was highly collaborative, working closely with other team members to ensure seamless integration of features, resolve cross-functional issues, and maintain alignment with project goals.

On the frontend, Efe Kaan Fidancı implemented and iteratively updated the user interfaces of the quiz, flashcard, and syllabus modules. These components were redesigned multiple times to align with the new microservice architecture and the Shadcn/UI component library. He introduced general responsiveness improvements, such as loading spinners, and visual enhancements, including an image loader for the sign-in page. The course dashboard's layout and appearance were also improved under his supervision. Additionally, he addressed and resolved multiple frontend bugs to ensure a smoother user experience. Beyond these tasks, he took ownership of implementing and managing updates to the project website, ensuring it remained up-to-date with the latest reports and improvements.

On the backend, Efe Kaan Fidancı implemented the initial CRUD operations for the flashcard service and assisted in developing CRUD functionalities for quizzes and syllabuses. He contributed to fixing several backend issues, such as incorrect validations, and supported the dockerization of microservices. Furthermore, he participated in the integration and debugging processes, working collaboratively with teammates to identify and resolve issues that arose during the microservices transition.

In prompt engineering, Efe Kaan Fidancı helped tailor and refine specific prompts used by the large language model (LLM) to ensure more accurate and context-aware responses aligned with the platform's educational goals. His collaborative approach ensured that these prompts were optimized through team feedback and testing.

For project management, he played an active role in generating and assigning tasks using Jira, fostering transparency and accountability within the team. He created multiple Confluence

pages to document research and planning efforts and occasionally took the lead during Scrum meetings to guide discussions and maintain progress.

Lastly, Efe Kaan Fidancı contributed to all required reports, supporting the preparation of comprehensive and well-structured documentation throughout the project. His attention to detail and teamwork ensured that all deliverables were completed to a high standard.

Görkem Kadir Solun

I am a principal contributor to the Edux platform, engaging across its full-stack lifecycle from system design and architecture through to deployment and documentation. While supporting all facets of the project, his most significant efforts centered on the Skill Tree, Profile, and Course modules, followed by pivotal work on the main page, the Syllabus & Study Plan, Notifications, Chat system, and Quiz functionality.

On the client side, I architected and implemented the dynamic Skill Tree interface using Next.js, introducing interactive progression indicators, drag-and-drop lesson ordering, and real-time unlock animations to guide learners through personalized learning paths. He overhauled the Profile section, integrating advanced settings panels, progress visuals, and preferences management, all within a reusable, component-driven UI library. For the Course pages, he crafted responsive layouts that support rich media embedding, lesson navigation controls, and contextual tooltips, ensuring a seamless study flow. I helped with the redesign of the main landing page optimizing sections, call-to-action modules, and performance-tuned image loading to boost first-impression engagement and reduce time-to-interactive.

I helped with microservices in FastAPI (with SQLAlchemy and MySQL) to power the Skill Tree's dependency logic, user progress tracking, and achievement validations. I contributed the data models and endpoints for the Profile service handling user metadata, privacy controls, and avatar management and extended the Course API to support modular lesson retrieval, versioning, and metadata filtering. My contributions to the Syllabus & Study Plan service automated weekly plan generation based on enrolled courses and user availability, complete with cron-driven notifications and adjustment hooks.

I instituted comprehensive unit and integration tests across his modules employing pytest fixtures, CI pipelines, and coverage thresholds to maintain system reliability as new features rolled out. I coordinated closely with peers, leading code reviews and pairing sessions focused on reusable patterns and performance optimizations. My documentation of API schemas, data flows, and deployment steps became the backbone of the project's final technical report.

Murat Çağrı Kara

Murat Çağrı Kara was a key developer responsible for the full-stack implementation of the *Edux* platform, delivering critical infrastructure and feature sets across frontend, backend,

and platform architecture. His contributions spanned system design, coding, debugging, optimization, and team coordination.

On the frontend, Murat Çağrı Kara designed and built a modular, responsive interface using Next.js, integrating seamlessly with the platform's evolving microservice architecture. He implemented core user flows in frontend, including authentication, subscription management, dashboard navigation, chat, skill-tree, skill-tree-quiz and notifications, ensuring a cohesive and intuitive user experience. Murat Çağrı Kara placed strong emphasis on reusability and component-driven architecture, enabling scalability as the platform grew. He also took the lead in enhancing key learning features, such as the weekly study plan, skill tree, and user profile interfaces, introducing dynamic layouts, visual indicators, and improved interaction patterns to boost user engagement. His attention to detail extended to UI polish and performance optimization, contributing significantly to the responsiveness and stability of the frontend experience.

On the backend, Murat Çağrı Kara developed multiple microservices using FastAPI with SQLAlchemy ORM and a MySQL database. He was responsible for the architecture and implementation of foundational systems such as subscription and payment logic, and user analytics tracking. He also built robust APIs for managing course data and user interactions, ensuring proper validation, error handling, and performance. His backend services were thoroughly dockerized and integrated into the platform's deployment pipeline, contributing to a more maintainable and production-ready system. Additionally, he led the resolution of several complex backend bugs, particularly in chat synchronization and course delivery mechanisms, which were critical to platform reliability.

In terms of project management and collaboration, Murat Çağrı Kara played a central role in setting development standards, managing implementation timelines, and supporting other team members. He actively participated in task planning and delegation, providing technical guidance when needed. His contributions extended to the reporting and documentation phases, where he helped compile and structure detailed technical documentation and final reports. His leadership ensured alignment between frontend and backend development efforts and maintained the overall coherence of the system throughout the project lifecycle.

7.3.2. Helping creating a collaborative and inclusive environment

Continuous collaboration and feedback were facilitated through the use of GitHub and Google Meets. Every idea brought forth was openly discussed among the team, and all decisions were ultimately reached through consensus. Furthermore, we placed a strong emphasis on ensuring the active involvement of every team member in both the design and development decision-making processes.

7.3.3. Taking lead role and sharing leadership on the team

To foster shared responsibility and diverse skill development, team members rotated leadership roles across different aspects of the project, including sprints, daily standups, and integration phases. This distributed leadership model encompassed DevOps coordination, ensuring smooth deployment and infrastructure management; System testing and QA, maintaining high standards of quality assurance; Frontend-Backend API integration, facilitating seamless communication between different parts of the application; and Documentation and presentations, ensuring clear communication of progress and technical details.

7.3.4. Meeting objectives

In assessing the progress of our Senior Design Project against the objectives outlined in the original *Project Plan* and *Functional Requirements*, we find that the Minimum Viable Product (MVP) phase has achieved a significant portion of its initial goals. The milestones we set have been partially or fully realized across most areas, though a few key features remain to be completed post-project.

Achieved Objectives

- **Interactive Learning Tools**
  Status: Met at a High Level
  Flashcards and quizzes are generated from user-uploaded content using LLMs. These tools were tested and validated for user engagement and educational value. The flashcard and quiz creation workflows were implemented successfully as described in the use case scenarios and activity diagrams in the Analysis Report.

- **Skill Tree System**
  Status: Met at a High Level
  The dynamic, interactive skill tree feature was developed and deployed, allowing users to unlock quizzes in a hierarchical learning path based on topic progression and performance. Backend logic, frontend components, and database integration were all completed as planned. Unit and integration tests validated its reliability.

- **Chat Functionality and Slide Interaction**
  Status: Met at a High Level
  Users can upload slides, receive page-by-page explanations, and engage in LLM-based conversations to enhance comprehension. Chat features are integrated into individual studies, with UI and logic working as intended.

- **Study Schedule Generation**
  Status: Met at a High Level
  The weekly study plan generation is operational. It updates dynamically upon syllabus

uploads and takes user performance into account. This matches the objectives outlined in the original plan.

- **System Stability, Usability, and Compliance**
  Status: Confirmed via Testing and User Feedback
  The system was tested for reliability and user satisfaction. It met the usability benchmark of 95%, amongst the users that used our app, task completion in under three clicks. GDPR-compliant data security measures were implemented.

Unmet Objectives

- **Retrieval-Augmented Generation (RAG)**
  Status: Partially Designed, Not Operational
  The RAG component, which was expected to generate context-aware and personalized practice questions, was not completed. While preliminary design and planning were done and its logic and models were outlined and UI components partially designed; implementation was postponed and will be addressed beyond the scope of Senior Design Project.

- **Report Code Checker**
  Status: Not implemented
  The feature intended to ensure consistency between student-submitted code and reports was not implemented. Before starting Senior Design Project II, this feature was discarded as it was not found useful in the context of Edux.

Summary

The Edux platform has successfully delivered the majority of its core features as outlined in the original analysis and requirements. While two components—RAG and the Report Code Checker—remain incomplete, the system fulfills its MVP goals and lays the foundation for further development.

7.4. New Knowledge Acquired and Applied

To successfully design and implement the advanced features of Edux, our team needed to deepen its understanding in several technical and pedagogical domains. One of the primary areas of learning was Large Language Models (LLMs). Through workshops, and individual study sessions, we explored the architecture and use cases of modern LLMs, including their capabilities for summarization, question generation, and interactive assistance. We gained hands-on experience integrating the Gemini API and evaluated alternatives such as OpenAI and Hugging Face. This knowledge was crucial in enabling features such as AI-generated flashcards, quizzes, contextual explanations, and the foundation of our chat-based learning assistant [26, 27].

Another critical area of expertise developed during the project was personalized learning tool integration. We investigated methods to transform static learning resources into dynamic study aids tailored to each learner's progress and needs. Concepts such as adaptive scheduling, content chunking, and performance-based prioritization were studied and implemented in our skill tree and personalized study schedule modules. Our understanding of educational psychology and personalized learning strategies grew as we examined academic research and industry implementations, which informed how Edux supports learners with different styles and schedules.

We also delved deeply into progress tracking methodologies, a key component in maintaining learner engagement and promoting continuous improvement. By analyzing dynamic data from user interactions such as quiz performance, study duration, and content revisitation we created models that offer real-time progress updates. This required learning about behavioral analytics, UX best practices, and data visualization techniques. Our application of these insights is visible in the dashboards and performance metrics available to both learners and instructors.Beyond technical domains, we emphasized collaborative learning practices to facilitate continuous team-wide growth. We conducted regular knowledge-sharing sessions where team members presented their findings, demoed prototypes, or discussed implementation challenges. This culture of shared learning not only accelerated our problem-solving ability but also ensured that every member contributed to and understood the evolving architecture of Edux. In parallel, we actively followed trends and developments in educational technology and AI by subscribing to journals, joining online communities, and attending relevant conferences. These efforts helped us stay aligned with best practices and emerging standards in data privacy (e.g., GDPR/KVKK), AI safety, and scalable architecture design. As a result, our system reflects not only academic rigor and engineering discipline but also ethical and practical considerations appropriate for a modern educational platform.

Through strategic planning, hands-on implementation, and continuous learning, our team successfully acquired and applied new knowledge across multiple dimensions. These efforts allowed us to transform our initial concept into a robust, forward-thinking solution that aligns with the needs of today's learners and educators.

8. Conclusion and Future Work

Throughout this senior design project, we have successfully architected, implemented, and validated Edux, an AI-driven learning platform that unifies content ingestion, personalized study aids, and interactive analytics within a scalable microservices framework. Beginning with a comprehensive requirements analysis, we decomposed the system into modular services like Authentication, User, Course, Chat, GenAI, and FileManager each secured via JWT and API-key

protocols and orchestrated on Azure Kubernetes Service [28]. On the frontend, we delivered a responsive React.js interface that empowers learners to upload a wide range of materials, engage with AI-powered explanations, and generate flashcards, quizzes, and skill trees. The backend transformation into RESTful microservices has not only boosted maintainability and performance under peak loads, but also laid the groundwork for smooth horizontal scaling. Functional and non-functional test suites validated our features against usability, reliability, security, and performance goals. Consideration of public health, safety, environmental, cultural, and economic factors further ensured that Edux meets both technical and societal standards. Our collaborative team dynamics and clear version-control practices enabled us to iterate efficiently, incorporate user feedback, and deliver a robust platform ready for real-world adoption.

Looking ahead, there are numerous avenues to extend and enrich Edux's capabilities. First, we plan to broaden media support by integrating audio and video resources: automatic transcription, summarization, and context-aware Q&A on lecture recordings would cater to varied learning styles. A native mobile application with offline caching would enable uninterrupted study in low-connectivity environments. To deepen personalization, we envision incorporating reinforcement-learning algorithms that adapt the sequence and difficulty of quizzes and flashcards based on real-time performance metrics.

On the social front, adding synchronous collaboration features such as group study rooms with live chat, shared whiteboards, and peer-review workflows can foster community and peer learning. An instructor dashboard with advanced analytics and early-warning alerts would empower educators to intervene proactively when learners struggle. Extending interoperability through LTI or RESTful APIs would allow Edux to plug into institutional LMS platforms (e.g., Moodle, Canvas) and third-party content repositories, broadening its reach.

Future iterations could explore Retrieval-Augmented Generation (RAG) with domain-specific knowledge bases to produce highly contextualized practice questions and project feedback. Furthermore it can have the functionality of Report Code Checker, which is intended for instructors to evaluate consistency between student code and reports. Incorporating generative models for code evaluation and plagiarism detection would streamline assignment grading and uphold academic integrity. Gamification elements such as badges, achievement unlocks, and leaderboards could boost motivation and sustained engagement. Finally, expanding multilingual support, accessibility features (voice interfaces, high-contrast themes), and privacy-preserving analytics will ensure that Edux grows into an inclusive, secure, and globally adaptable learning ecosystem [29].

9. Glossary

AKS (Azure Kubernetes Service)

Azure Kubernetes Service (AKS) is a managed Kubernetes service provided by Microsoft Azure. It simplifies the deployment, scaling, and management of Kubernetes clusters, integrating with Azure's security, monitoring, and networking features.

API (Application Programming Interface)

An Application Programming Interface (API) is a set of protocols and tools that allow different software applications to communicate with each other. APIs enable developers to access specific functionalities or data from external services without understanding their internal workings.

Azure Blob Storage

Azure Blob Storage is Microsoft's cloud-based object storage solution designed for unstructured data such as images, videos, backups, and logs. It provides high availability, scalability, and security for storing large amounts of data [7].

D

Docker

Docker is an open-source platform that enables developers to automate the deployment of applications within lightweight, portable containers. These containers package the application along with their dependencies, ensuring consistency across different environments.

G

GDPR (General Data Protection Regulation)

The General Data Protection Regulation (GDPR) is a comprehensive data protection law enacted by the European Union to safeguard individuals' data and privacy. Implemented in 2018, it imposes strict guidelines on data collection, processing, and storage, with significant penalties for non-compliance [8].

H

HTTPS (HyperText Transfer Protocol Secure)

HTTPS is an internet communication protocol that ensures secure data transfer between a user's web browser and a website. It is an encrypted version of HTTP (HyperText Transfer Protocol) and is widely used to protect sensitive information such as login credentials, payment details, and personal data from cyber threats.

I

ID (Identifier)

An Identifier (ID) is a unique symbol or sequence assigned to an entity to distinguish it from others. IDs are essential in databases, programming, and various systems to efficiently reference and manage specific records or objects.

K

Kubernetes (K8s)

Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications. It provides load balancing, service discovery, and self-healing capabilities, making it ideal for managing large-scale containerized workloads.

L

LLM (Large Language Model)

A Large Language Model (LLM) is an advanced artificial intelligence system trained on extensive text datasets to understand and generate human-like language. These models are pivotal in natural language processing tasks, including text generation, translation, and sentiment analysis.

M

MySQL

MySQL is an open-source relational database management system (RDBMS) that uses SQL for accessing and managing data. It is widely used for web applications, data storage, and processing large-scale structured data.

R

RAG (Retrieval Augmented Generation)

Retrieval Augmented Generation (RAG) combines large language models with external data sources to enhance the accuracy and relevance of generated content. RAG systems can produce more informed and contextually appropriate responses by retrieving pertinent information from databases or documents.

S

SHA-256 (Secure Hash Algorithm 256-bit)

SHA-256 is a cryptographic hash function used widely in various security applications and protocols, including SSL certificates and blockchain technology. It provides a fixed-size 256-bit (32-byte) hash that is nearly unique for different inputs and is used for ensuring data integrity.

T

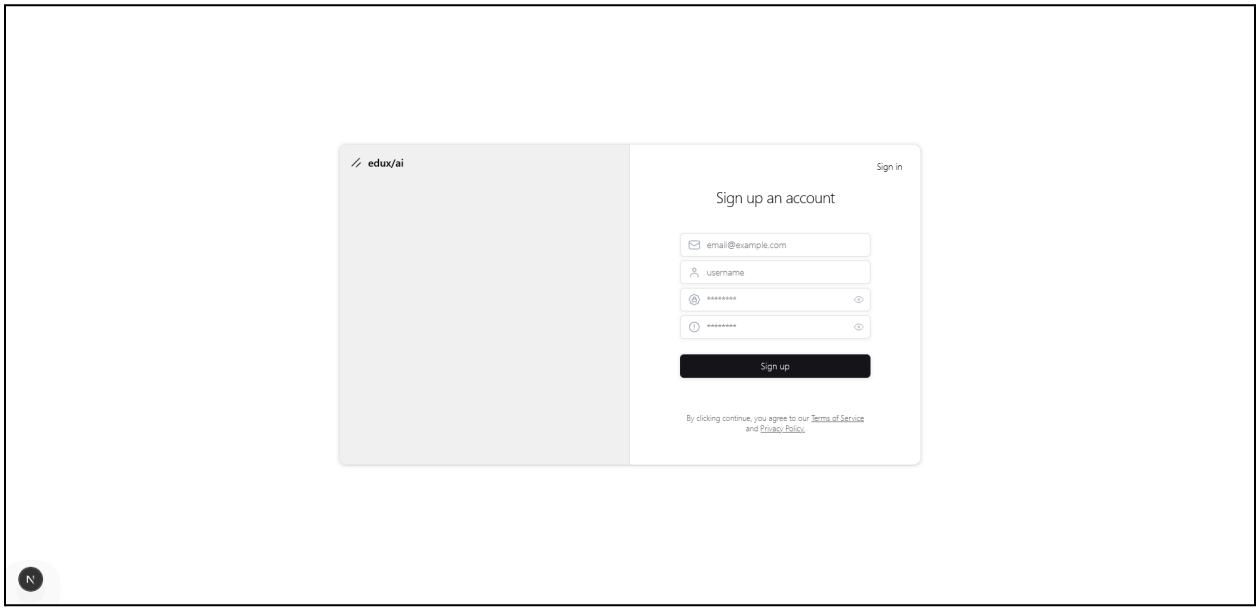
TPS (Transactions Per Second)

Transactions Per Second (TPS) is a metric used to measure the number of transactions a system can process in one second. It's commonly used to assess the performance and scalability of databases, networks, and other transactional systems.

Edux User Manual

1 - Logging into Edux

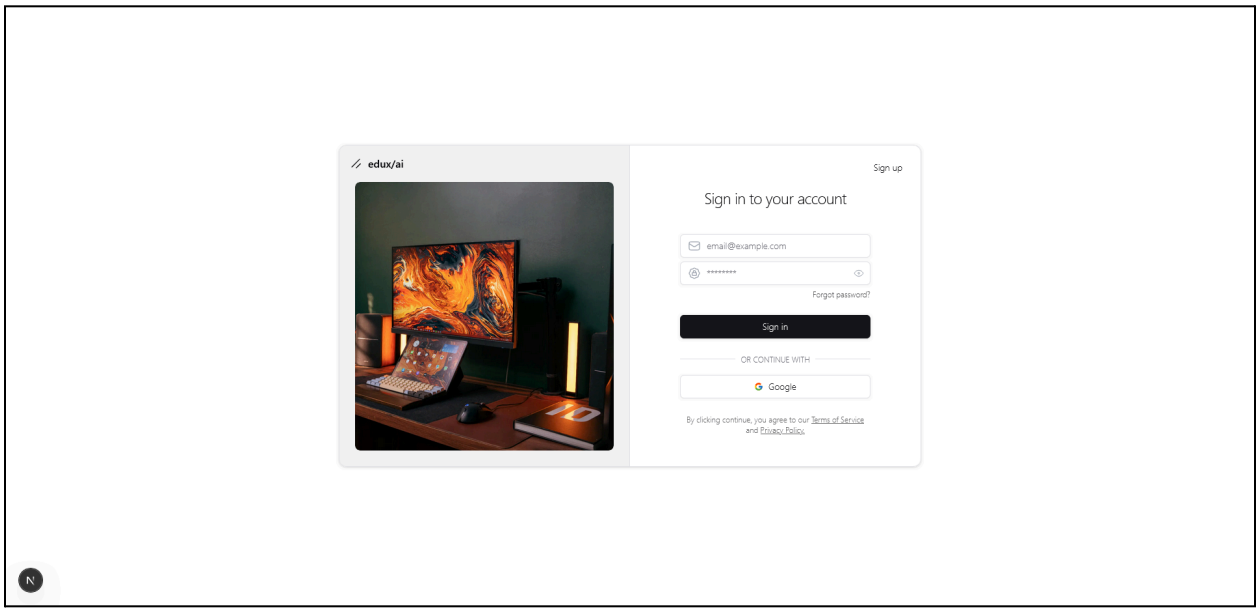
1.1 - Sign-up



In the top left corner, the logo of Edux is displayed, identifying the website's brand. Situated in the top right corner is a "Sign in" link, providing users with access to either create a new account or log in to an existing one. Via the page new users may create a new account. This form includes fields for users to input their email address, desired username, and password, along with a confirmation field for the password.

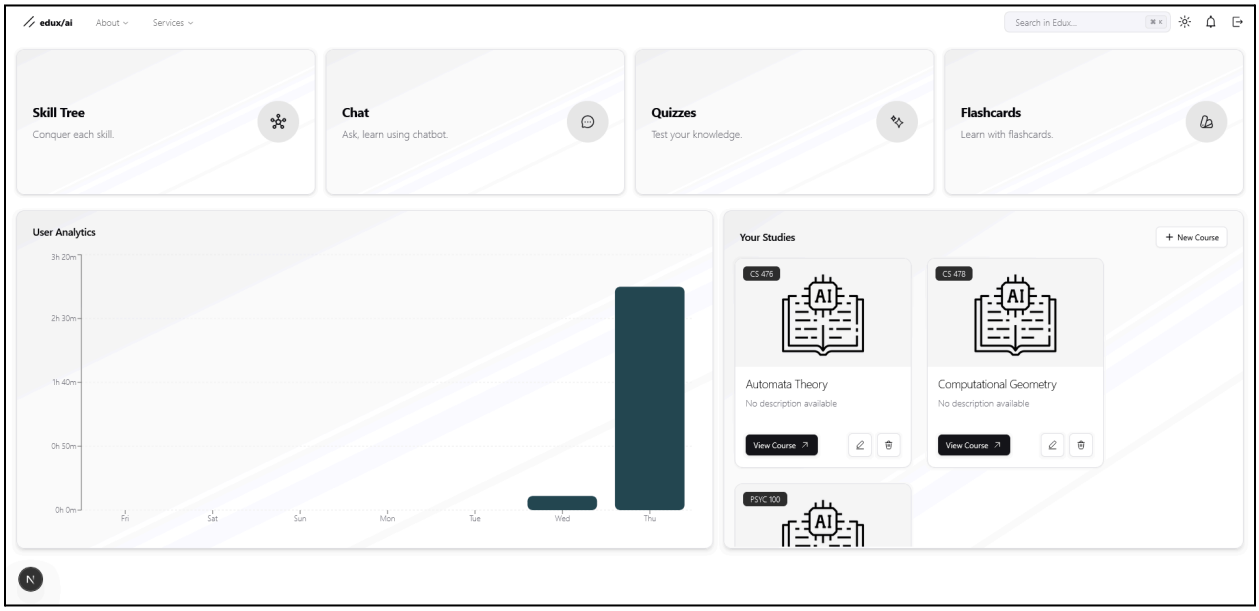
At the bottom of the form, a "Sign up" button allows users to submit their details and establish an account. Directly below this button, a note informs users that by clicking "continue," they are agreeing to the website's Terms of Service and Privacy Policy, a standard practice for ensuring user awareness of the legal agreements.

1.2 - Sign-in



This depicts the sign-in page for the Edux website, designed to facilitate user access to their accounts. At the top left, the Edux logo represents the brand, while the top right offers a "Sign up" link for new users. The central sign-in form includes fields for entering an email address and password, accompanied by a "Sign in" button and a "Forgot password?" link for password recovery. Users also have the option to log in conveniently using their Google accounts through the "Continue with Google" button. The page emphasizes user agreement to the Terms of Service and Privacy Policy upon login. Additionally, the left side features an image displayer, enhancing the professional and inviting atmosphere of the platform.

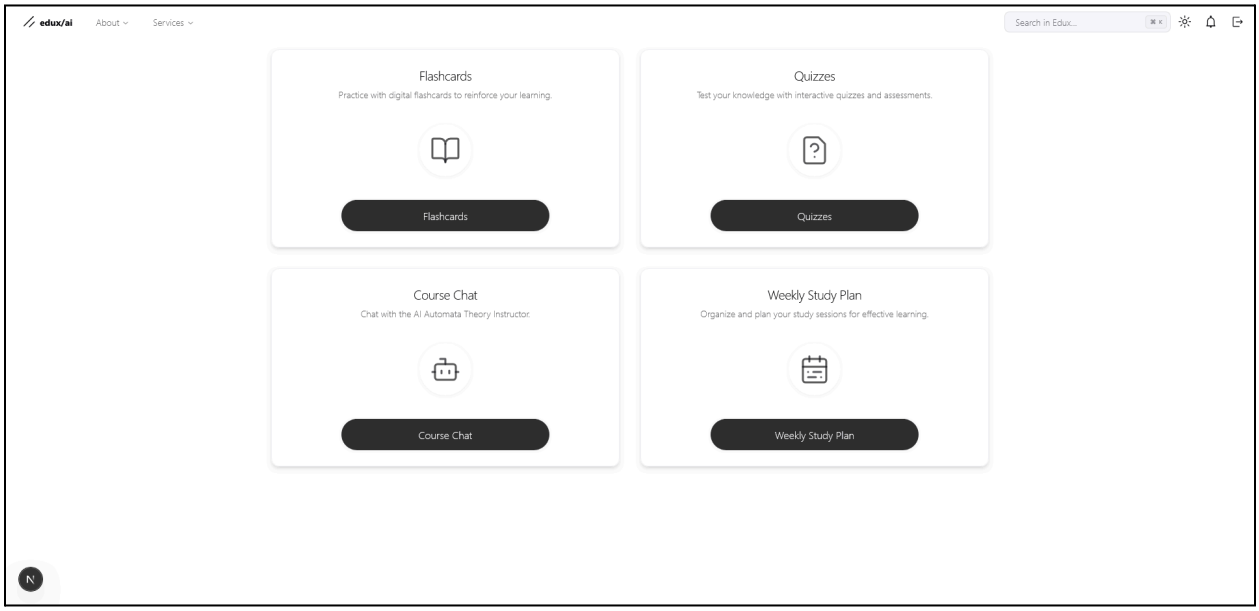
2 - Edux Homepage



This page displays the dashboard of the Edux platform, designed to provide users with a comprehensive and interactive learning experience. At the top, the navigation bar includes links to "About" and "Services," along with a search function and icons for notifications and a profile menu. The main section features several educational tools: "Skill Tree" for structured learning, "Chat" for interactive learning via chatbot, "Quizzes" for knowledge testing, and "Flashcards" for memorization. Additionally, a user analytics section presents a bar graph of weekly activity, highlighting engagement patterns. The "Your Studies" section lists user generated courses such as "Automata Theory," "Computational Geometry," and "Algorithms," with options to view each course and add new ones.

This dashboard offers a user-friendly interface, allowing easy navigation and access to various learning resources and progress tracking.

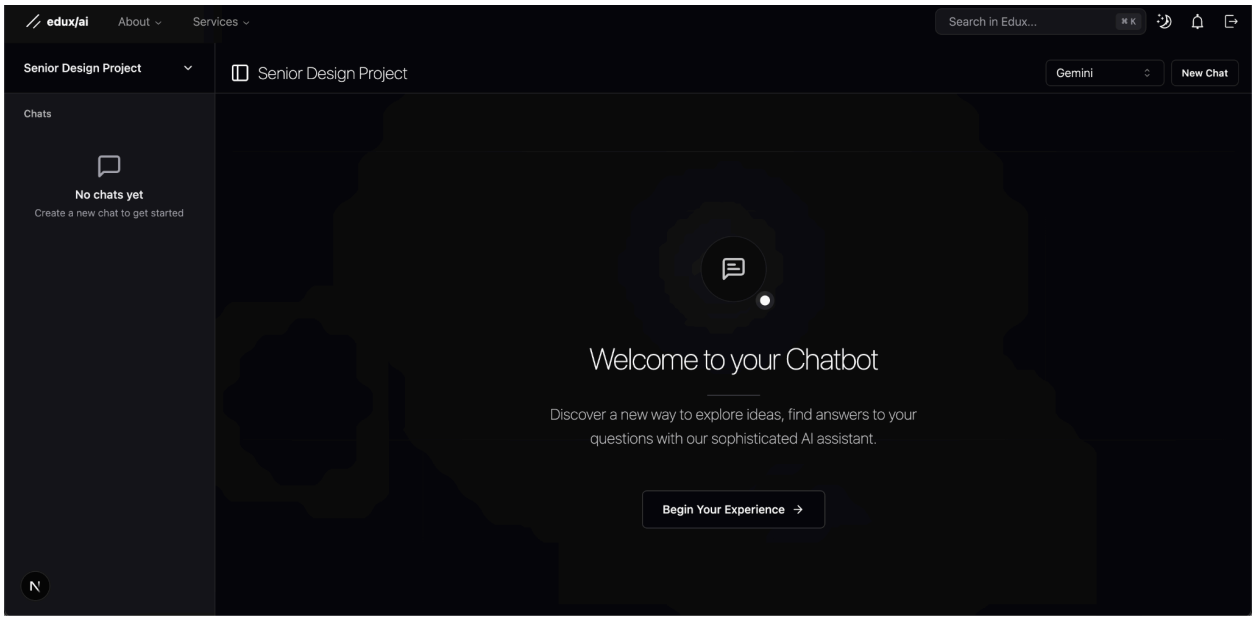
3 - Edux Course Dashboard

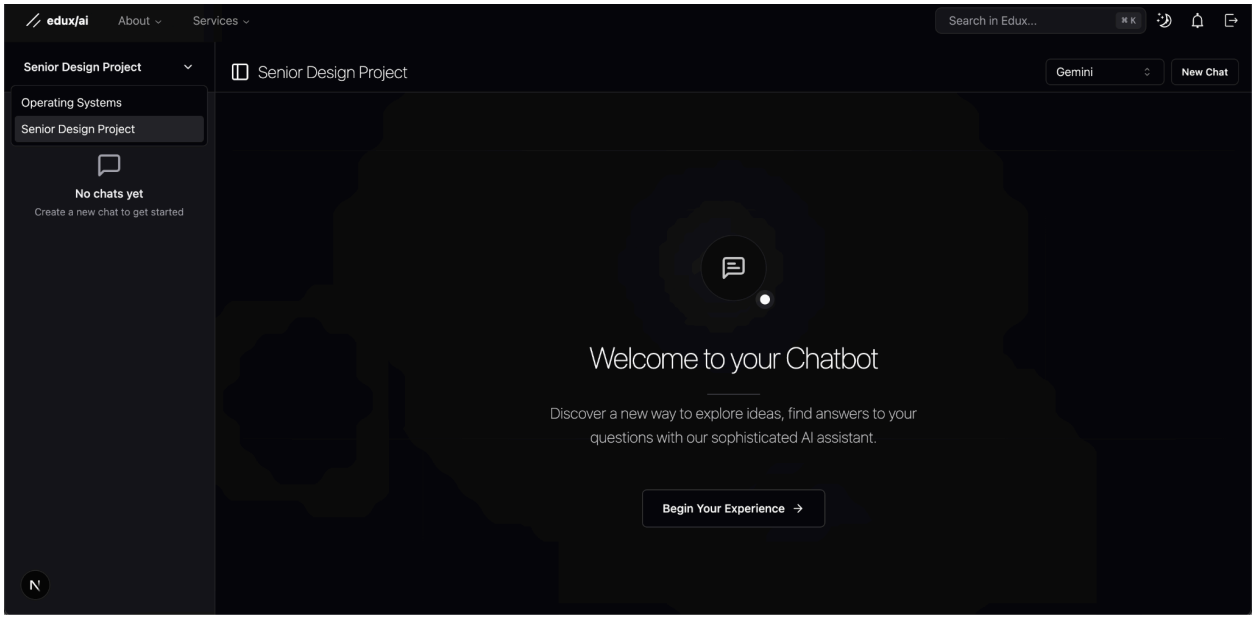


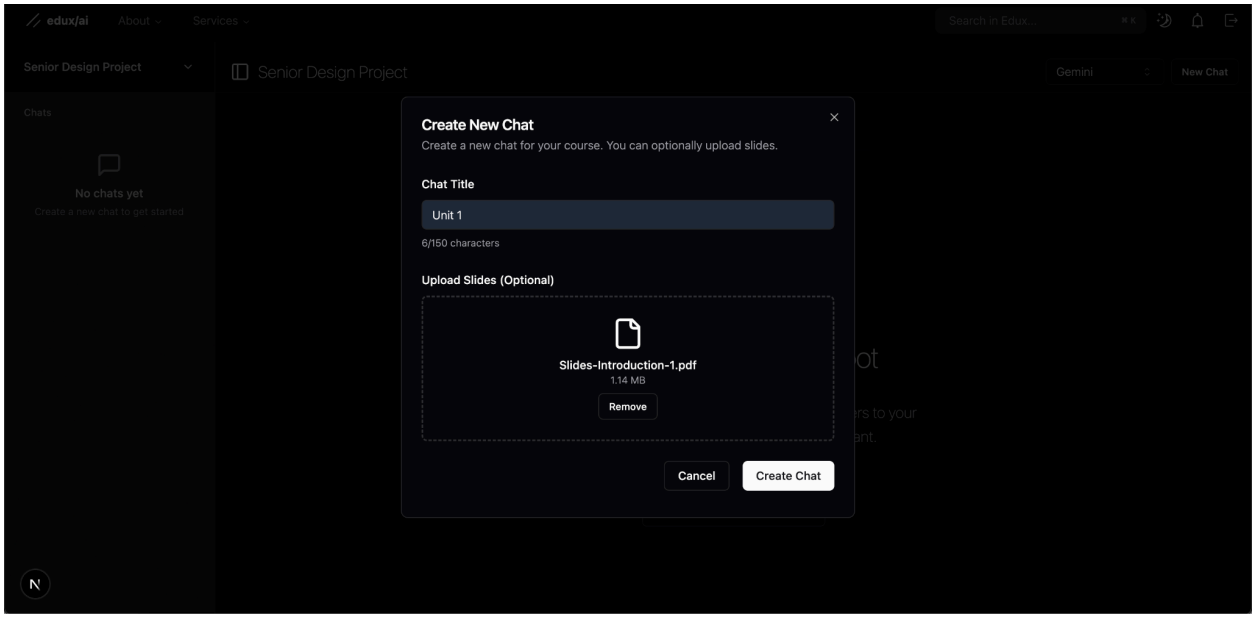
Welcome to your edux course dashboard! This area provides access to four key study tools:

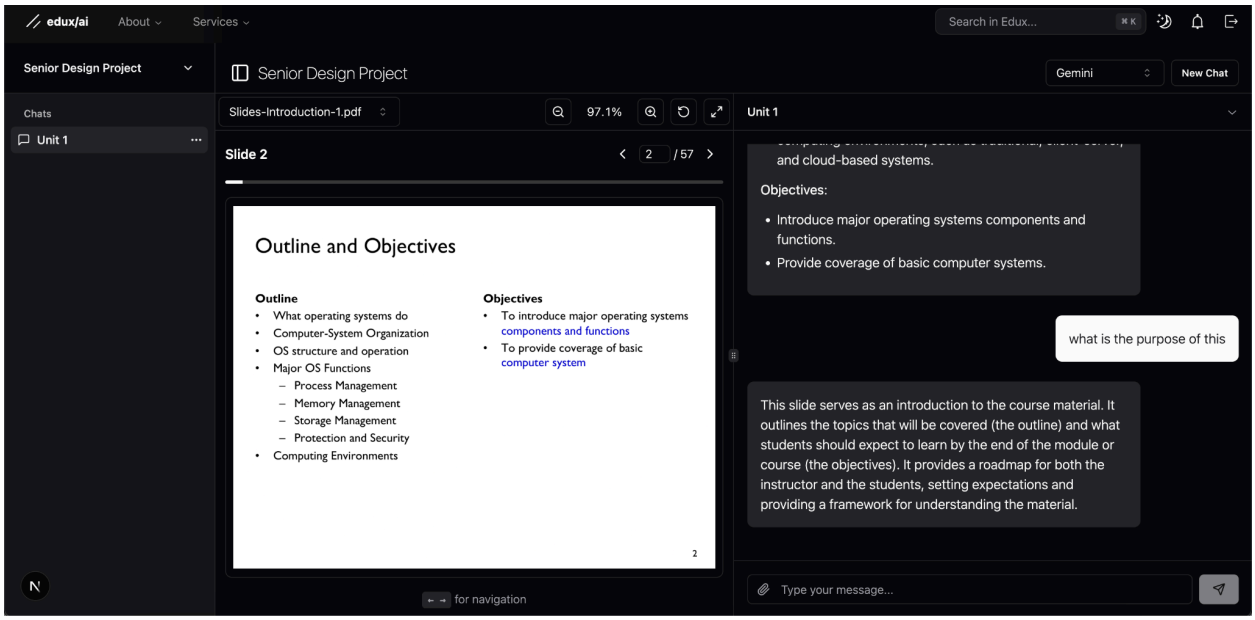
- Flashcards: These digital, flippable cards are AI-generated based on your specific chat interactions, offering a dynamic way to review concepts. You'll find them within each chat, with the possibility of having none or several.
- Quizzes: Similar to the flashcards, these multiple-choice assessments are tailored to individual chat sessions, allowing you to test your understanding of the material discussed.
- Instructor (LLM-Chat): Engage directly with our AI-powered chat to explore lecture slides and deepen your understanding of the course content.
- Syllabus: Here, you can leverage the LLM to create a personalized study plan based on the course syllabus.

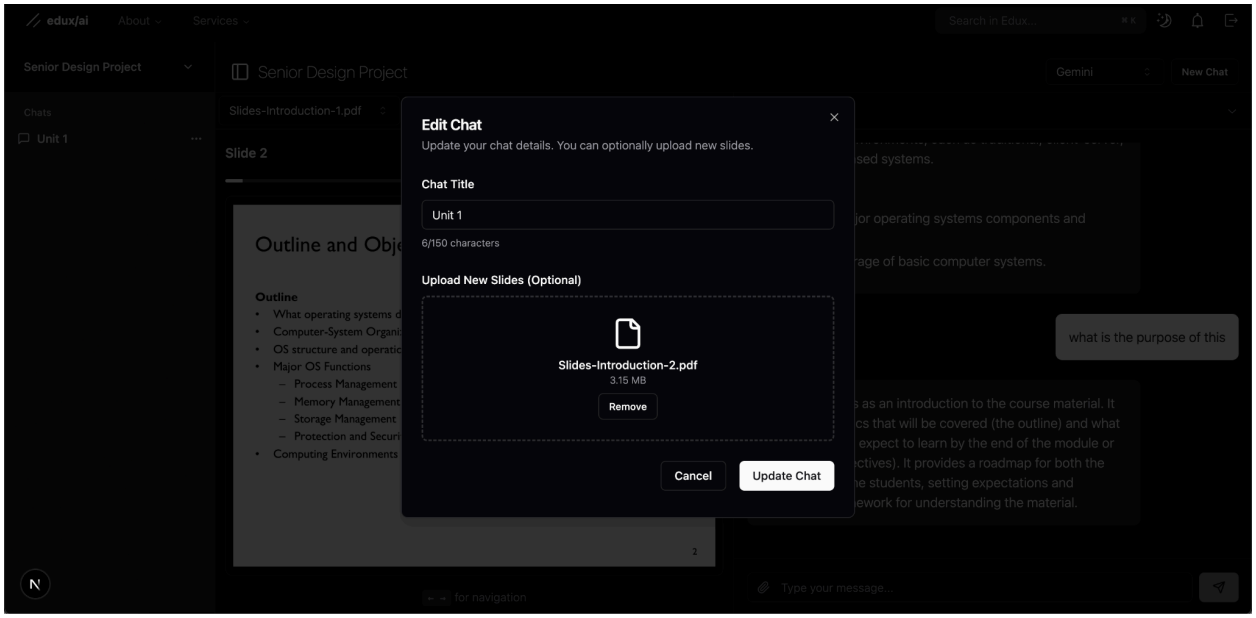
4 - Edux Chat Window

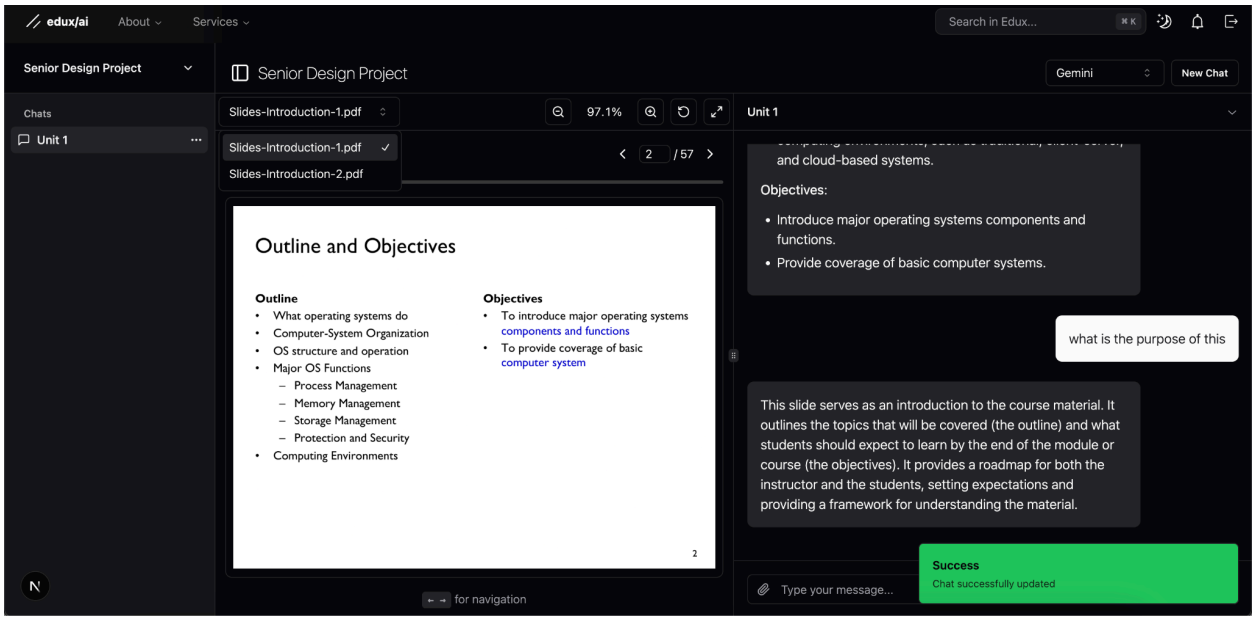


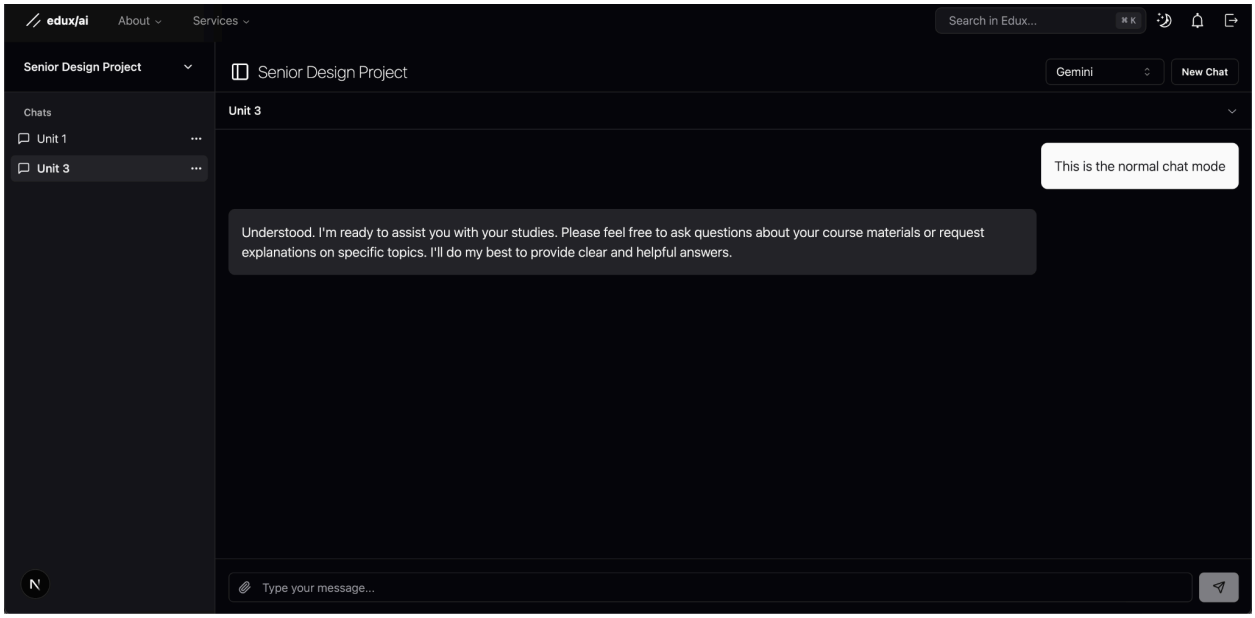


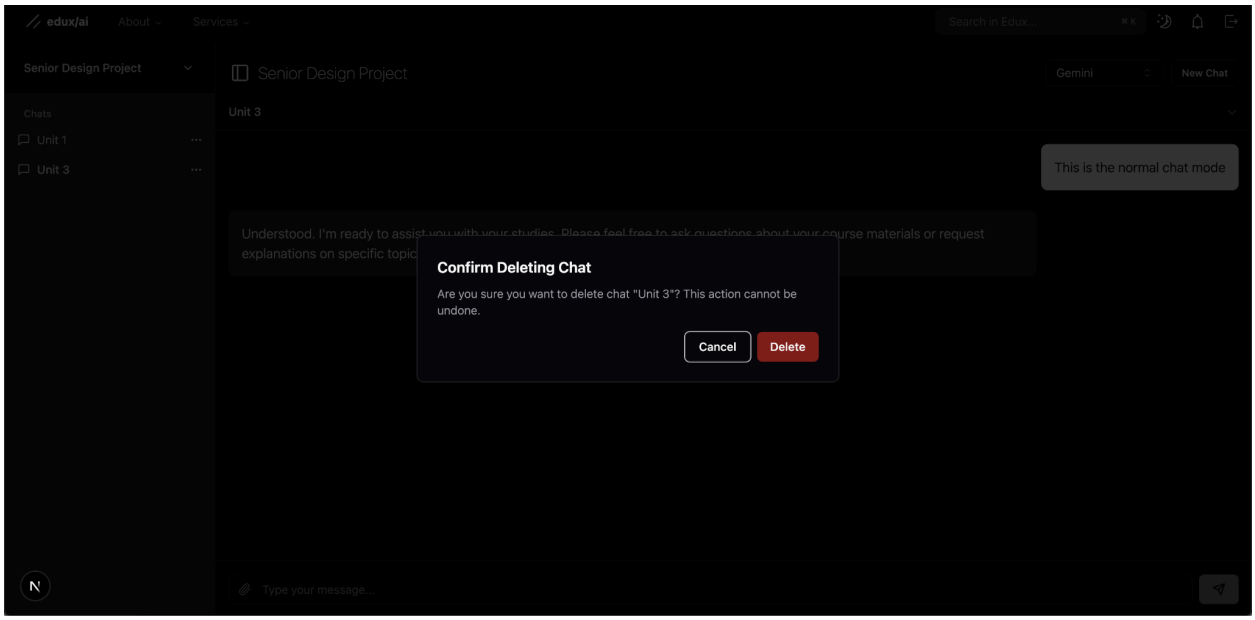

edux/ai
About
Services
Search in Edux...
Senior Design Project
Chats
No chats yet
Create a new chat to get started
Senior Design Project
Gemini
New Chat
Create New Chat
Create a new chat for your course. You can optionally upload slides.
Chat Title
Unit 1
6/150 characters
Upload Slides (Optional)
Slides-Introduction-1.pdf
1.14 MB
Remove
Cancel
Create Chat


edux/ai
About
Services
Search in Edux...
Senior Design Project
Chats
Unit 1
Senior Design Project
Gemini
New Chat
Slides-Introduction-1.pdf
97.1%
Unit 1
Slide 2
2
/ 57
and cloud-based systems.
Objectives:
Introduce major operating systems components and functions.
Provide coverage of basic computer systems.
Outline and Objectives
Outline
What operating systems do
Computer-System Organization
OS structure and operation
Major OS Functions
Process Management
Memory Management
Storage Management
Protection and Security
Computing Environments
Objectives
To introduce major operating systems components and functions
To provide coverage of basic computer system
2
what is the purpose of this
This slide serves as an introduction to the course material. It outlines the topics that will be covered (the outline) and what students should expect to learn by the end of the module or course (the objectives). It provides a roadmap for both the instructor and the students, setting expectations and providing a framework for understanding the material.
for navigation
Type your message...


edux/ai
About
Services
Search in Edux...
Senior Design Project
Chats
Unit 1
Senior Design Project
Gemini
New Chat
Slides-Introduction-1.pdf
Slide 2
Edit Chat
Update your chat details. You can optionally upload new slides.
Chat Title
Unit 1
6/150 characters
Upload New Slides (Optional)
Slides-Introduction-2.pdf
3.15 MB
Remove
Cancel
Update Chat
Outline and Obj
Outline
What operating systems d
Computer-System Organi
OS structure and operatic
Major OS Functions
Process Management
Memory Management
Storage Management
Protection and Securi
Computing Environments
what is the purpose of this
Type your message...
for navigation


edux/ai
About
Services
Search in Edux...
Senior Design Project
Chats
Unit 1
Senior Design Project
Gemini
New Chat
Slides-Introduction-1.pdf
97.1%
Slides-Introduction-1.pdf
Slides-Introduction-2.pdf
2 / 57
Outline and Objectives
Outline
What operating systems do
Computer-System Organization
OS structure and operation
Major OS Functions
Process Management
Memory Management
Storage Management
Protection and Security
Computing Environments
Objectives
To introduce major operating systems components and functions
To provide coverage of basic computer system
2
for navigation
Unit 1
and cloud-based systems.
Objectives:
Introduce major operating systems components and functions.
Provide coverage of basic computer systems.
what is the purpose of this
This slide serves as an introduction to the course material. It outlines the topics that will be covered (the outline) and what students should expect to learn by the end of the module or course (the objectives). It provides a roadmap for both the instructor and the students, setting expectations and providing a framework for understanding the material.
Type your message...
Success
Chat successfully updated





The Edux Chat feature is designed as an interactive learning platform where you can engage with a sophisticated AI assistant to explore ideas and get answers related to your coursework.

1. Accessing Chat:

- From the main Edux Homepage, you can access the general "Chat" feature.
- Within a specific course, like "Senior Design Project," there's a dedicated chat section.

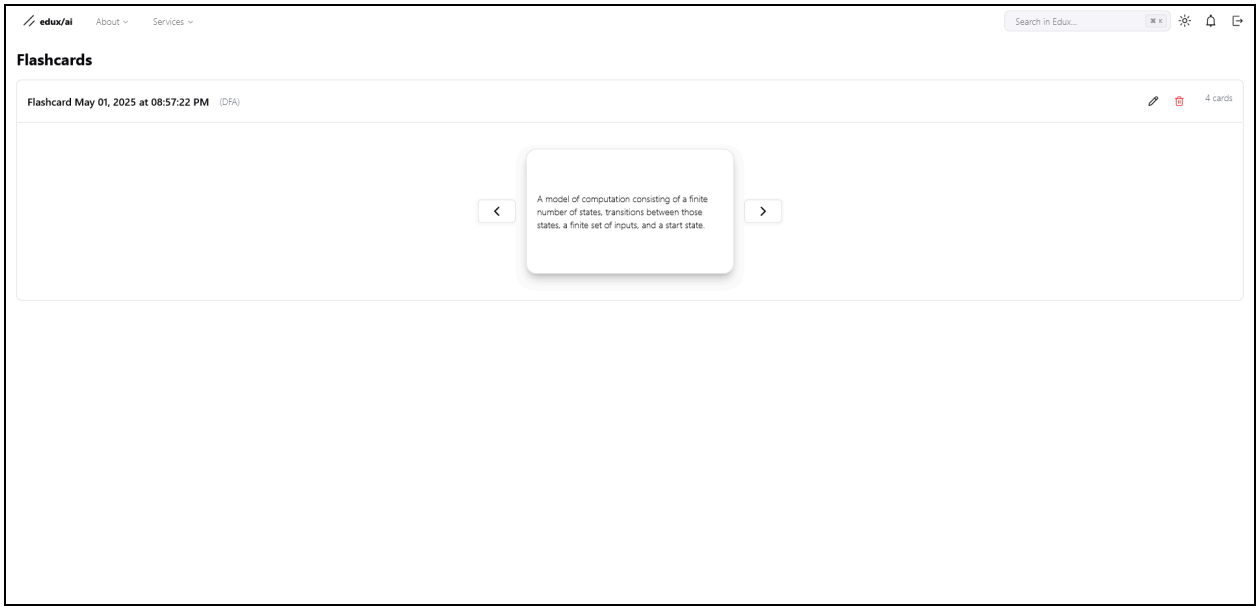
2. Chat Interface Layout:

- **Course Context:** The top of the chat interface clearly shows the course context you are currently in (e.g., "Senior Design Project").
- **Chat List Sidebar (Left):** This panel on the left lists all the individual chat sessions you've created within that specific course context. Initially, it might show "No chats yet". You can select different chats (e.g., "Unit 1", "Unit 3") from this list.
- **Main Chat Area (Center):** This is where the conversation with the AI assistant takes place. It displays both your messages and the AI's responses.
- **Slide Viewer (Optional):** If you've uploaded slides for a chat, they can be displayed alongside the conversation, allowing you to reference specific content like outlines and objectives while chatting.
- **Top Bar:** Contains navigation elements like "About" and "Services," a search bar for Edux, and potentially profile/notification icons. It also has buttons like "New Chat".

3. Key Features:

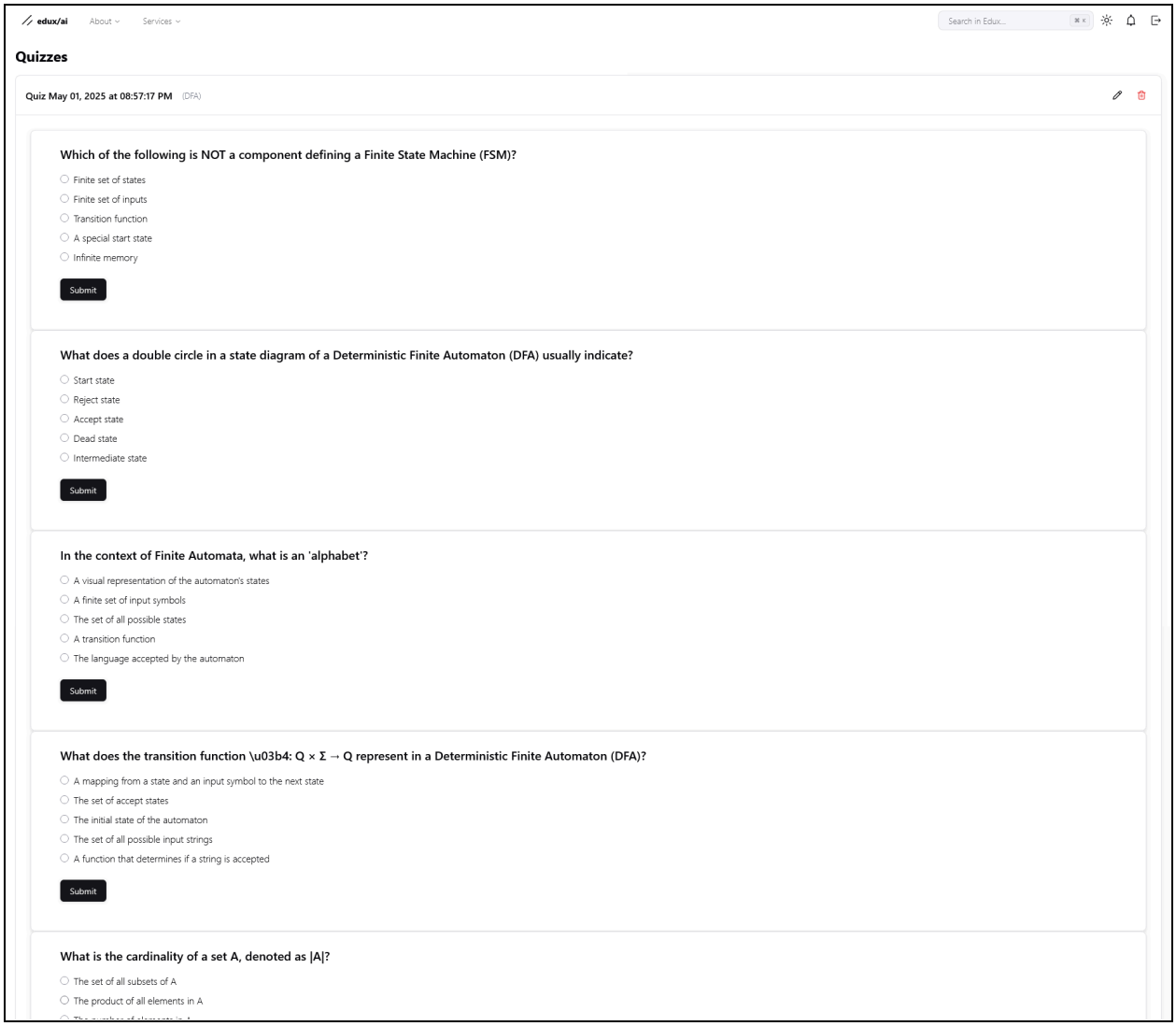
- **Starting a Chat:** When you first enter the chat section for a course, you might see a welcome message prompting you to "Begin Your Experience".
- **Creating New Chats:** Click the "New Chat" button. You'll be prompted to provide a "Chat Title" (e.g., "Unit 1") and can optionally upload presentation slides relevant to that chat topic.
- **AI Interaction:** Type your questions or prompts into the message box at the bottom. The AI assistant can help explain course materials, answer questions about specific topics shown on slides, or discuss concepts.
- **Editing Chats:** You can modify existing chats. The "Edit Chat" option allows you to update the chat title and upload different or additional slides.
- **Deleting Chats:** You can remove chats you no longer need. The system will ask for confirmation because deletion is permanent.
- **Changing Courses:** The chat sidebar *lists the chats within the currently selected course* ("Senior Design Project" in the examples). You can change the course from the upper left corner.

5 - Edux Course Quizzes



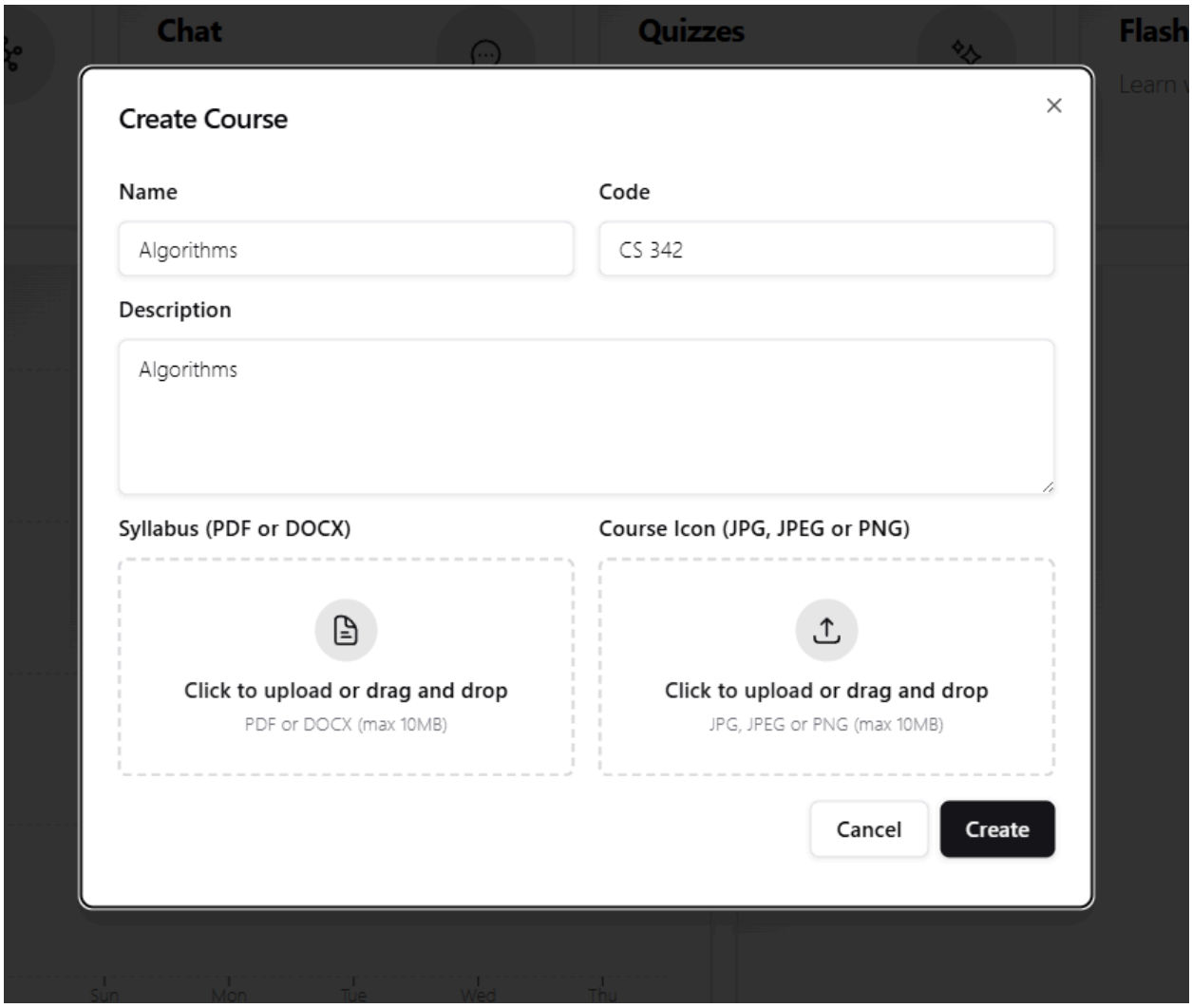
On the Flashcards page, you'll find AI-generated flashcards specifically designed to reinforce your understanding of the course material based on your lecture explanation chats. This section displays all flashcards created for the course, allowing you to rename or delete them as needed. Simply click on a card to reveal its content.

6 - Edux Course Flashcards

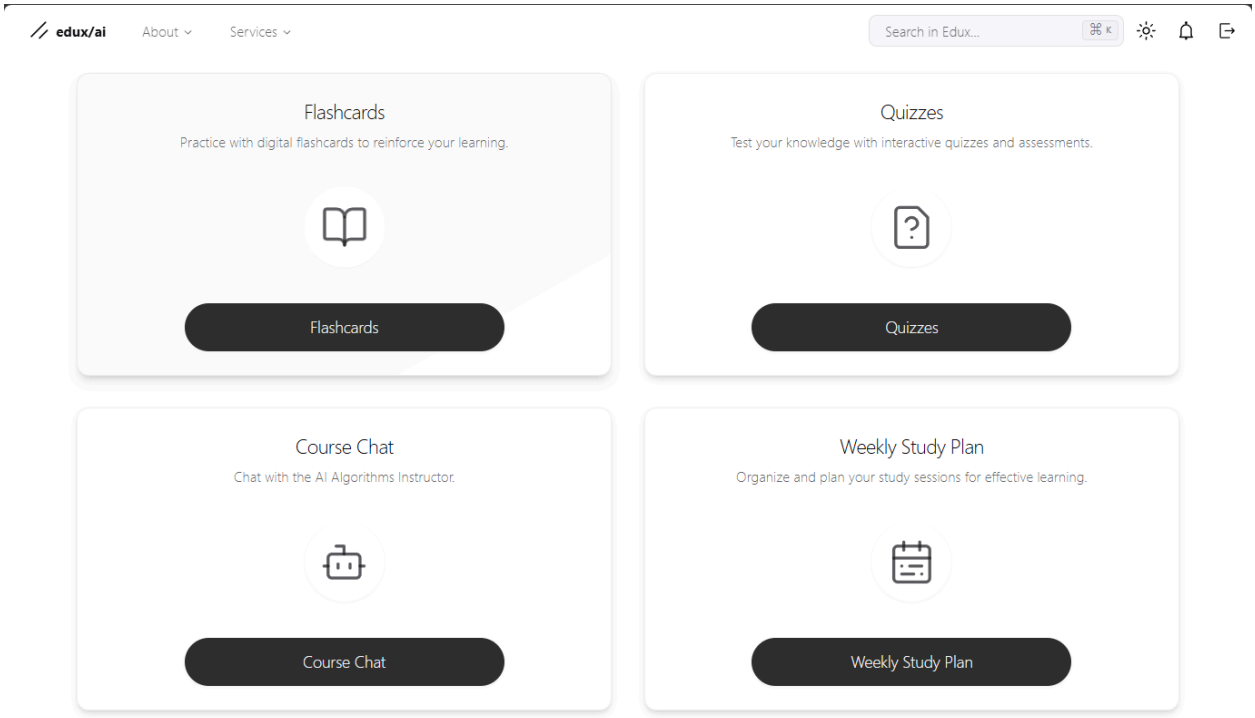


The Quizzes page presents AI-generated multiple-choice quizzes derived from your lecture explanation chats, enabling you to assess your comprehension of specific topics. Here, you can access all quizzes created for the course. You have the option to rename or delete quizzes. Click on a quiz to begin.

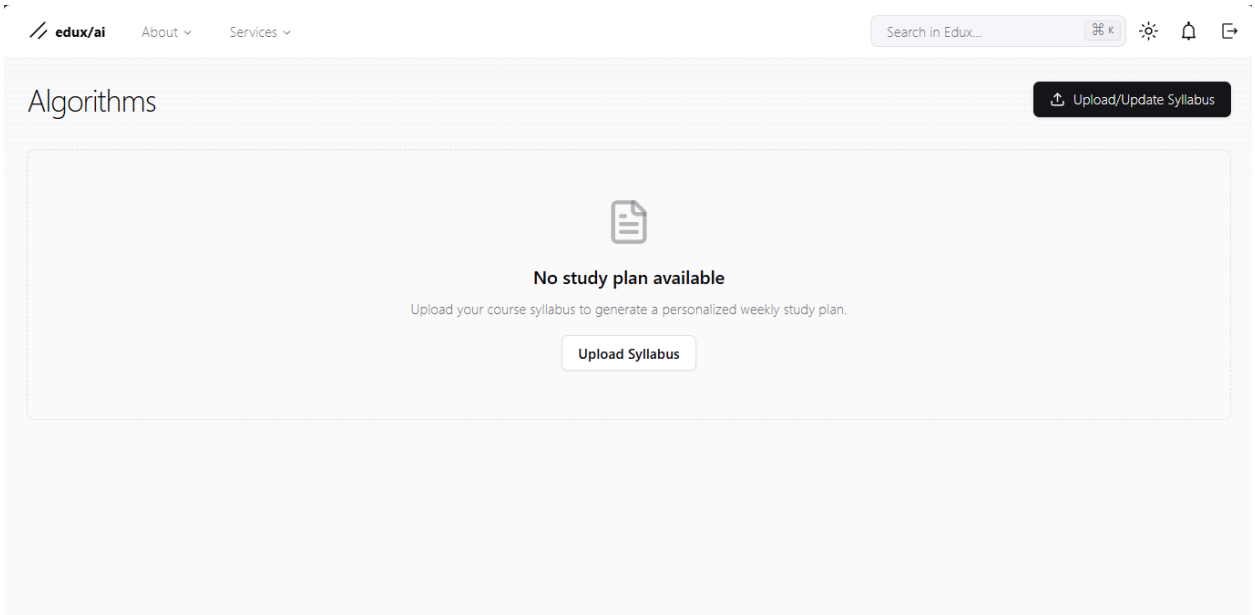
7 - Edux Course Syllabus

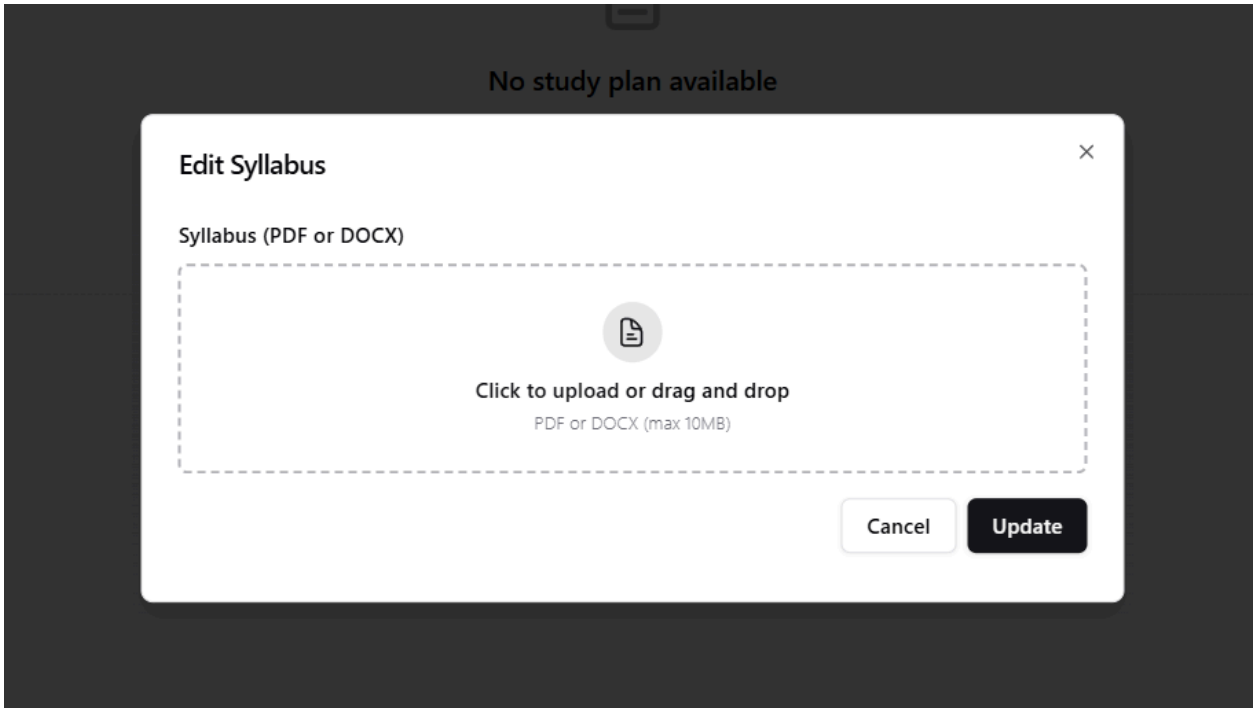


A syllabus for the course can be uploaded when creating a couse. There is a section for that in the bottom lef tof the course creation modal. Weekly study plan will be created immediately after the upload of syllabus.



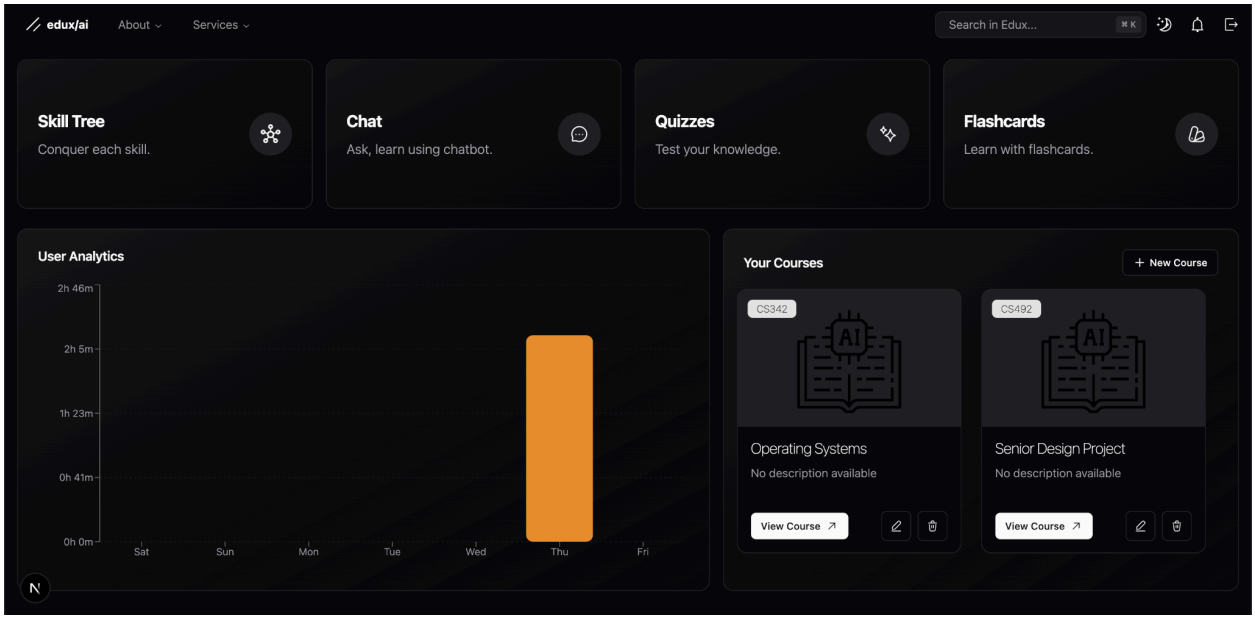
Weekly study plan can be accessed through the course homepage. This section provides a study plan for uploaded course syllabus.

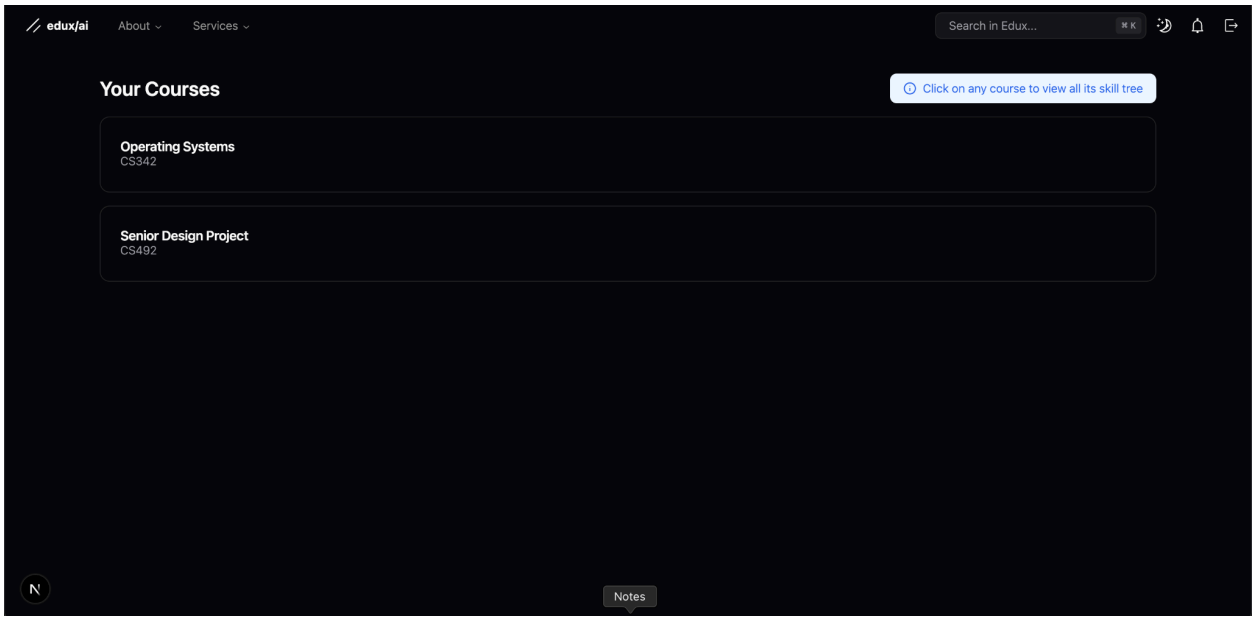


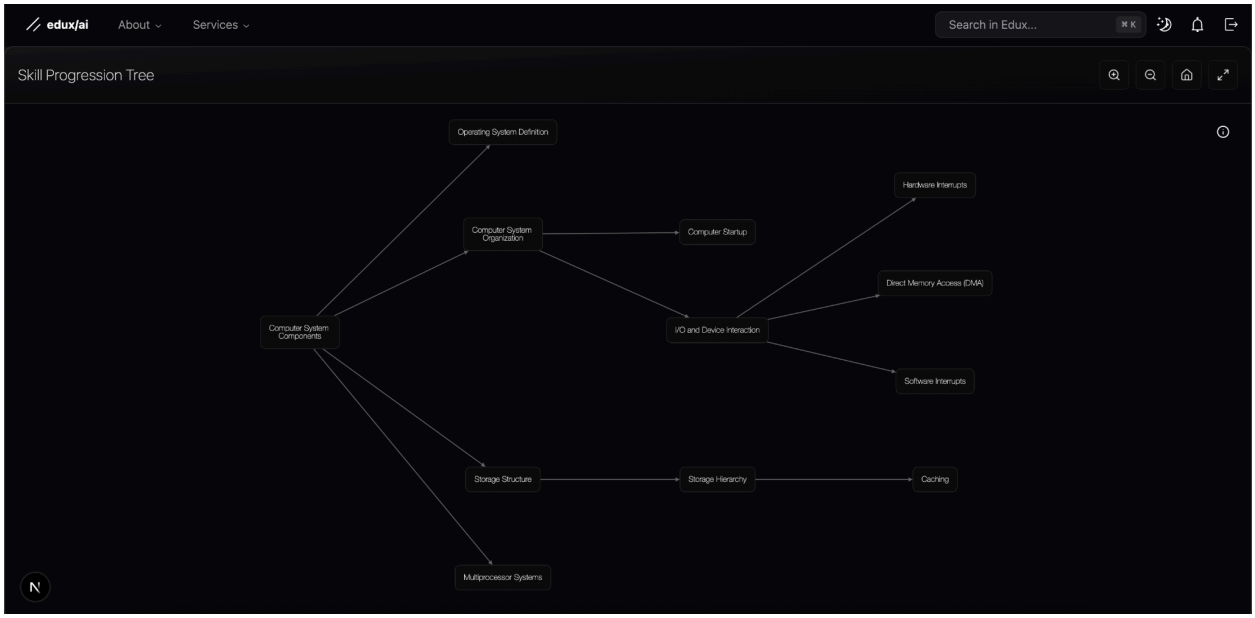


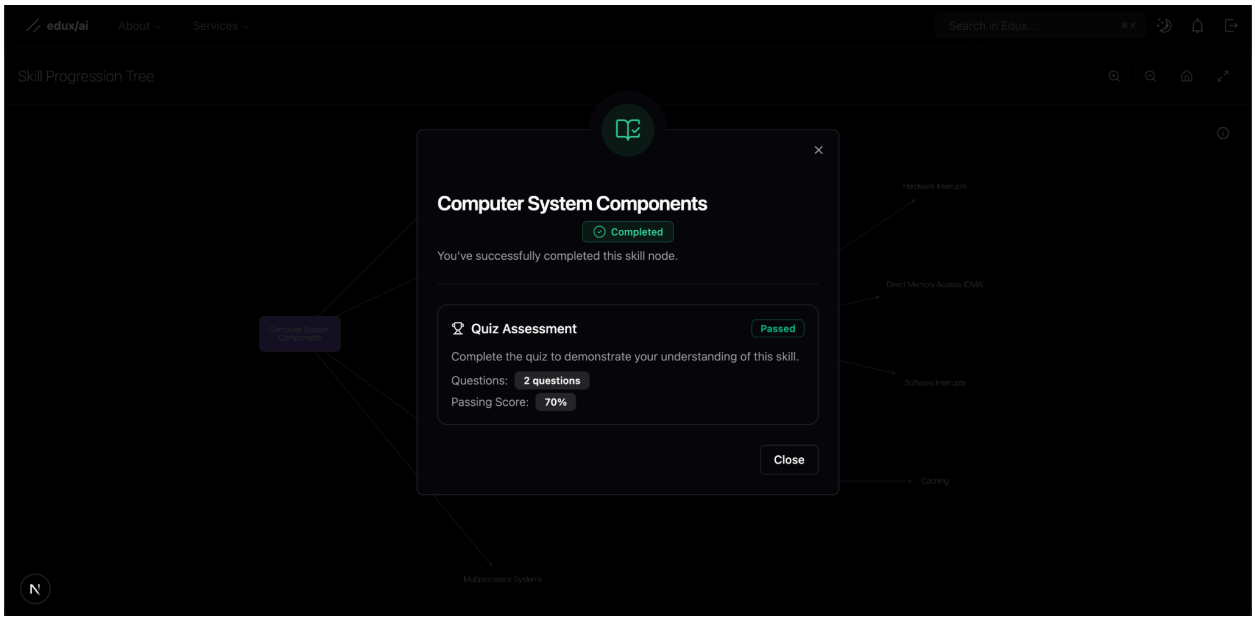
If there is no existing syllabus uploaded or not uploaded in the creation of the course, user can add the syllabus through Upload/Update Syllabus. Weekly study plan will be created immediately after the upload of syllabus.

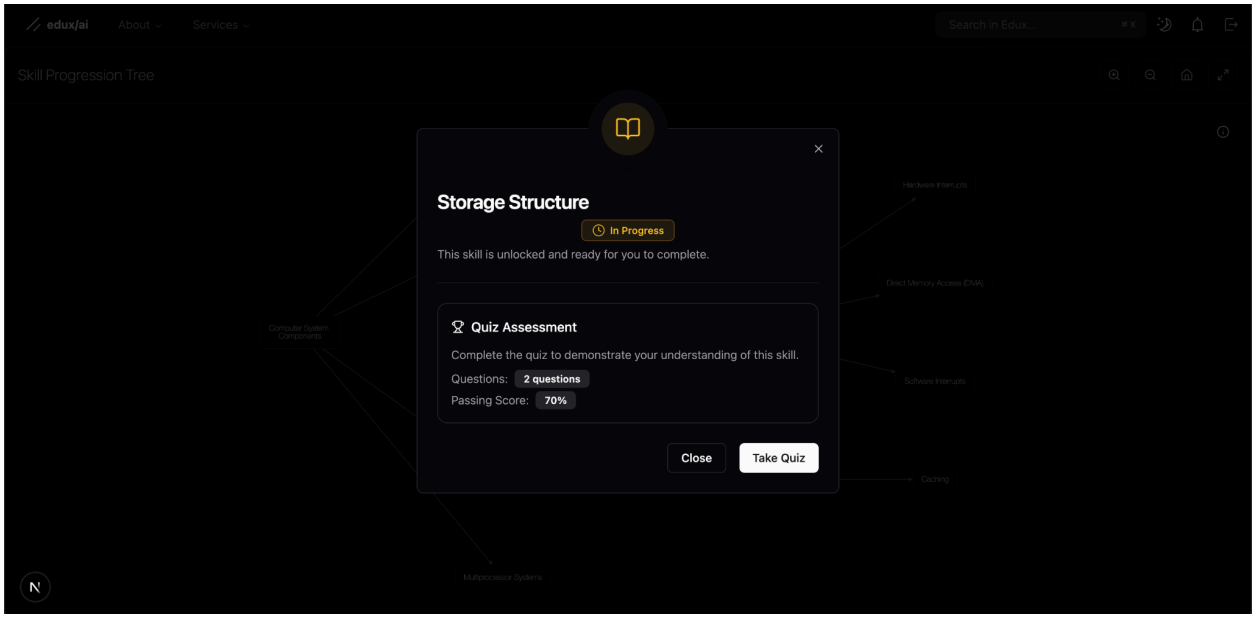
8 - Edux Skill Tree

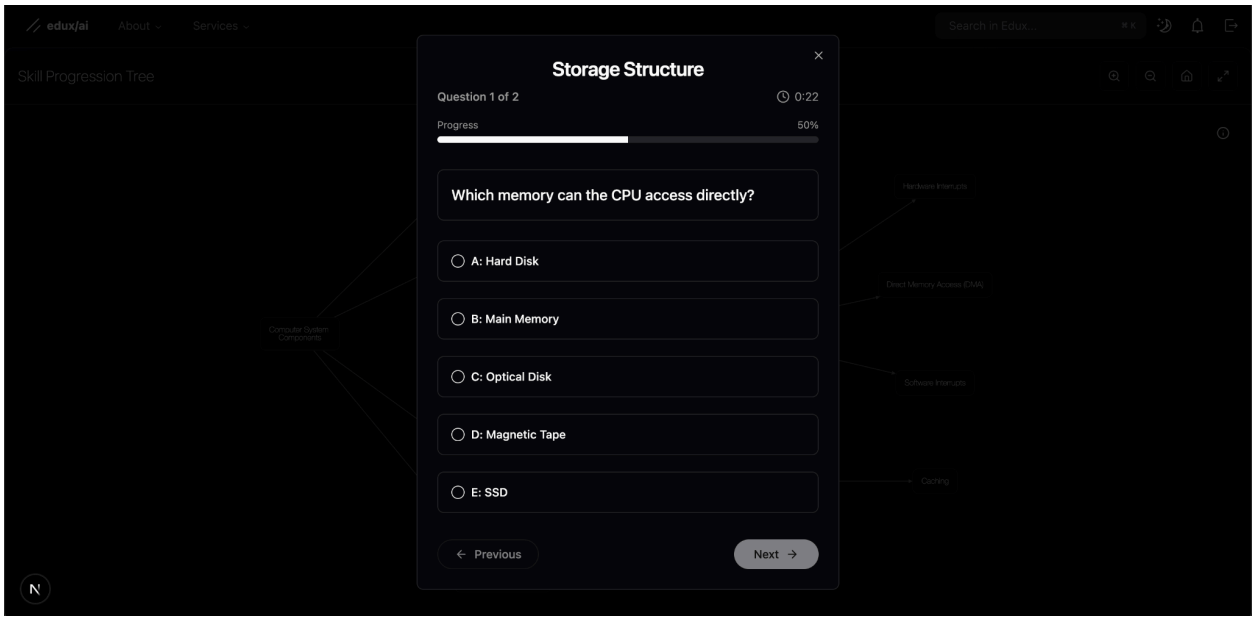


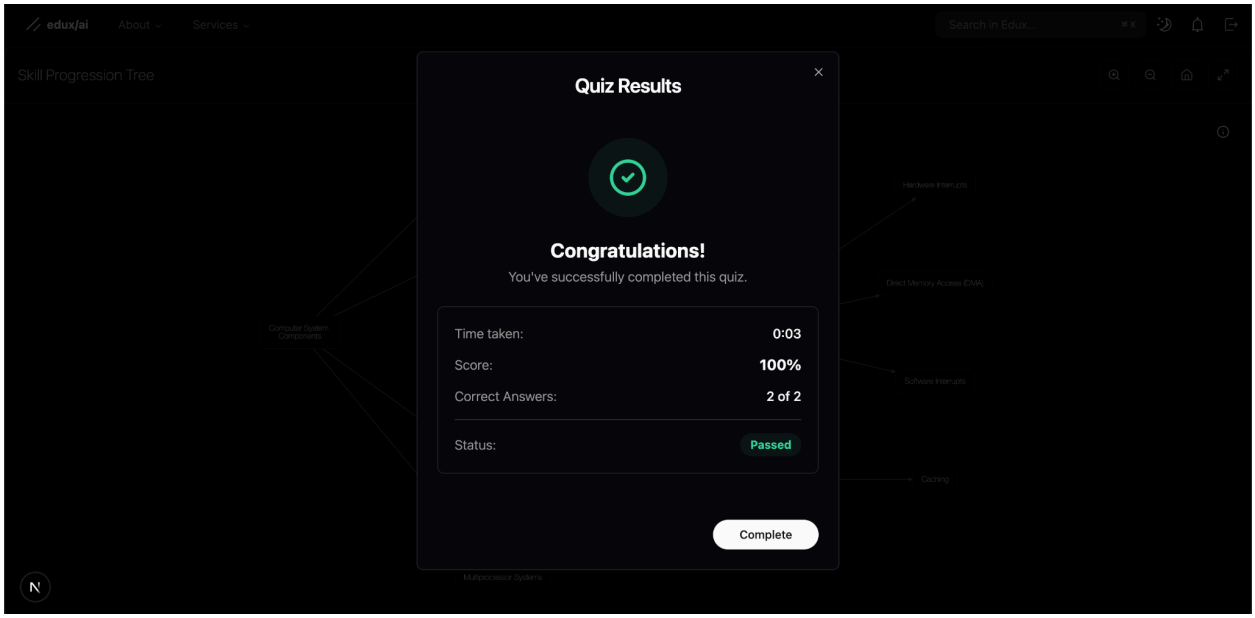


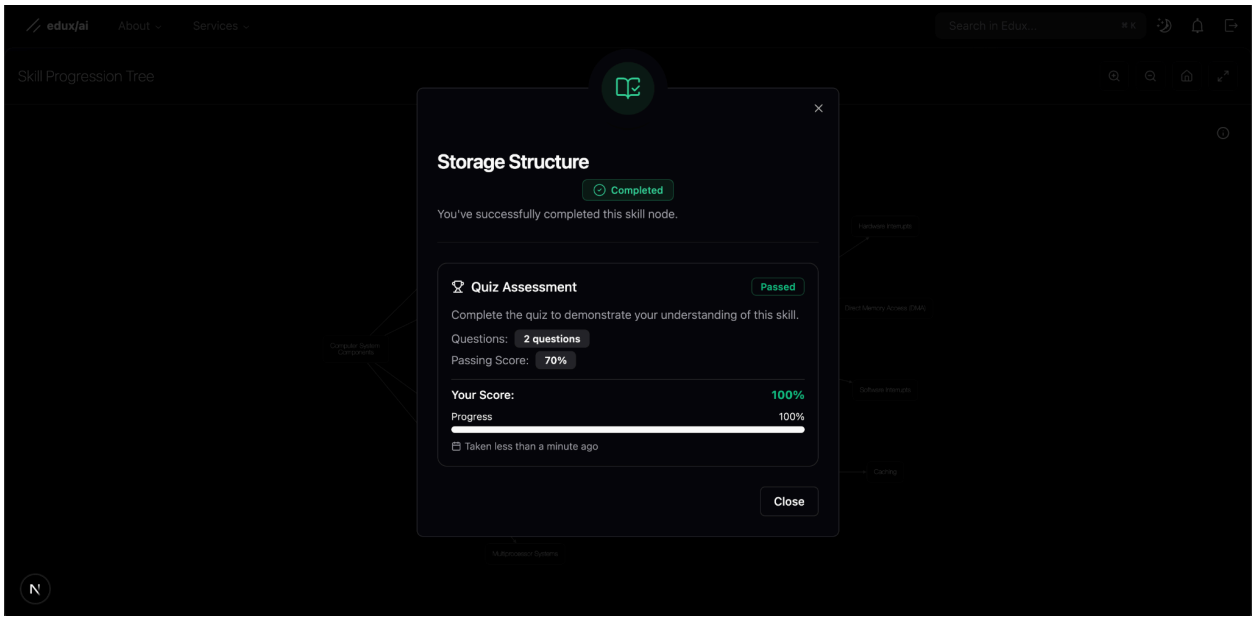

edux/ai
About
Services
Search in Edux...
⌘ K
Skill Progression Tree
Operating System Definition
Computer System Organization
Computer Startup
Hardware Interrupts
Direct Memory Access (DMA)
Computer System Components
I/O and Device Interaction
Software Interrupts
Storage Structure
Storage Hierarchy
Caching
Multiprocessor Systems

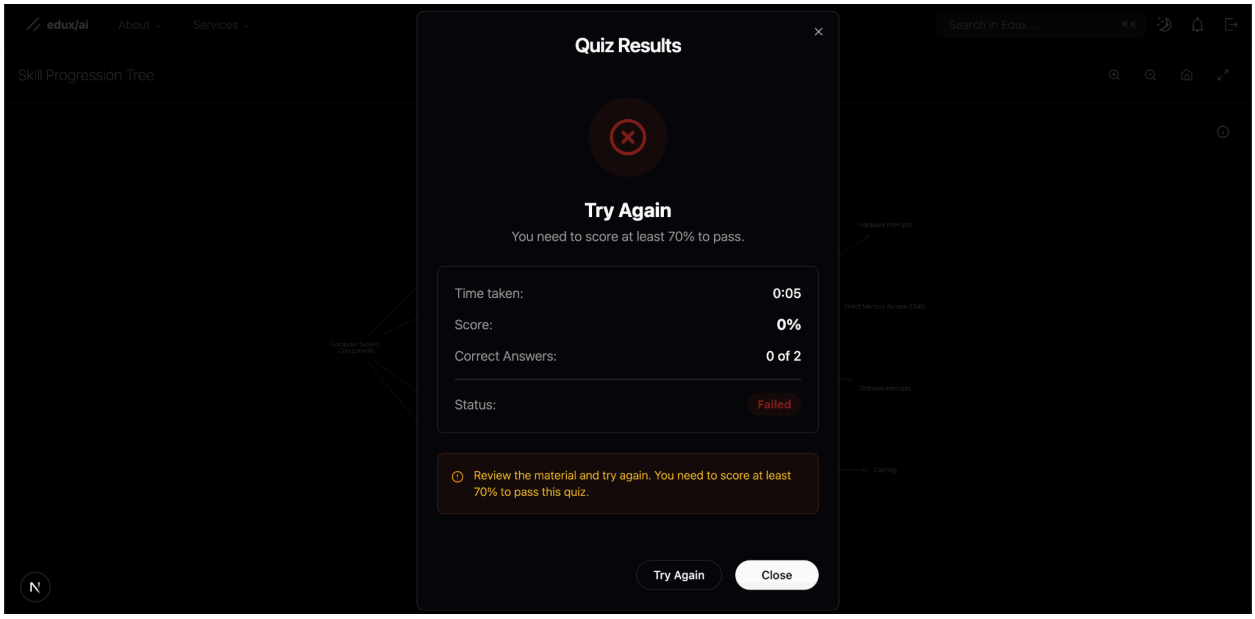

edux/ai
About
Services
Search in Edux...
⌘ K
Skill Progression Tree
Computer System Components
Completed
You've successfully completed this skill node.
Quiz Assessment
Passed
Complete the quiz to demonstrate your understanding of this skill.
Questions: 2 questions
Passing Score: 70%
Close


edux/ai
About
Services
Search in Edux...
Skill Progression Tree
Storage Structure
In Progress
This skill is unlocked and ready for you to complete.
Quiz Assessment
Complete the quiz to demonstrate your understanding of this skill.
Questions: 2 questions
Passing Score: 70%
Close
Take Quiz
Computer System Components
Hardware Interrupts
Direct Memory Access (DMA)
Software Interrupts
Caching
Multiprocessor Systems


edux/ai
About
Services
Search in Edux...
Skill Progression Tree
Storage Structure
Question 1 of 2
0:22
Progress
50%
Which memory can the CPU access directly?
A: Hard Disk
B: Main Memory
C: Optical Disk
D: Magnetic Tape
E: SSD
Previous
Next
Computer System Components
Hardware Interrupts
Direct Memory Access (DMA)
Software Interrupts
Caching


edux/ai
About
Services
Search in Edux...
Skill Progression Tree
Quiz Results
Congratulations!
You've successfully completed this quiz.
Time taken: 0:03
Score: 100%
Correct Answers: 2 of 2
Status: Passed
Complete
Hardware Interrupts
Direct Memory Access (DMA)
Software Interrupts
Caching
Computer System Components
Multiprocessor Systems


edux/ai
About
Services
Search in Edux...
Skill Progression Tree
Storage Structure
Completed
You've successfully completed this skill node.
Quiz Assessment
Passed
Complete the quiz to demonstrate your understanding of this skill.
Questions: 2 questions
Passing Score: 70%
Your Score: 100%
Progress 100%
Taken less than a minute ago
Close
Hardware Interrupts
Direct Memory Access (DMA)
Software Interrupts
Caching
Computer System Components
Multiprocessor Systems



Edux Skill Tree

The **Edux Skill Tree** is a dynamic and visual representation of your learning progress and mastery over course topics. It evolves as you interact with the platform, especially through the AI-powered chat and course features.

How It Works

1. **Starts from Chat Interactions**:
When you begin chatting with the AI about course topics—asking questions, exploring concepts, or uploading slides—the system captures what you're learning. These interactions are analyzed to determine which skills or subtopics you've engaged with.

2. **Skill Tree is Updated Automatically**:
Every time you enter the **Skill Tree screen**, it reflects the latest updates. This includes new skills you've unlocked or expanded based on your recent chat conversations. So if you've recently discussed a new unit or concept in the chat, your skill tree will show progress in that area the next time you open it.

3. **Course List Connection**:
The skill tree is also tied to your **Course List screen**. When you navigate to a specific course and then view the skill tree, it shows your progression within that course's structure—what you've learned, what's in progress, and what's left to explore.

4. **Visual and Interactive**:
The tree is designed to be intuitive and engaging. Skills are displayed as nodes or branches, with connections showing the prerequisites or dependencies between topics. You can click on each skill to review what you've learned, revisit related chats, or dive deeper into the content.

5. **Continuous Feedback Loop**:
Every time you use the chat, learn something new, or even edit an existing conversation, the system re-evaluates and updates your skill progression. This ensures your skill tree always mirrors your actual learning journey.

Course List & Skill Tree Entry

Below the analytics, you see **Your Courses**—for example, Operating Systems (CS342) and Senior Design Project (CS492). Clicking **"View Course"** takes you into that course's dedicated Skill Tree view. Here, the tree is pre-filtered to only that course's topics. This linkage ensures that whatever you've discussed in the chat about Operating Systems appears in its specific tree when you enter it.

Dynamic Skill Tree Visualization

In the **Skill Progression Tree**, each node represents a discrete topic or sub-skill (e.g., "Storage Structure," "Hardware Interrupts"). Lines between nodes show the prerequisite relationships: you must master foundational nodes before unlocking the next ones. Every time you visit this screen, the tree redraws itself to reflect any new chat-based learning—highlighting nodes you've unlocked or made progress on since your last visit.

Completed Skill Pop-Up

Clicking on a fully mastered node (like **"Computer System Components"**) opens a **Completed Skill** dialog. It shows a green "Completed" badge and the quiz outcome used to verify mastery (for instance, 2 questions passed at a 70% threshold). This popup confirms that your chat discussions and quiz performance have been successfully merged into your Skill Tree progress.

Quiz Workflow for In-Progress Skills

For nodes you've unlocked but not yet mastered—such as **"Storage Structure"**—clicking the node opens an **In Progress** dialog. It lists the quiz details (number of questions and passing score) and offers a **Take Quiz** button. Once you start, a **Quiz Question Screen** displays one question at a time (e.g., "Which memory can the CPU access directly?"), shows a progress bar, and lets you

navigate between questions. Completing the quiz updates the node's status and feeds back into the Skill Tree visualization on your next visit.

Completed Skill Pop-Up

Clicking a fully mastered node opens a "Quiz Results" dialog with a green checkmark and "Congratulations!" message. It shows time taken (e.g. 0:03), score (100%), correct answers (2/2), and a "Passed" status. This confirms the quiz has been recorded and the node marked complete.

Storage Structure Confirmation

The "Storage Structure" node's modal displays a green "Completed" badge and a "Quiz Assessment" summary: 2 questions, 70% passing score, your score (100%), and a filled progress bar. A timestamp notes when you passed. This links your quiz success directly to that course topic.

Failed Quiz Feedback

If you score below the pass threshold, the "Quiz Results" dialog shows a red "Try Again" header, your score (0%), correct answers (0/2), and a "Failed" status. A warning box advises reviewing the material and retaking the quiz, with "Try Again" and "Close" options.

Installation Guide

For installation guide check the Edux project implementation in GitHub. The information can be found in the website.

10. References

Below is a list of 20 references in IEEE format that you can include in your report's bibliography:

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 6000–6010.

[2] T. Brown *et al.*, "Language Models Are Few-Shot Learners," in *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.

[3] S. Mitra and M. B. Choudhury, "The Evolution of Personalized Learning Systems: A Survey of Techniques and Technologies," *IEEE Trans. Learn. Technol.*, vol. 14, no. 1, pp. 24–36, Jan.–Mar. 2021.

[4] R. Rus and D. Graesser, "Research on Question Generation in Educational Contexts," *Int. J. Artif. Intell. Educ.*, vol. 28, no. 2, pp. 435–463, Jun. 2018.

[5] R. S. Baker and K. Yacef, “The State of Educational Data Mining in 2019: A Review and Future Visions,” *J. Educ. Data Min.*, vol. 11, no. 2, pp. 1–17, 2019.

[6] Docker Inc., “Docker Documentation,” 2024. [Online]. Available: https://docs.docker.com/ [Accessed: Apr. 17, 2025].

[7] Microsoft Azure, “Azure Blob Storage Documentation,” Microsoft, 2024. [Online]. Available: https://docs.microsoft.com/azure/storage/blobs/ [Accessed: Apr. 9, 2025].

[8] European Parliament and Council, “Regulation (EU) 2016/679 (General Data Protection Regulation),” *Off. J. Eur. Union*, Apr. 27, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj [Accessed: Apr. 28, 2025].

[9] Microsoft Corporation, “TypeScript: JavaScript With Syntax For Types,” TypeScript, 2025. [Online]. Available: https://www.typescriptlang.org/ [Accessed: Apr. 27, 2025].

[10] Tailwind Labs Inc., “Tailwind CSS: Rapidly Build Modern Websites Without Ever Leaving Your HTML,” Tailwind CSS, 2025. [Online]. Available: https://tailwindcss.com/ [Accessed: Apr. 27, 2025].

[11] Vercel Inc., "Next.js: The React Framework for the Web," Next.js, 2025. [Online]. Available: https://nextjs.org/ [Accessed: Apr. 27, 2025].

[12] Radix UI, “Radix UI Documentation,” 2024. [Online]. Available: https://www.radix-ui.com/ [Accessed: Apr. 28, 2025].

[13] React Flow, “React Flow: A Library for Building Node-Based Applications,” 2024. [Online]. Available: https://reactflow.dev/ [Accessed: Mar. 28, 2025].

[14] "react-card-flip," npm, [Online]. Available: https://www.npmjs.com/package/react-card-flip. [Accessed: 28-Apr-2025].

[15] Recharts, “Recharts: Redefined Chart Library Built with React and D3,” 2024. [Online]. Available: https://recharts.org/ [Accessed: Apr. 14, 2025].

[16] M. Fowler and J. Lewis, “Microservices: a Definition of this New Architectural Term,” ThoughtWorks, Mar. 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html [Accessed: Apr. 14, 2025].

[17] Tiangolo, “FastAPI: high performance, easy to learn, fast to code,” 2023. [Online]. Available: https://fastapi.tiangolo.com/ [Accessed: Apr. 6, 2025].

[18] D. Hardt, “The OAuth 2.0 Authorization Framework,” RFC 6749, Oct. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6749 [Accessed: Apr. 27, 2025].

[19] M. Bayer, “SQLAlchemy,” SQLAlchemy, 2025. [Online]. Available: https://www.sqlalchemy.org/ [Accessed: Apr. 27, 2025].

[20] S. Pydantic Core Team, “Pydantic,” Pydantic, 2025. [Online]. Available: https://docs.pydantic.dev/ [Accessed: Apr. 27, 2025].

[21] Atlassian, “Jira Software Documentation,” Atlassian Support, 2025. [Online]. Available: https://confluence.atlassian.com/jira [Accessed: Apr. 28, 2025].

[22] GitHub, “GitHub Docs,” GitHub, 2025. [Online]. Available: https://docs.github.com/ [Accessed: Apr. 28, 2025].

[23] Slack Technologies, “Slack,” Slack, 2025. [Online]. Available: https://slack.com/ [Accessed: Apr. 27, 2025].

[24] Microsoft Corporation, “Microsoft Teams,” Microsoft, 2025. [Online]. Available: https://www.microsoft.com/microsoft-teams/ [Accessed: Apr. 27, 2025].

[25] Republic of Turkey, "Kişisel Verilerin Korunması Kanunu [Personal Data Protection Law]," Resmî Gazete [Official Gazette], no. 29677, Apr. 7, 2016. [Online]. Available: https://www.resmigazete.gov.tr/eskiler/2016/04/20160407-1.htm [Accessed: Apr. 27, 2025].

[26] OpenAI, "OpenAI API Documentation", OpenAI, 2025. [Online]. Available: https://platform.openai.com/docs [Accessed: Apr. 27, 2025].

[27] Hugging Face, "Transformers: State-of-the-art Machine Learning for Pytorch, TensorFlow, and JAX", Hugging Face, 2025. [Online]. Available: https://huggingface.co/docs/transformers/index [Accessed: Apr. 27, 2025].

[28] Cloud Native Computing Foundation, “Kubernetes Documentation,” 2024. [Online]. Available: https://kubernetes.io/docs/home/ [Accessed: Apr. 12, 2025].

[29] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," arXiv:2005.11401, May 2020. [Online]. Available: https://arxiv.org/abs/2005.11401 [Accessed: Apr. 27, 2025].