Software Engineering Project

Sep 2025 - Team 38

Milestone 6

Team Members

Name	Roll Number
Omkar Shankar Pawar	23F1001898
Palash Johri	21F1000512
Prabhu	21F2001015
Raghavendra Narayan Jha	21F1003534
Rishav Kumar	22F3001352
Sohini Sarkar	21F2001369
Treasa Janet	21F1002081

Table of Contents

Milestone - 1 & 2

Problem | User Requirements | User Stories

1. Problem Statement

Design and implement a web application for teaching assistants to help them with tackle the problems they face, like:

- Addressing Repetitive Queries
- No proper segmentation of teaching content
- Burden of administering tasks
- Time consuming teaching content generation

1.1 User Research & Needs Identification

- Conducted interviews/surveys with target user groups (Teaching Assistants).
- Identified gaps and challenges faced by the users with the current system.
- Selected priority areas based on impact, feasibility, and interest.

2. Various Users

2.1 User Types

Primary Users:

- IIT Madras BS Teaching Assistants: Teaching Assistants are the primary users of the application. They are responsible for managing day-to-day course operations such as grading assignments, monitoring student progress, answering queries, and assisting instructors in content delivery. They rely on the system for streamlined communication, efficient task management, and easy access to course data.

Secondary Users:

- IIT Madras BS Program Support Team: Program Administrators oversee the broader academic program and manage multiple courses, instructors, and TAs. They use the application to track course performance, allocate teaching resources, and ensure compliance with institutional policies. Their focus is on maintaining smooth program operations and ensuring data consistency across courses.

Tertiary Users:

- IIT Madras BS Course Instructors: Course Instructors are the subject

matter experts who design and deliver course content. While they are not the main users of the system, they occasionally interact with it to upload materials, review student performance summaries, and coordinate with TAs and administrators. Their goal is to ensure teaching quality and learning effectiveness.

2.2 User Stories

1. For Unprepared Students in Live Sessions

As a Teaching Assistant,

I want students to complete the recorded lectures before joining my live session,

So that I can ensure they have the basic knowledge required and I can focus the session on solving problems instead of re-teaching the entire theory.

2. For Exam-Focused Learning

As a Teaching Assistant,

I want some feature like where I can tag course materials like topics, questions, or video timestamps as "High-Priority".

So that I can quickly filter and deliver the specific content that students want during revision sessions close to their quizzes.

3. For Unawareness of Official Information

As a Teaching Assistant,

I want new students to complete a tutorial on their course portal that guides them to use essential resources like the student handbook and grading documents,

So that they know where to find official and accurate information from the start, which will reduce their reliance on potentially incorrect information from their peers.

4. For Underutilization of Help Resources

As a Teaching Assistant,

I want a something great on the course dashboard that is trained to answer common administrative and policy questions by directly quoting from the official course documents,

So that students get authenticated answers 24/7 and learn to trust and use the official resources provided to them easily.

5. For Repetitive Questions

As a Teaching Assistant,

I want a feature where a discussion forum automatically suggests similar questions that have already been answered as a student type of their new question,

So that students can find their answers immediately and avoid creating

duplicate posts.

6. For Unread Course Documents

As a Teaching Assistant,

I want a feature that helps people to easily find and share a direct link to a specific paragraph or section within course documents like the handbook or grading document,

So that when a student asks a question, I can guide them to the exact source of information easily and precisely.

7. For Handling Repetitive Questions

As a Teaching Assistant,

I want a dedicated and searchable FAQ section for my course,

So that students can find answers to common questions about deadlines and requirements themselves, reducing my need to answer the same queries repeatedly.

8. For Managing Workload

As a Teaching Assistant,

I want a tool to help automate repetitive administrative tasks like tracking submissions,

So that I can reduce time spent on manual work and focus more on preparing for and conducting my sessions.

9. For Handling Deadline Extension Requests

As a Teaching Assistant,

I want a formal system where students can submit requests for deadline extensions with a valid reason,

So that I can track all requests in one place and make fair, consistent decisions.

2.3 Core Features

1. RAG Chatbot

- For answering repetitive queries

2. Content Tagger

- For segmentation of teaching content

3. Workflow Agent

- Automating the task of Teaching Assistants

4. Assessment Generator

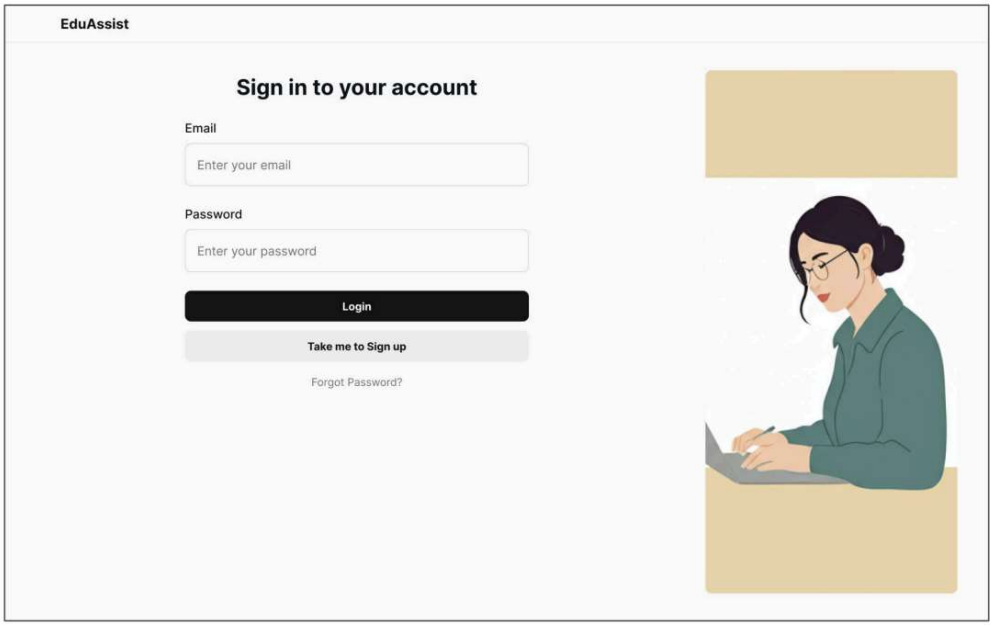
- For constant creation of assessments as per the coursework.

5.Slide Deck Generator

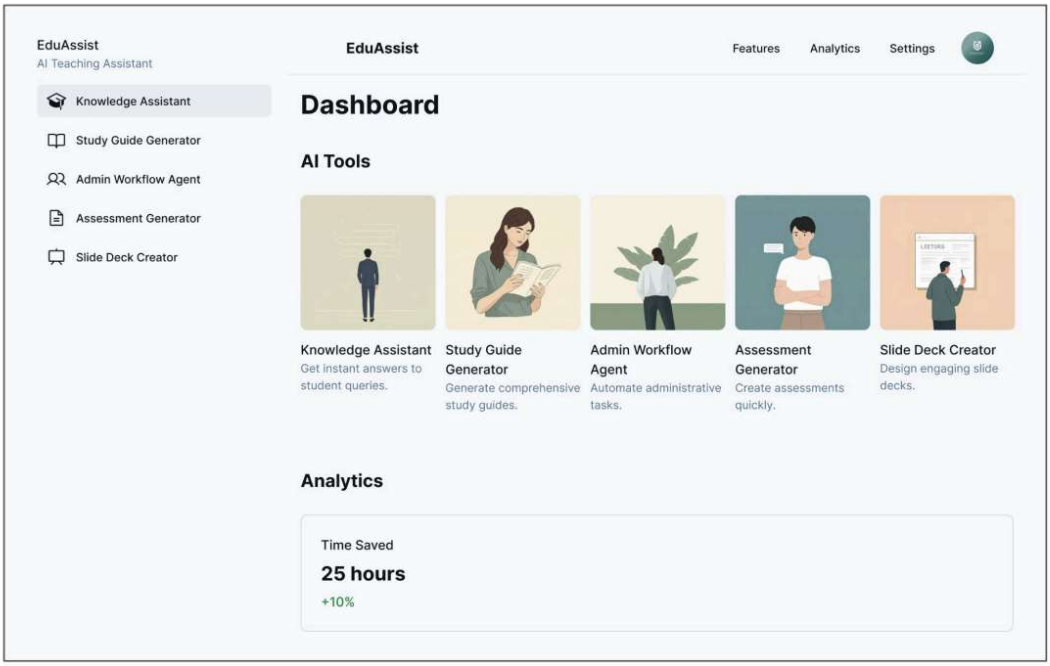
- For easy creation of presentation for teaching

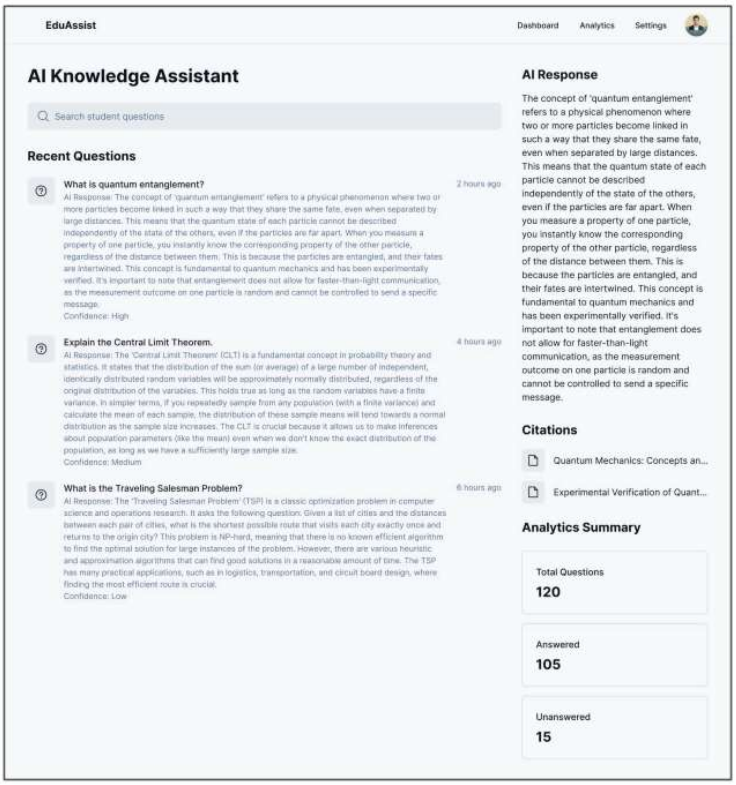
2.3 Sample Wireframes and Storyboards

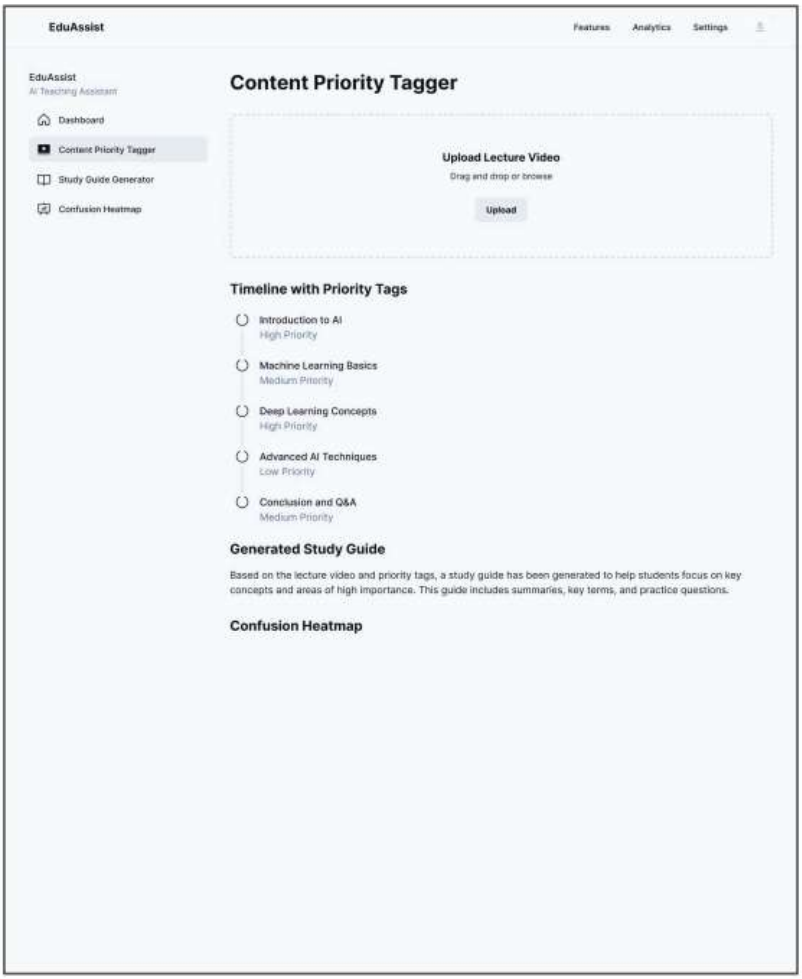
Login



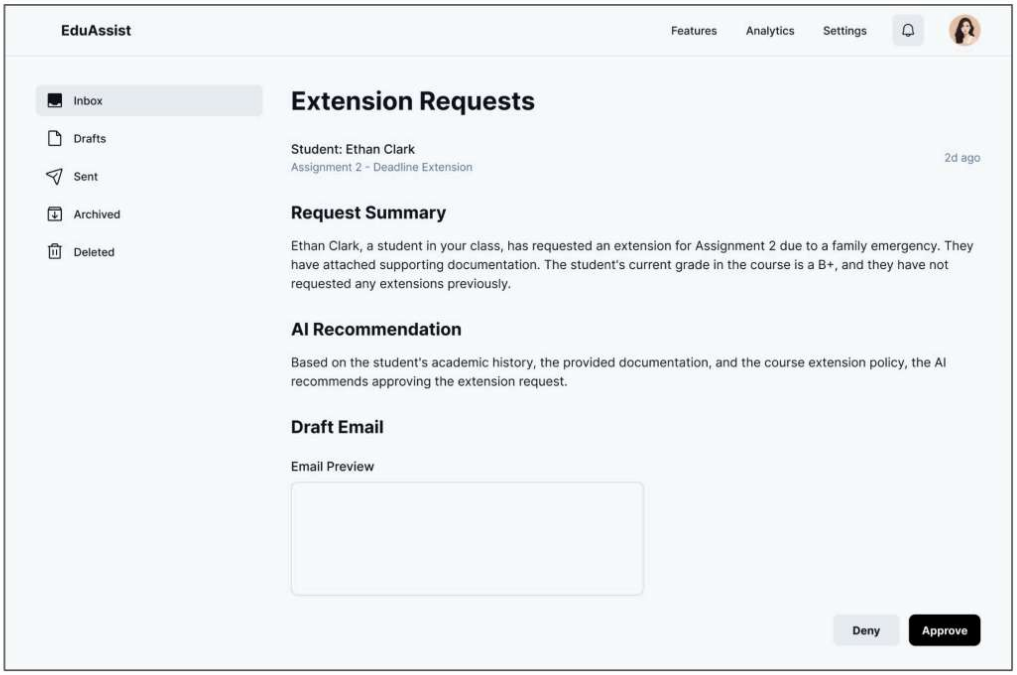
Landing Dashboard



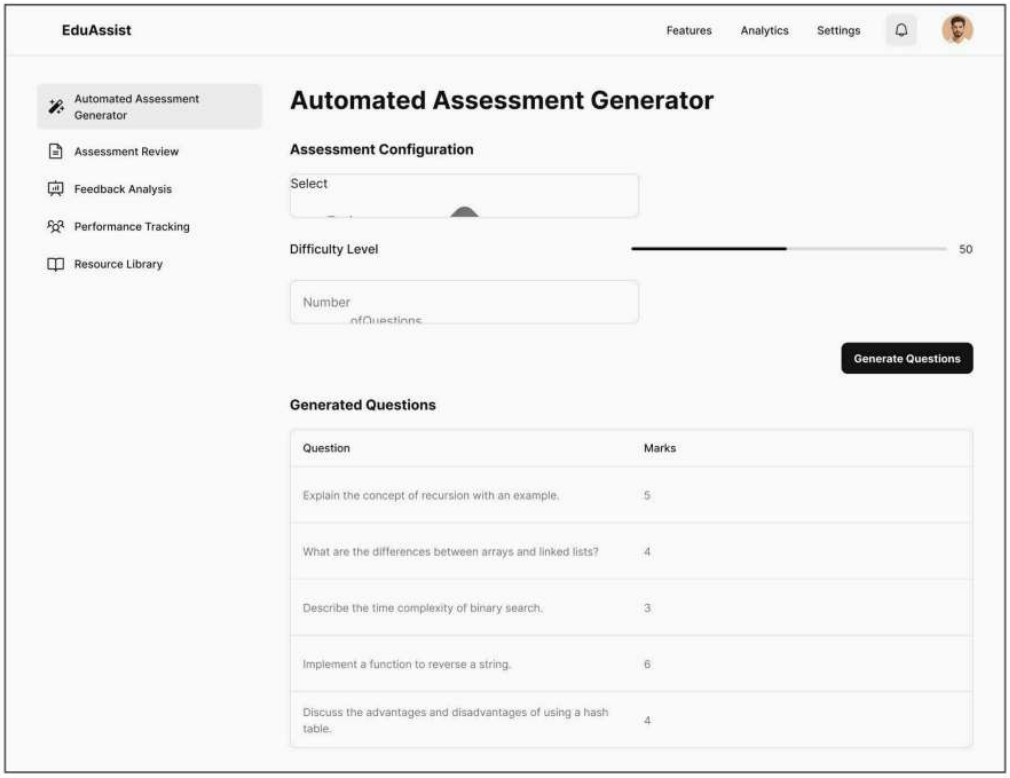

EduAssist
Dashboard
Analytics
Settings
AI Knowledge Assistant
Search student questions
Recent Questions
What is quantum entanglement?
2 hours ago
AI Response: The concept of 'quantum entanglement' refers to a physical phenomenon where two or more particles become linked in such a way that they share the same fate, even when separated by large distances. This means that the quantum state of each particle cannot be described independently of the state of the others, even if the particles are far apart. When you measure a property of one particle, you instantly know the corresponding property of the other particle, regardless of the distance between them. This is because the particles are entangled, and their fates are intertwined. This concept is fundamental to quantum mechanics and has been experimentally verified. It's important to note that entanglement does not allow for faster-than-light communication, as the measurement outcome on one particle is random and cannot be controlled to send a specific message.
Confidence: High
Explain the Central Limit Theorem.
4 hours ago
AI Response: The 'Central Limit Theorem' (CLT) is a fundamental concept in probability theory and statistics. It states that the distribution of the sum (or average) of a large number of independent, identically distributed random variables will be approximately normally distributed, regardless of the original distribution of the variables. This holds true as long as the random variables have a finite variance. In simpler terms, if you repeatedly sample from any population (with a finite variance) and calculate the mean of each sample, the distribution of these sample means will tend towards a normal distribution as the sample size increases. The CLT is crucial because it allows us to make inferences about population parameters (like the mean) even when we don't know the exact distribution of the population, as long as we have a sufficiently large sample size.
Confidence: Medium
What is the Traveling Salesman Problem?
6 hours ago
AI Response: The 'Traveling Salesman Problem' (TSP) is a classic optimization problem in computer science and operations research. It asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? This problem is NP-hard, meaning that there is no known efficient algorithm to find the optimal solution for large instances of the problem. However, there are various heuristic and approximation algorithms that can find good solutions in a reasonable amount of time. The TSP has many practical applications, such as in logistics, transportation, and circuit board design, where finding the most efficient route is crucial.
Confidence: Low
AI Response
The concept of 'quantum entanglement' refers to a physical phenomenon where two or more particles become linked in such a way that they share the same fate, even when separated by large distances. This means that the quantum state of each particle cannot be described independently of the state of the others, even if the particles are far apart. When you measure a property of one particle, you instantly know the corresponding property of the other particle, regardless of the distance between them. This is because the particles are entangled, and their fates are intertwined. This concept is fundamental to quantum mechanics and has been experimentally verified. It's important to note that entanglement does not allow for faster-than-light communication, as the measurement outcome on one particle is random and cannot be controlled to send a specific message.
Citations
Quantum Mechanics: Concepts an...
Experimental Verification of Quant...
Analytics Summary
Total Questions
120
Answered
105
Unanswered
15


EduAssist
Features
Analytics
Settings
EduAssist
AI Teaching Assistant
Dashboard
Content Priority Tagger
Study Guide Generator
Confusion Heatmap
Content Priority Tagger
Upload Lecture Video
Drag and drop or browse
Upload
Timeline with Priority Tags
Introduction to AI
High Priority
Machine Learning Basics
Medium Priority
Deep Learning Concepts
High Priority
Advanced AI Techniques
Low Priority
Conclusion and Q&A
Medium Priority
Generated Study Guide
Based on the lecture video and priority tags, a study guide has been generated to help students focus on key concepts and areas of high importance. This guide includes summaries, key terms, and practice questions.
Confusion Heatmap

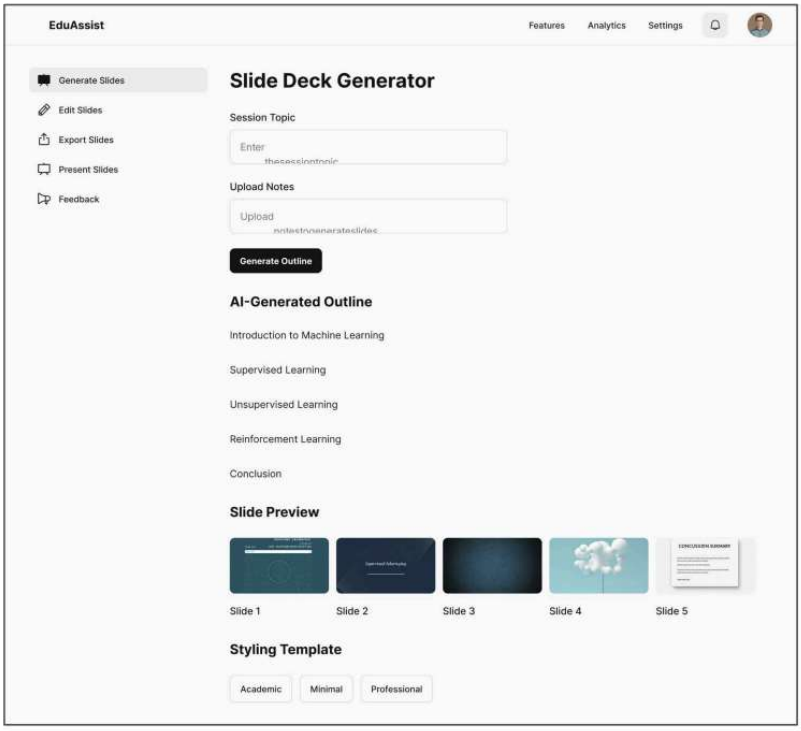
Admin Workflow Agent



Assessment Generator

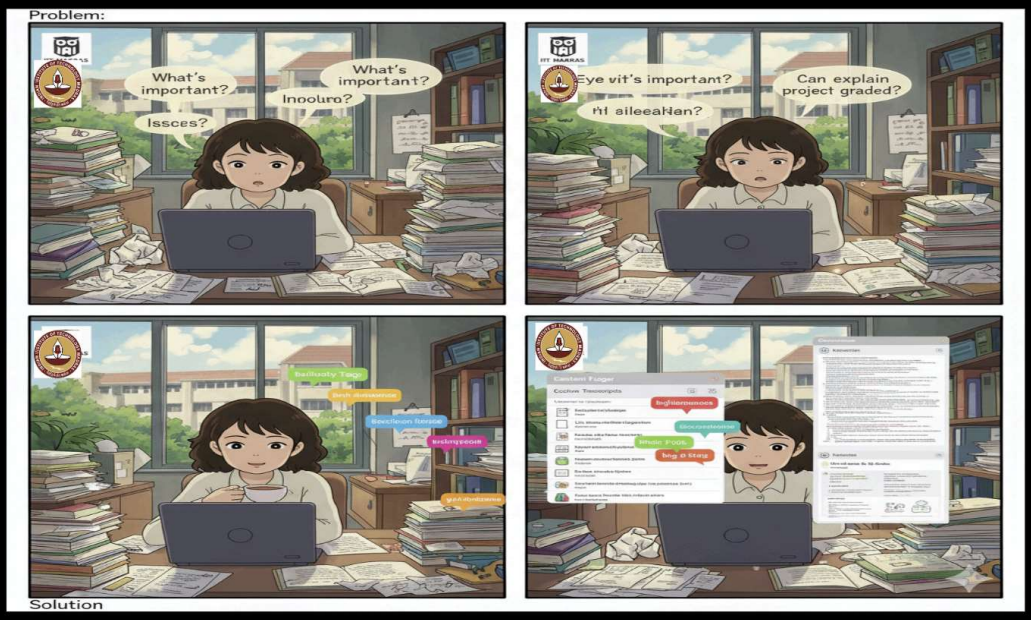


Slide Deck Generator

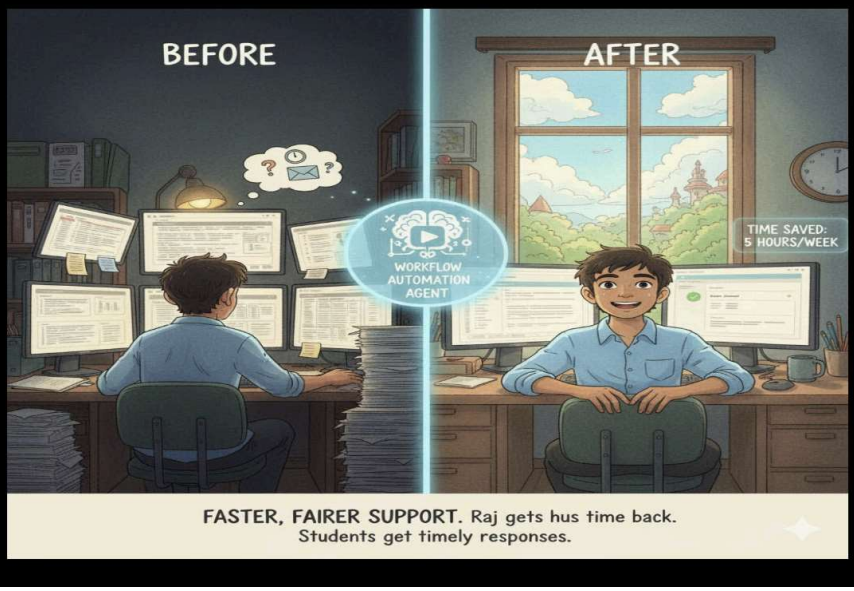


Storyboard 1: RAG Chatbot – Answering Student Questions at Scale

Storyboard 2: Content Tagger – Prioritizing What Matters Most



Storyboard 3: Workflow Agent – Automating Administrative Task



Storyboard 4 & 5: Assessment + Slide Deck Generation – Creating Content with Ease

Milestone - 3

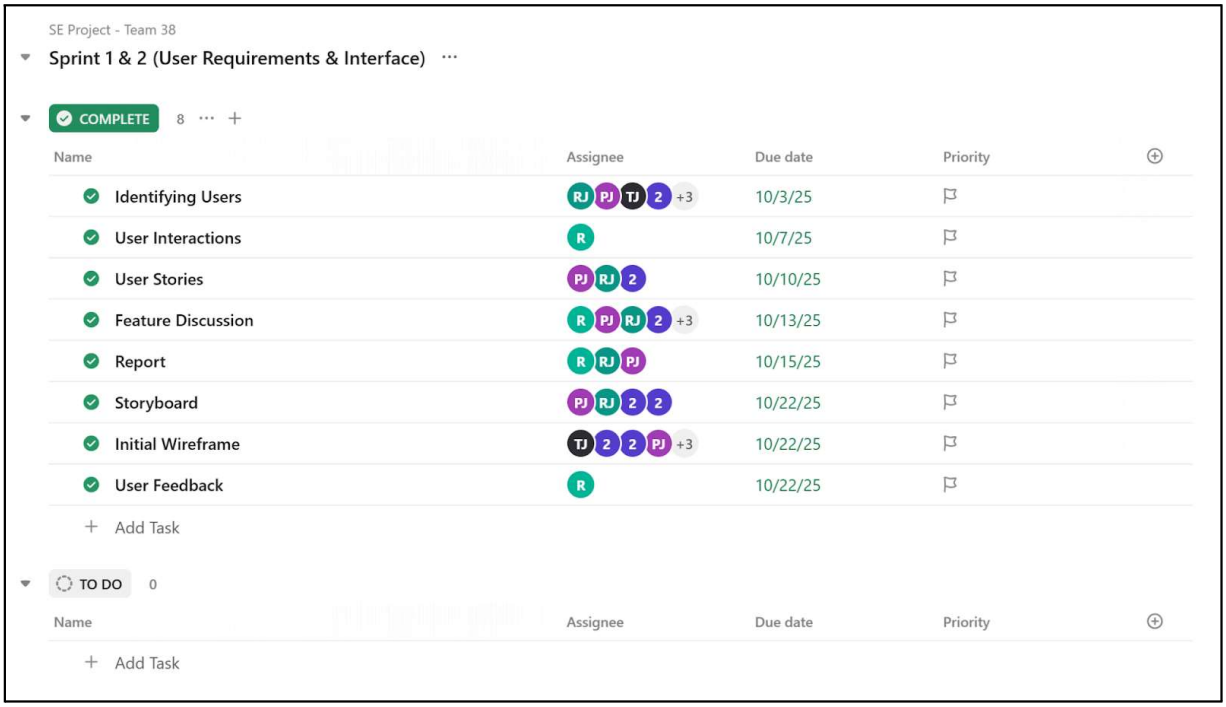
Schedule | Design

3. Project Schedule

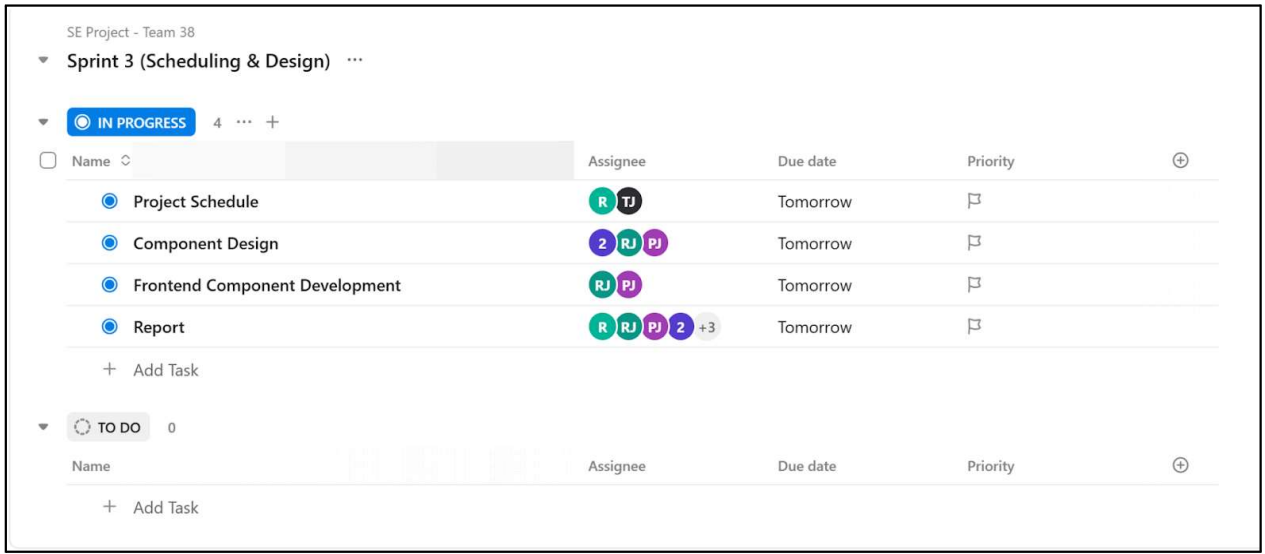
3.1 Task Distribution

We've strategically mapped out each project milestone, crafting sub-tasks in line with the SMART guidelines— ensuring clarity and achievability. To elevate our approach, we've evenly assigned these tasks to team members, capitalizing on their unique strengths.

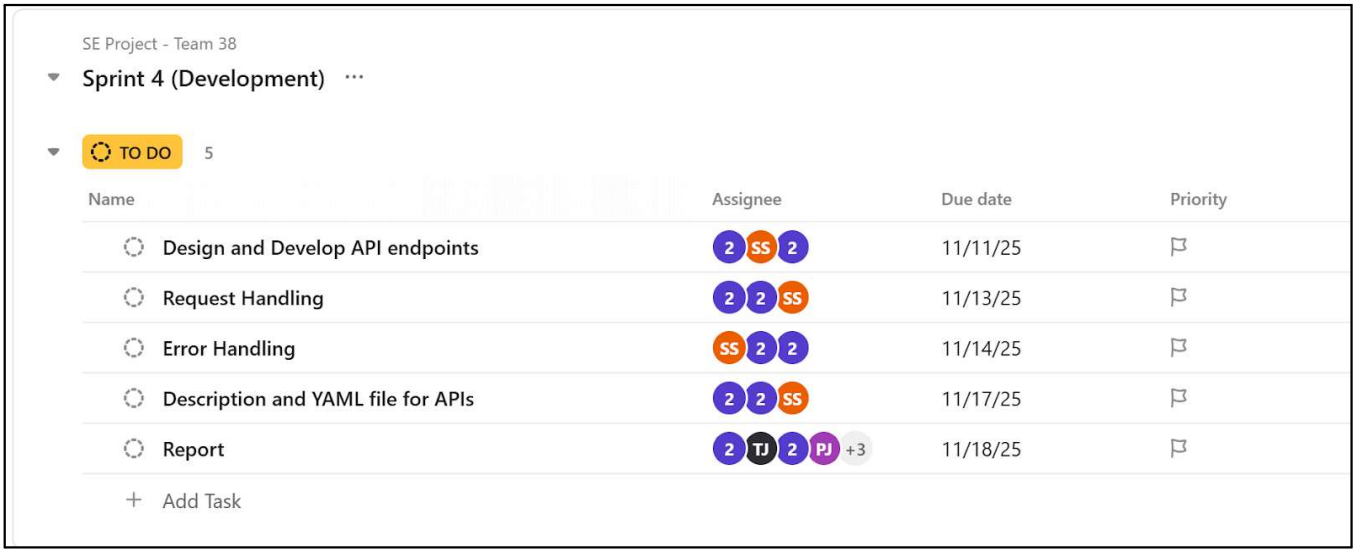
Sprint 1 & 2 (User Requirements & Interface):



Sprint 3 (Scheduling & Design):



Sprint 4 (Development):



SE Project - Team 38

Sprint 4 (Development) ···

TO DO 5

Name	Assignee	Due date	Priority
Design and Develop API endpoints	2 SS 2	11/11/25	
Request Handling	2 2 SS	11/13/25	
Error Handling	SS 2 2	11/14/25	
Description and YAML file for APIs	2 2 SS	11/17/25	
Report	2 TJ 2 PJ +3	11/18/25	

+ Add Task

Sprint 5 (Testing):

SE Project - Team 38

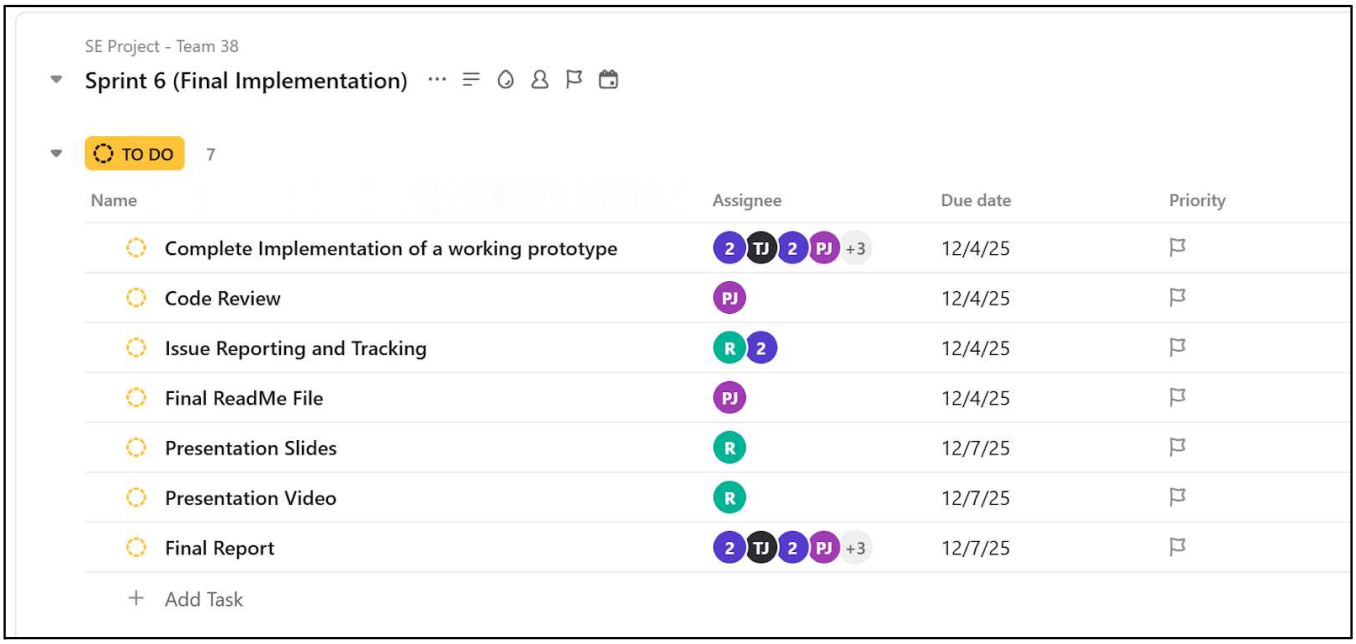
Sprint 5 (Testing) ···

TO DO 5 ··· +

Name	Assignee	Due date	Priority
Test Case Design	2 2	11/22/25	
Unit Testing	2 SS	11/28/25	
Integration Testing	R TJ PJ	11/28/25	
User Assessment Testing	R PJ TJ RJ +3	11/28/25	
Report	PJ R 2 RJ +3	11/29/25	

+ Add Task

Sprint 6 (Final Implementation):



SE Project - Team 38

Sprint 6 (Final Implementation)

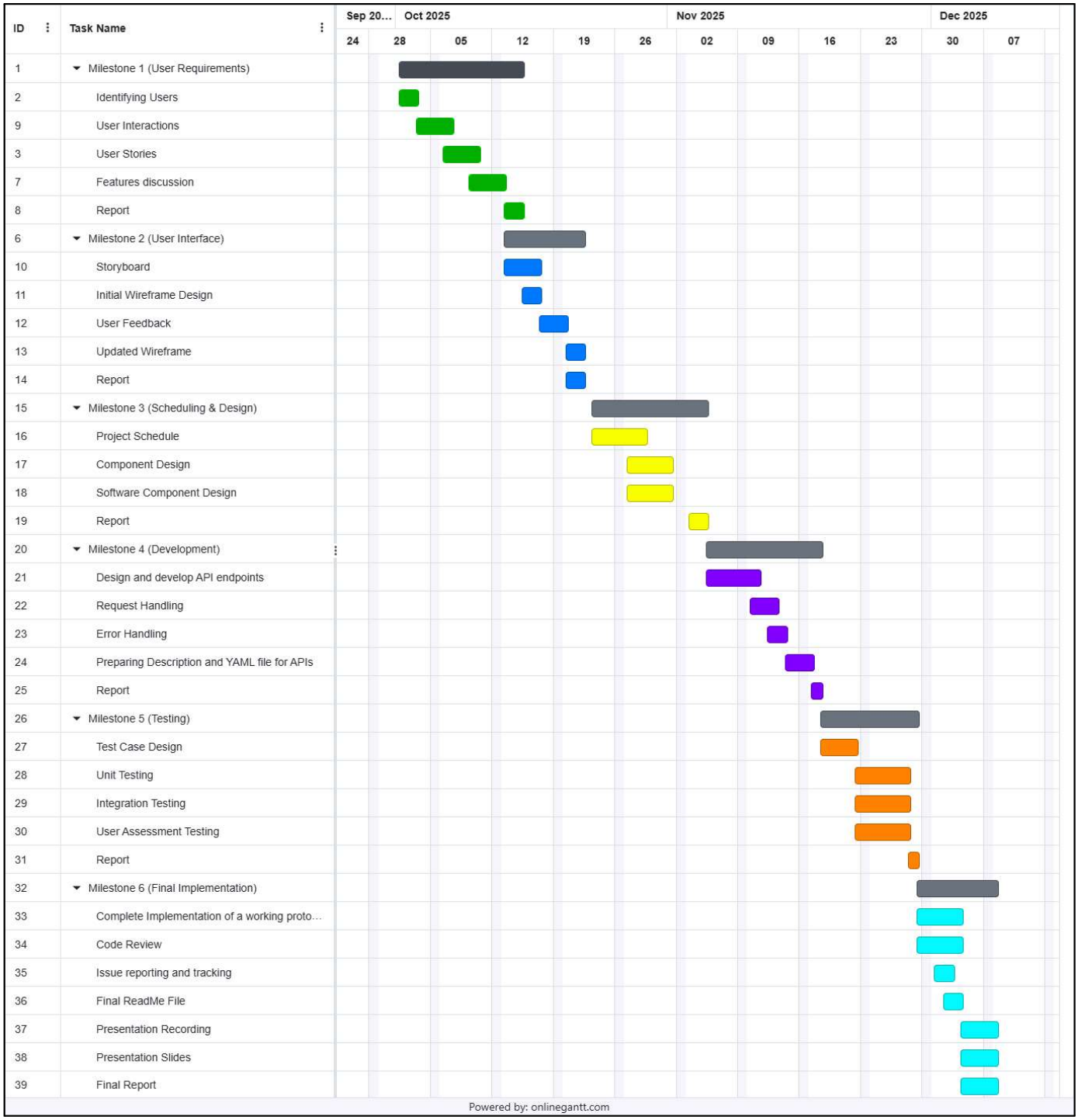
TO DO 7

Name	Assignee	Due date	Priority
Complete Implementation of a working prototype	2 TJ 2 PJ +3	12/4/25	
Code Review	PJ	12/4/25	
Issue Reporting and Tracking	R 2	12/4/25	
Final ReadMe File	PJ	12/4/25	
Presentation Slides	R	12/7/25	
Presentation Video	R	12/7/25	
Final Report	2 TJ 2 PJ +3	12/7/25	

Add Task

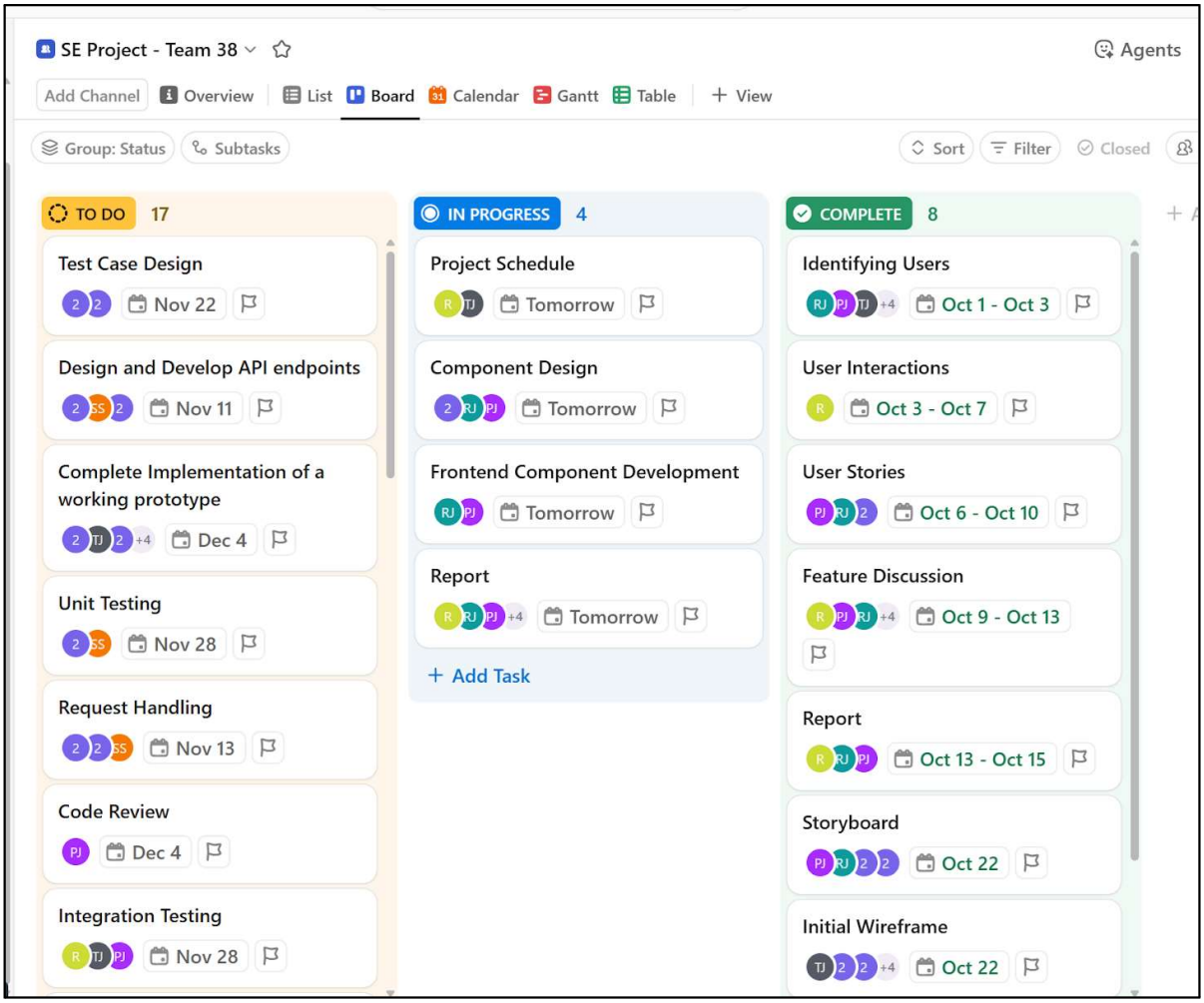
3.2 Gantt Chart

Below is the visual representation of our project schedule through a Gantt Chart. The sprints have been designed in such a way that the dependency of each component is satisfied. The scheduling and management aspects of the project were coordinated using the tool ClickUp.



3.3 Scrum Board

The Scrum board tracks each task in the backlog as it moves from ‘To Do’ status through ‘In Progress’ and finally to ‘Complete’ status.



3.4 Scrum Meetings

The scrum meetings were scheduled thrice a week and at critical progress checkpoints.

Scrum Meeting Schedule:

● **Frequency:** Every **Monday**, **Wednesday** and **Friday**, **9:00 – 9:30 PM**

Below is the meeting minutes of some of the scrum meetings conducted for the project, formatted for clarity and alignment with standard software engineering documentation practices:

Minutes of Meeting:

✦ Scrum Meeting | Oct 8, 2025

Attendees: RISHAV , PALASH JOHRI , RAGHAVENDRA NARAYAN JHA , Prabhu , OMKAR SHANKAR PAWAR , TREASA JANET

Main Points Discussed:

1. **Interview Insights:**
    - Rishav shared findings from his interview with a Teaching Assistant (TA).
    - Students often rely on external sources for course-related information due to:
        - Language barriers
        - Lack of engagement with official communication channels
2. **Focus on Target Users:**
    - Palash emphasized the importance of prioritizing Teaching Assistants (TAs) as key users for further data collection.
    - Suggested using synthetic data to model and understand TA-specific challenges more effectively.
3. **Regional Language Support:**
    - The team discussed the need for regional language integration in communication tools to enhance accessibility and inclusiveness.
4. **User Research and Outreach:**
    - The team agreed to conduct additional client and user interviews to refine user stories and validate assumptions.
    - Plans were made to reach out to course instructors and Bharti (Operations Head) for deeper operational insights.
5. **Contact Strategy for Bharti ma’am:**
    - Palash proposed that two team members draft a professional outreach email.
    - Omkar volunteered to lead the email drafting effort.
6. **Project Management and Coordination:**
    - Rishav will set up project management tools to track progress and milestones.
    - The team will maintain a shared Excel sheet to log outreach activities and avoid duplication.
7. **External Connections:**

- Raghavendra will reach out to his contacts in the SPG group for potential interviews and share updates with the team.

Action Items:

1. Draft outreach email for Bharti ma'am and course instructors — Omkar (lead), Palash (support)
2. Set up project management tools for progress tracking — Rishav
3. Create and maintain shared Excel sheet for documenting outreach activities — Treasa
4. Conduct additional client and TA interviews to refine user stories — All team members
5. Reach out for potential interviews — Raghavendra
6. Explore integration of regional language support in communication tools — Palash and team

✦ Scrum Meeting | Oct 13, 2025

Attendees: RISHAV, PALASH JOHRI, OMKAR SHANKAR PAWAR, Prabhu, RAGHAVENDRA NARAYAN JHA, SOHINI SARKAR, TREASA JANET

Main points discussed:

1. Updates on Project Deliverables and Instructor Outreach
    a. Raghavendra shared his ongoing research on Lang Graph and Lang Chain.
    b. Omkar reported Bharthi ma'am's suggestion to involve TAs in project insights.
    c. Treasa outlined project deliverables for Milestones 1 and 2, confirming ongoing interviews.
    d. Discussion included role allocation and progress tracking.
2. User Interviews and Project Milestones Discussion
    a. Rishav mentioned completion of three TA interviews; instructor interviews are pending.
    b. Omkar suggested involving students to capture diverse user perspectives.
    c. Treasa emphasized setting strict interview deadlines aligned with milestones.
    d. Palash encouraged collaboration for developing user stories and wireframes using interview findings.
3. Client Interview Preparation and Feature Proposals
    a. Palash discussed preparation for the upcoming client interview with Nidhish.
    b. Proposed integrating multimodal models as new project features.
    c. Raghavendra highlighted workflow enhancements for project clarity.
    d. Sohini volunteered to contribute via Figma prototypes.
    e. Palash coordinated responsibilities for storyboard and wireframe development.

Action items:

1. Reach out to instructors and request interviews - Omkar
2. Schedule a meeting for storyboard & wireframe planning - Palash Johri (with Raghavendra & Rishav)

3. Write user stories based on completed TA interviews. - Raghavendra Narayan Jha & Rishav
4. Summarize key pain points and reasoning for user stories - Rishav
5. Maintain Excel sheet with interview status and tracking - Treasa Janet

✦ Scrum Meeting | Oct 18, 2025

Attendees: NIDHISH, RISHAV, PALASH JOHRI, OMKAR SHANKAR PAWAR, Prabhu, RAGHAVENDRA NARAYAN JHA, SOHINI SARKAR, TREASA JANET

Main points discussed:

1. Palash reviewed the progress of the design and storyboard, inviting inputs from the team.
2. Nidhish advised categorizing features into top-level feature statements to simplify presentation and enhance clarity.
3. Emphasis on including login pages and user onboarding screens as part of the design.
4. Discussion on timeline adjustments, with a new deadline of October 22 for finalizing wireframes.
5. The team decided to share a comprehensive documentation package by Monday, integrating feedback and final deliverables.

Action items:

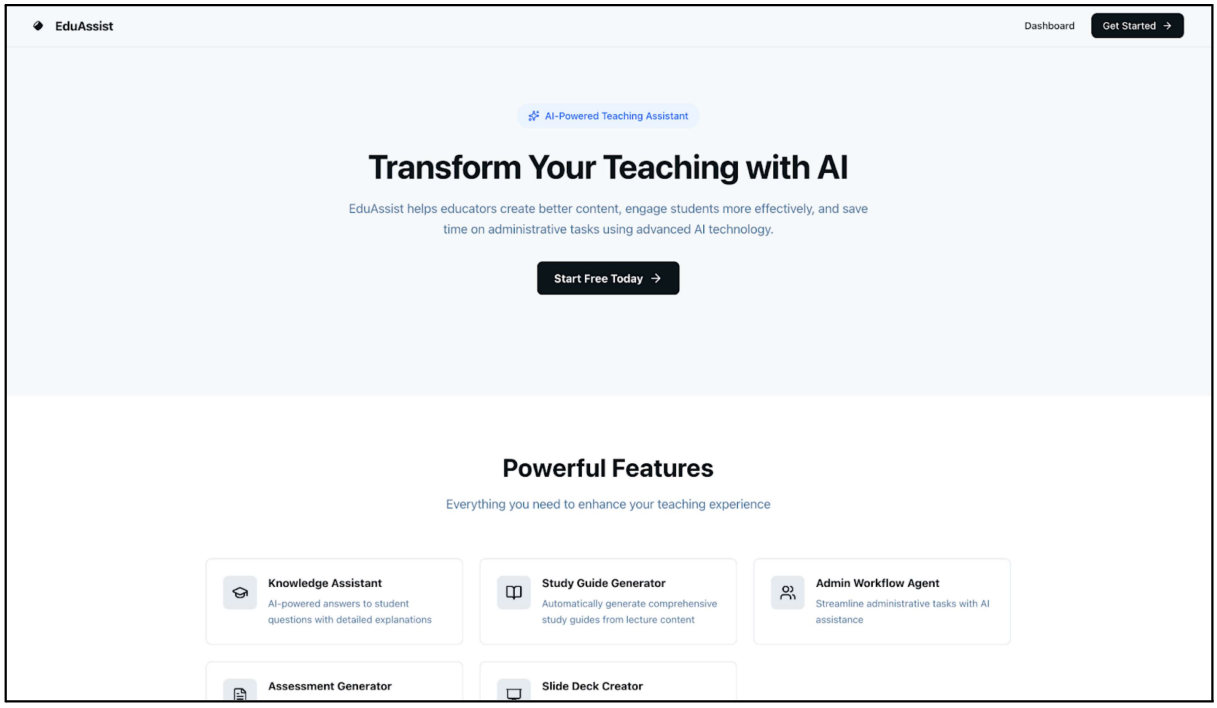
1. Share Figma link and storyboard PPT with the team for feedback.
2. Add login pages and necessary updates to the prototype.
3. Prepare the feedback video after wireframe finalization.
4. Create storyboard aligning with milestone one deliverables.
5. Compile and share the completed document with Nidhish.

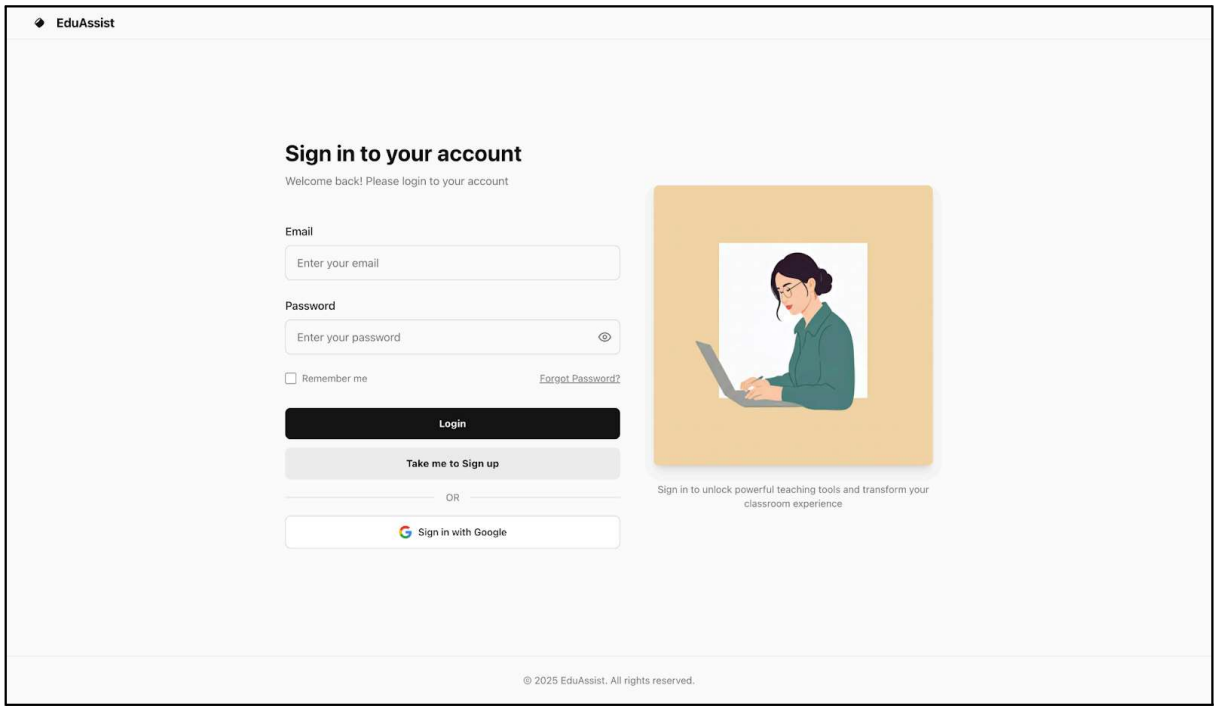
Proof of Interaction

4. Project Design

4.1 Frontend

Landing & Authentication





EduAssist

Reset your password

Enter your email address and we'll send you a verification code

Email Address

Enter your email

Send Verification Code →

Remember your password? Back to login

We'll help you recover your account with a simple verification process

EduAssist

Create your account

Name

Enter your name

Email

Enter your email

Password

Enter your password

Course Code

Enter course code

TA Role Verification

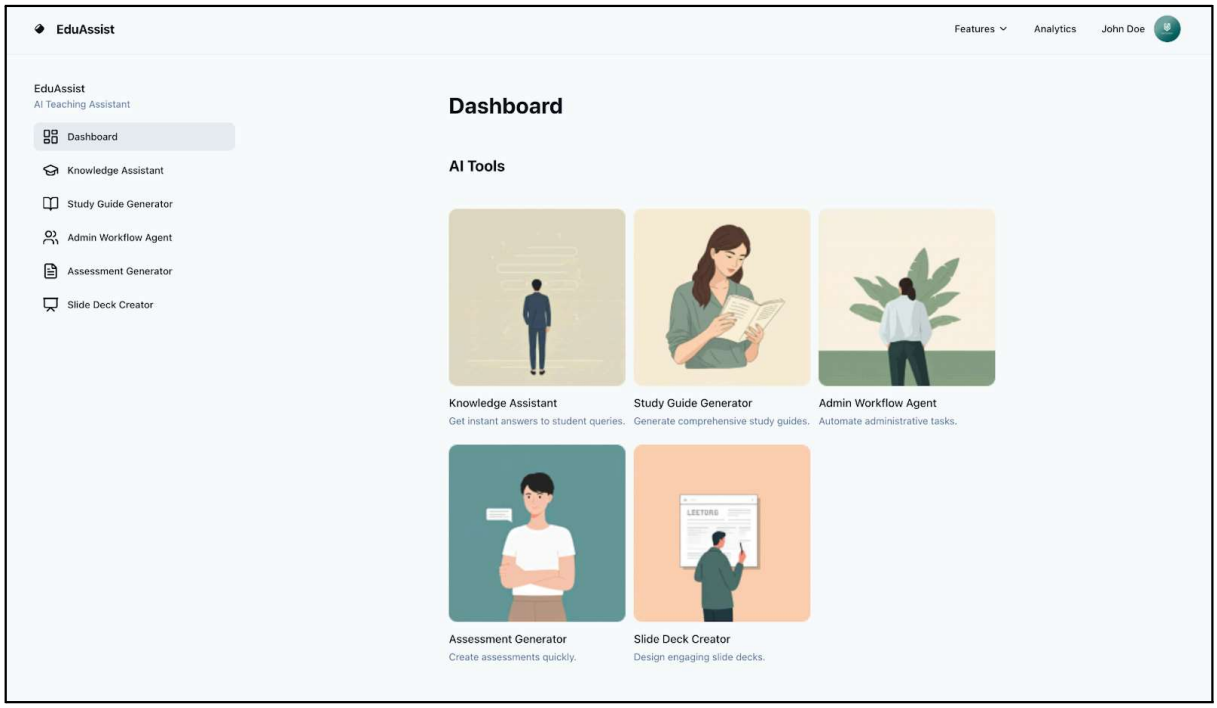
Select role

I agree to the Terms & Conditions and Privacy Policy

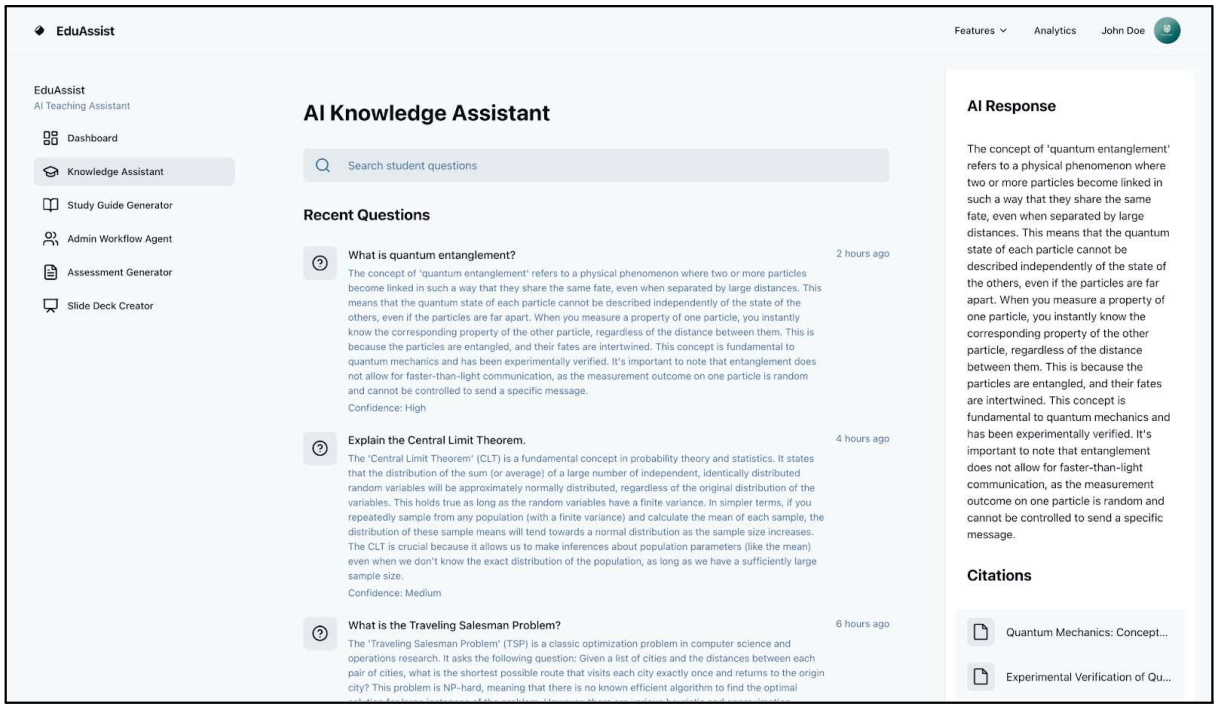
Create Account

Already have an account? Login

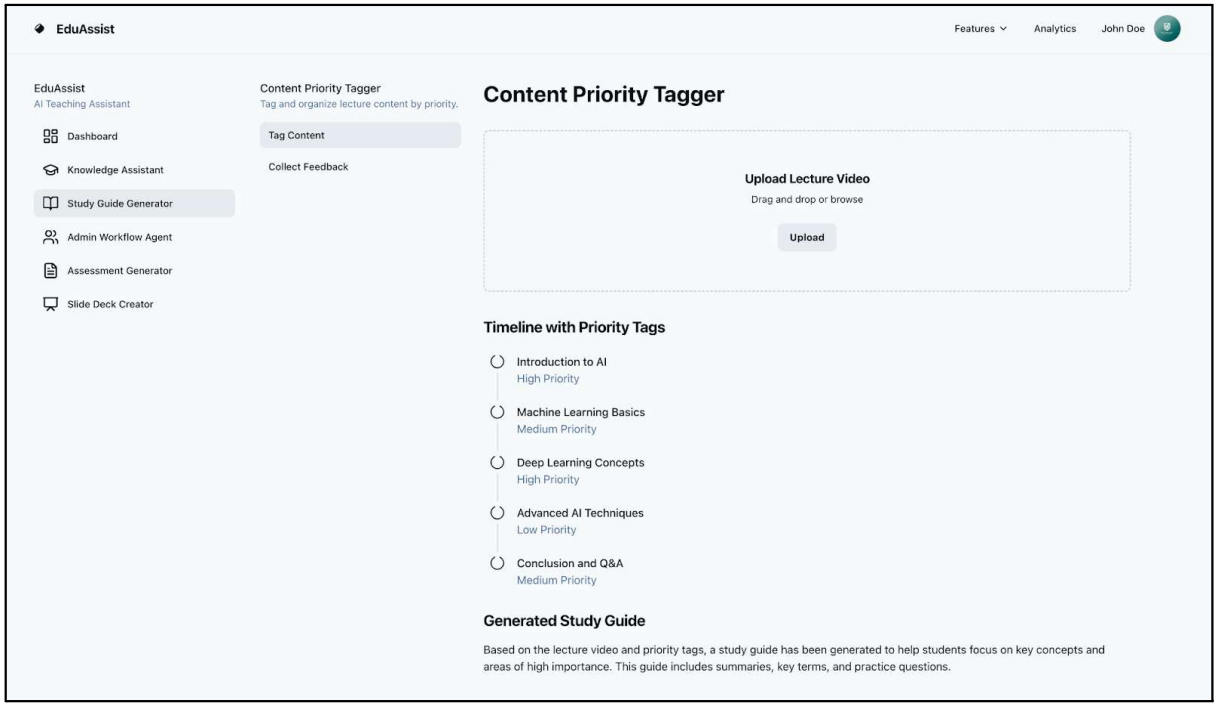
Join thousands of educators and TAs using EduAssist to enhance their teaching



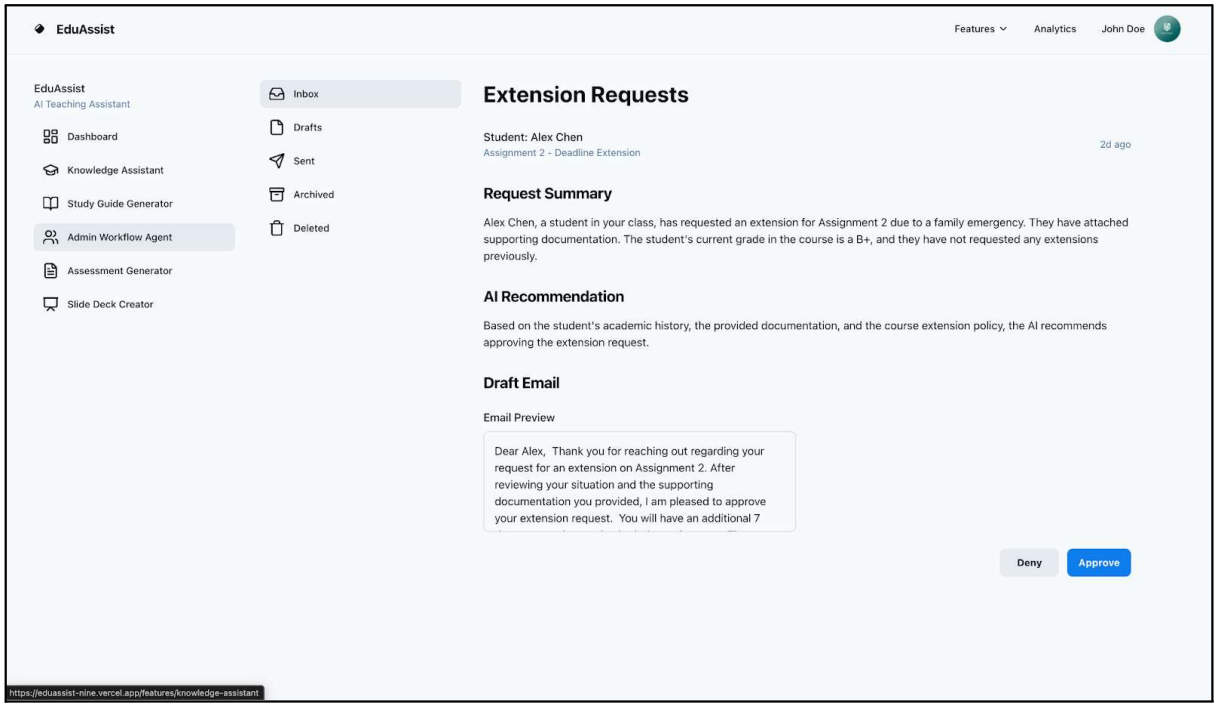
Knowledge Assistant



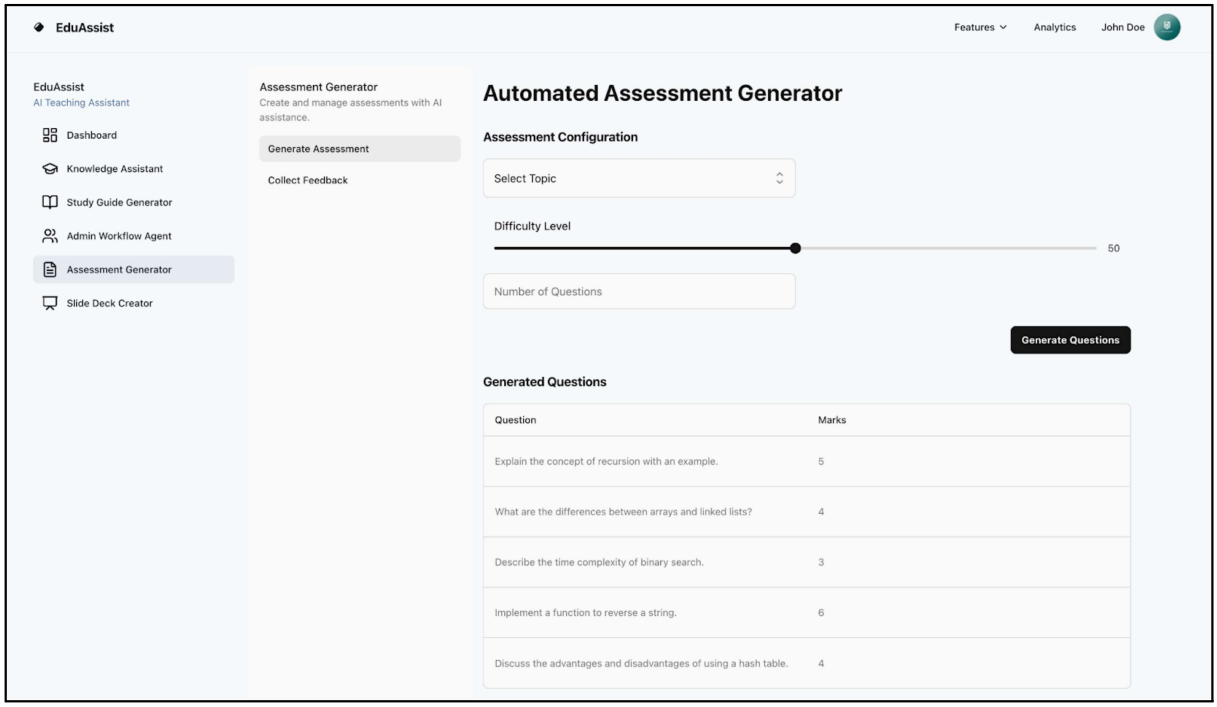
Study Guide Generator



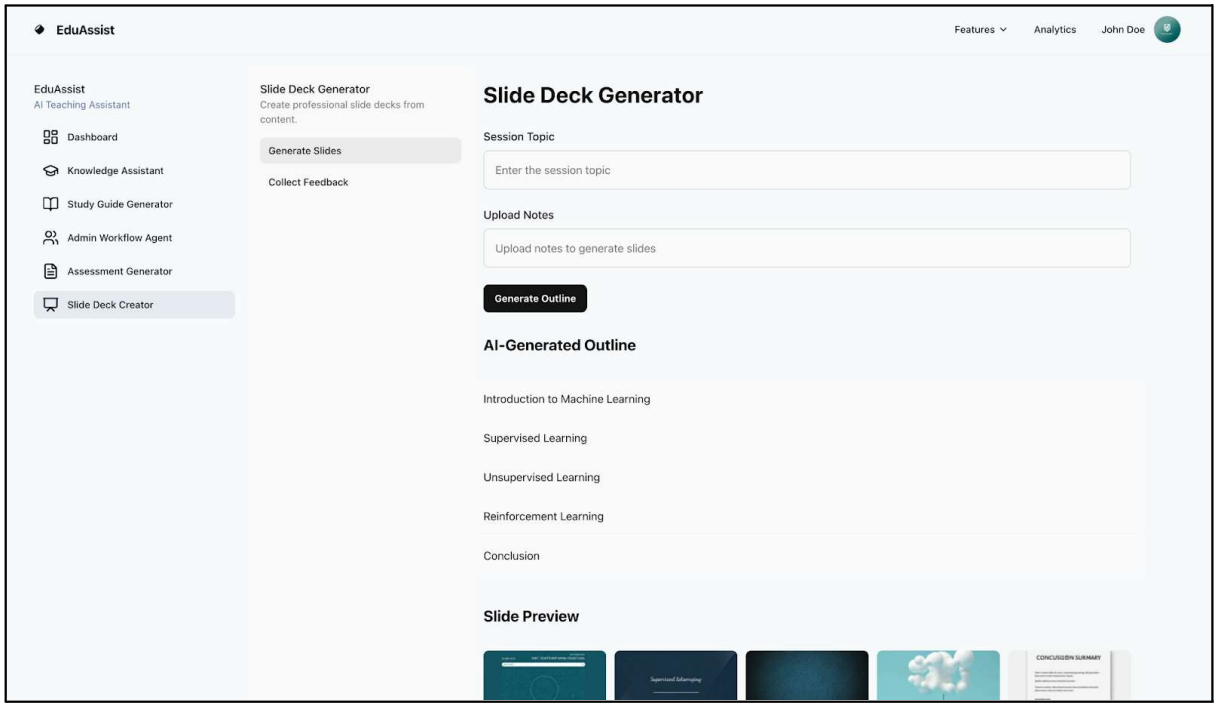
Admin Workflow Agent



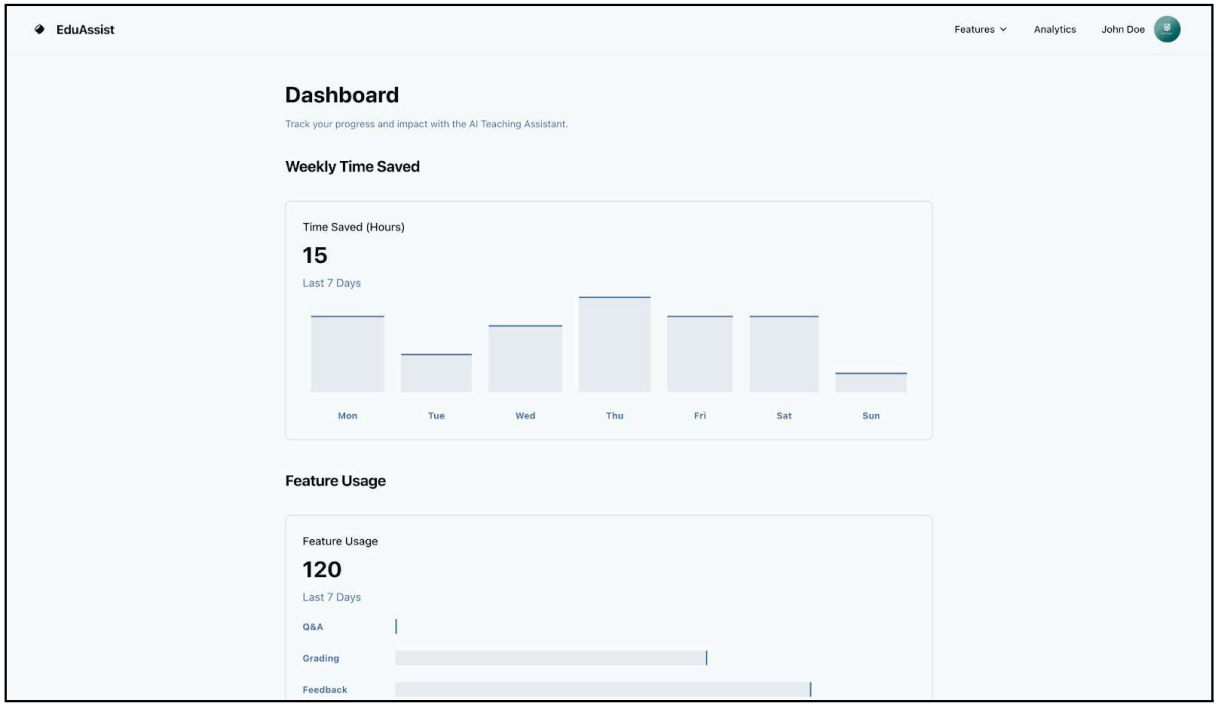
Assessment Generator



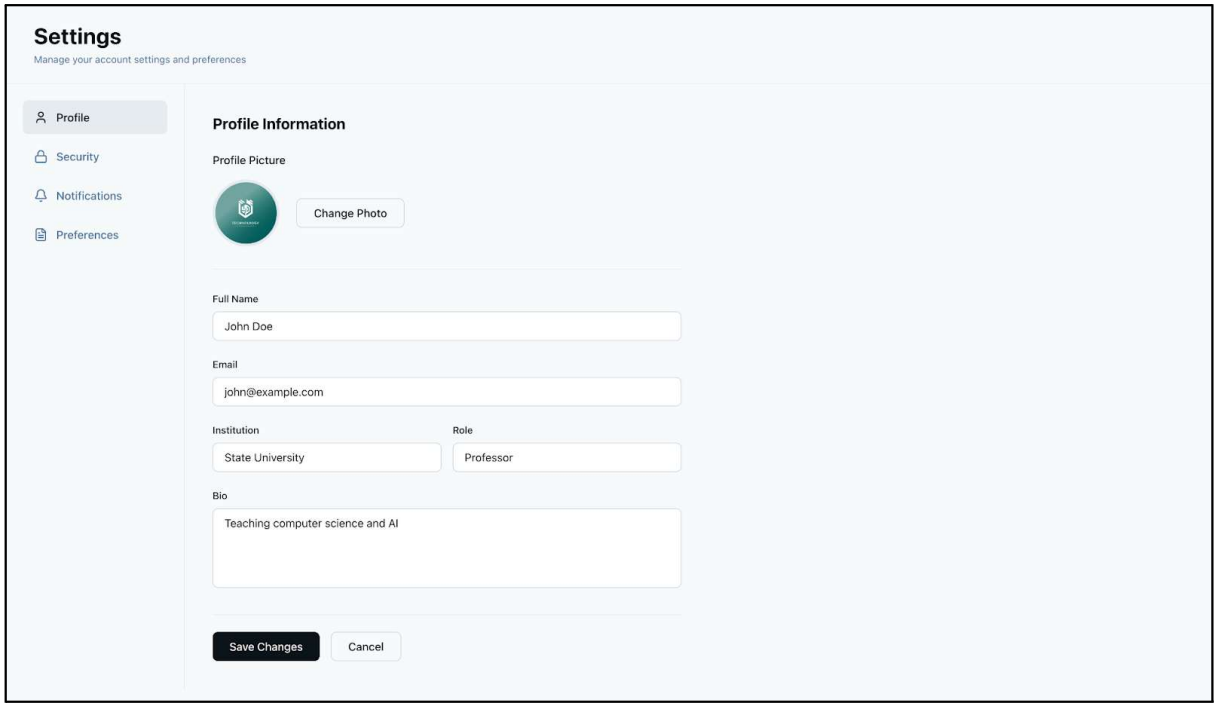
Slide Deck Creator



Analytics



User Settings



4.2 Components of the Project

FRONTEND DESIGN

1. Programming Language & Framework

TypeScript 5.3+ - Type-safe development with compile-time checking - Enhanced IDE support and code maintainability - Interface definitions for API contracts

Next.js 14 (App Router) - Modern React framework with server-side rendering - File-based routing and automatic code splitting - Vercel integration for seamless deployment

2. Architecture Overview

Component-based architecture with clear separation: - **Pages Layer:** Authentication, Dashboard, Feature pages - **Components Layer:** Reusable UI components (layout, forms, widgets) - **Services Layer:** API client, authentication utilities, type definitions - **Hooks Layer:** Custom React hooks for data fetching and state management

3. UI Components & Styling

shadcn/ui Component Library - Pre-built accessible components (Button, Card, Dialog, Form, Input, Select) - Built on Radix UI + Tailwind CSS - Dark mode support and WCAG 2.1 compliance

Tailwind CSS 3.3+ - Utility-first CSS framework for rapid development - Responsive breakpoints (mobile-first) - Custom theme configuration

4. Data Management

Axios HTTP Client - Request/response interceptors for auth tokens - Automatic error handling and retry logic - JSON transformation

State Management - React Hooks (useState, useContext, useReducer) - Custom hooks for data fetching (useAPI, useChat, useAssessment) - Context API for global state (authentication, user profile)

5. Key Frontend Features

- Responsive design (mobile, tablet, desktop)
- Real-time chat interface with typing indicators
- Document drag-and-drop upload
- Interactive charts and analytics
- Form validation with instant feedback
- Loading states and error boundaries

BACKEND DESIGN

1. Programming Language & Framework

Python 3.10+ - Rich ecosystem for GenAI integration - Strong typing with Pydantic - Async/await for concurrent operations

FastAPI 0.104+ - Modern async web framework - Automatic OpenAPI/Swagger documentation - Built-in request validation - High performance (async-first)

Uvicorn ASGI Server - Production-ready performance - WebSocket support for streaming - Auto-reload in development

2. Architecture Overview

Layered Modular Monolith: - **API Layer:** Route handlers and request validation - **Service Layer:** Business logic and GenAI orchestration - **Repository Layer:** Database CRUD operations - **Model Layer:** ORM entities and relationships

Clear separation ensures testability and maintainability.

3. Database & ORM

PostgreSQL 15+ - Relational database with ACID properties - Hosted on Neon (free cloud PostgreSQL) - Full-text search and JSON support

SQLAlchemy 2.0+ - Type-safe ORM with async queries **-** Relationship management and lazy loading - Alembic for database migrations

Core Data Models: - User hierarchy (User, TeachingAssistant, Student) - Course and content (Course, Document, Lecture) - Conversations (Conversation, Message) - Assessments (Assessment, Question) - Administrative (ExtensionRequest, Assignment) - Feedback (Feedback, FeedbackAnalytics)

4. Authentication & Authorization

JWT Token-Based Authentication - Access token (30 min expiry) - Refresh token (7 day expiry) - Token stored securely (localStorage + HTTP-only cookies)

Role-Based Access Control (RBAC) - Roles: TA, Student, Admin - Permission enforcement at endpoint level - Resource-level authorization

Security Measures - bcrypt password hashing (12 salt rounds) - Token signature verification - CORS configuration for allowed origins

5. GenAI Integration

Gemini API (Google) - Models: gemini-2.5-pro (primary), gemini-2.0-flash (fast) - Free tier: 15 RPM, 1.5M tokens/day - Multi-modal capabilities (text, images) - Streaming response support

LangChain Framework - LLM orchestration for complex workflows - Agentic reasoning with tool use - Prompt template management - Conversation memory

ChromaDB (Vector Database) - Semantic similarity search - Document embedding storage - Collection management per course - In-memory or persistent mode

6. Key Backend Features

- Async request handling for high concurrency
- Structured logging for debugging
- Rate limiting to prevent abuse
- Comprehensive error handling
- Database connection pooling
- Environment-based configuration

DESIGN COMPONENTS & INTERACTIONS

This section details the major system components and how they interact.

Component 1: Authentication System

Purpose: Secure user authentication and session management

Frontend Components: - LoginForm: Email/password input with validation - SignupForm: Registration with role selection - AuthContext: Global authentication state management

Backend Components: - AuthService: Credential validation and token generation - UserRepository: Database operations for user data - Security utilities: Password hashing and token verification

Interaction Flow:

User enters credentials in LoginForm
↓
Frontend validates input format
↓
Axios sends POST request to /auth/login
↓
Backend AuthService validates credentials
↓
UserRepository queries database for user
↓
Password hash verified with bcrypt
↓
JWT tokens generated (access + refresh)
↓
Tokens returned to frontend
↓
Frontend stores tokens securely
↓
Axios interceptor adds token to all future requests
↓
User redirected to dashboard

Key Interactions: - Frontend ↔ Backend: HTTP requests with JSON payloads - Backend ↔ Database: SQL queries via SQLAlchemy ORM - Token validation on every protected API call

---

Component 2: Knowledge Assistant (RAG Chatbot)

Purpose: AI-powered Q&A system with document retrieval

Frontend Components: - ChatInterface: Real-time message display and input - DocumentUpload: Drag-and-drop file upload with progress - ConversationList: History of past conversations - ConfidenceIndicator: Visual display of AI confidence

Backend Components: - KnowledgeAssistantService: Query processing orchestration - RAGService: Document processing and retrieval - GeminiClient: LLM integration for embeddings and completions - VectorStoreRepository: ChromaDB operations - ConversationRepository: Chat history persistence

Interaction Flow (Document Upload):

TA uploads course PDF via DocumentUpload
↓
Frontend sends file via multipart form
↓
Backend saves file to storage
↓
RAGService extracts text from PDF
↓
Text split into 512-token chunks with overlap
↓
GeminiClient generates embeddings for each chunk
↓
Embeddings stored in ChromaDB collection
↓
Document marked as processed in database
↓
Frontend shows "Processed" status

Interaction Flow (Student Query):

Student types question in ChatInterface
↓
Frontend sends message to backend
↓
KnowledgeAssistantService receives query
↓
Query converted to embedding via Gemini
↓
VectorStore performs similarity search
↓
Top 5 relevant document chunks retrieved
↓
Context + query sent to Gemini for completion
↓
LLM generates answer with citations

↓
Confidence score calculated based on context match
↓
Message saved to database with sources
↓
Response returned to frontend
↓
ChatInterface displays message with confidence badge
↓
If confidence < 60%: Show "Escalate to TA" option

Key Interactions: - Frontend ↔ Backend: REST API for messages and documents - Backend ↔ Gemini API: Embeddings and text generation - Backend ↔ ChromaDB: Vector similarity search - Backend ↔ PostgreSQL: Conversation and message persistence

---

Component 3: Assessment Generator

Purpose: Automated assessment creation with AI

Frontend Components: - ConfigurationPanel: Topic, difficulty, and question type selection - QuestionReview: Table display of generated questions with edit controls - DifficultySetter: Slider for difficulty calibration - ExportOptions: PDF and LMS export formats

Backend Components: - AssessmentGeneratorService: Question generation orchestration - GeminiClient: LLM for question creation - AssessmentRepository: Database persistence - QuestionBankRepository: Reusable question storage

Interaction Flow:

TA selects configuration (topics, difficulty, count)
↓
Frontend sends generation request
↓
Backend creates Assessment record
↓
For each question needed:
Prompt sent to Gemini with topic and difficulty
↓
Gemini generates question with options/answer
↓
Question quality validated
↓
Difficulty calibrated based on complexity
↓
Marks allocated based on difficulty
↓
Question saved to database
↓
Question added to reusable bank
↓
All questions returned to frontend

↓
QuestionReview displays editable table
↓
TA can regenerate individual questions
↓
TA publishes or exports final assessment

Key Interactions: - Frontend ↔ Backend: RESTful assessment CRUD operations - Backend ↔ Gemini API: Dynamic question generation - Backend ↔ Database: Assessment and question persistence - Question reuse: Bank checks for similar questions before generating

---

Component 4: Workflow Automation Agent

Purpose: Intelligent processing of student extension requests

Frontend Components: - RequestInbox: Table of pending requests with filters - AIAnalysis: Display of agent reasoning and recommendation - EmailDraft: Auto-generated email with edit capability - DecisionPanel: Approve/Deny/Request Info buttons

Backend Components: - WorkflowAgentService: Agentic workflow orchestration - LangChain Agent: Multi-step reasoning with tools - GeminiClient: LLM for agent reasoning - ExtensionRequestRepository: Request data access

Agent Tools: - StudentHistoryTool: Retrieves past submission records - PolicyCheckerTool: Verifies compliance with course policies - SimilarCasesTool: Finds comparable past requests

Interaction Flow:

Student submits extension request
↓
Request stored in database with PENDING status
↓
TA opens request in WorkflowAgent interface
↓
TA clicks "Analyze" button
↓
Frontend requests AI analysis
↓
Backend WorkflowAgentService receives request
↓
LangChain Agent begins reasoning:
↓
Step 1: Agent decides to check student history
Calls StudentHistoryTool
Tool queries database for submission records
Returns: on-time count, late count, past extensions
↓
Step 2: Agent decides to verify policy compliance
Calls PolicyCheckerTool
Tool performs RAG search on policy documents
Returns: relevant policy sections and compliance status

↓
Step 3: Agent decides to find similar cases
Calls SimilarCasesTool
Tool queries database for similar requests
Returns: 2-3 comparable cases with decisions
↓
Step 4: Agent synthesizes information
Gemini generates recommendation with reasoning
Outputs: APPROVE/DENY/REQUEST_INFO + confidence
↓
Analysis saved to database
↓
Frontend displays reasoning in AIAnalysis panel
↓
TA reviews and makes final decision
↓
Email draft auto-generated based on decision
↓
TA reviews/edits and sends email

Key Interactions: - Frontend ↔ Backend: Request management API - LangChain Agent ↔ Tools: Sequential tool calls with results - Tools ↔ Database/RAG: Data retrieval operations - Backend ↔ Gemini: Agent reasoning and email generation

---

Component 5: Feedback System

Purpose: Collect and analyze user feedback on AI features

Frontend Components: - FeedbackWidget: Thumbs up/down with comment box - TAFeedbackDashboard: Queue of feedback items - AnalyticsView: Charts showing feedback trends

Backend Components: - FeedbackService: Feedback processing and analytics - FeedbackRepository: Database operations - NotificationService: Alert TAs to critical feedback

Interaction Flow:

Student receives AI response in chat
↓
FeedbackWidget appears below response
↓
Student clicks thumbs down
↓
Comment box expands
↓
Student types issue and submits
↓
Frontend sends feedback to backend
↓
FeedbackService creates feedback record
↓
If rating < 3: Alert TA via NotificationService
↓

Feedback appears in TA dashboard
↓
TA reviews feedback and investigates issue
↓
TA takes corrective action (fix document, update response)
↓
TA marks feedback as reviewed
↓
Status updated in database
↓
Analytics aggregated for reporting

Key Interactions: - Frontend ↔ Backend: Feedback submission and retrieval - Backend ↔ Database: Feedback persistence and queries - Feedback triggers: Notifications sent to TAs for critical issues - Analytics: Periodic aggregation of feedback metrics

---

Component 6: Cross-Cutting Concerns

Purpose: Middleware and shared services

Components:

CORS Middleware - Validates origin of incoming requests - Allows specific frontend domains - Blocks unauthorized cross-origin requests

Authentication Middleware - Extracts JWT token from Authorization header - Verifies token signature and expiry - Injects current user into request context - Returns 401 if invalid token

Rate Limiting Middleware - Tracks requests per IP address - Enforces limits (100 requests/min general, 15 RPM for Gemini) - Returns 429 if limit exceeded

Request Logging - Logs all incoming requests with metadata - Records response status and duration - Stores errors with stack traces

Error Handling - Catches all exceptions globally - Formats errors consistently - Maps exception types to HTTP status codes - Returns structured error responses

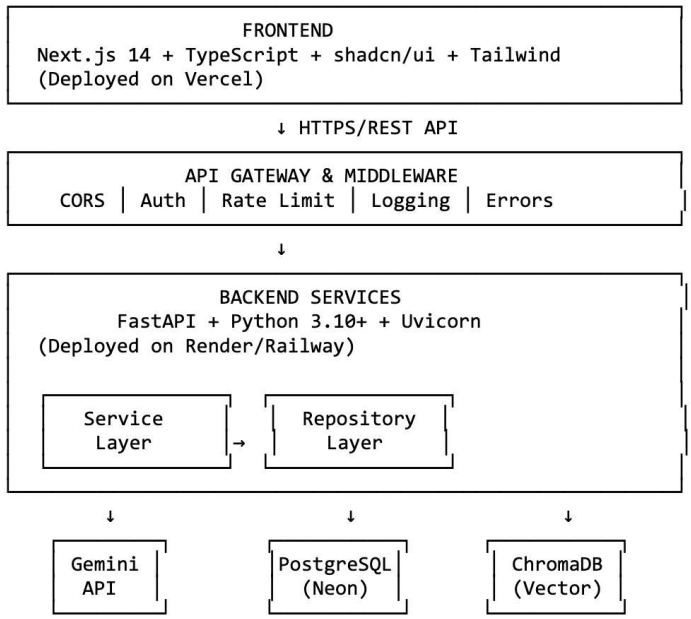
Interaction Flow (Protected Request):

Frontend makes authenticated request
↓
CORS Middleware validates origin
↓
Request Logging records incoming request
↓
Authentication Middleware extracts token
↓
Token verified (signature + expiry)
↓
User context injected into request
↓
Rate Limiting checks request count

↓
Route handler processes request
↓
Response returned
↓
Response Logging records outcome
↓
Error Handler (if error occurred)

SYSTEM ARCHITECTURE

High-Level Architecture



Component Communication Patterns

Frontend → Backend: - REST API calls via Axios - JSON request/response format - Bearer token authentication - WebSocket for streaming (future)

Backend → Gemini API: - HTTP POST requests - JSON payload with prompts - API key authentication - Rate limit: 15 requests/minute

Backend → PostgreSQL: - SQL queries via SQLAlchemy ORM - Connection pooling (5-10 connections) - Async queries for performance - Transaction management

Backend → ChromaDB: - Python client library - Collection-based organization - Vector similarity queries - In-memory or persistent storage

Data Flow Examples

User Authentication: Frontend → Backend → Database → Backend → Frontend

RAG Query: Frontend → Backend → ChromaDB (similarity) → Gemini (completion) → Database (save) → Backend → Frontend

Assessment Generation: Frontend → Backend → Gemini (generate questions) → Database (save) → Backend → Frontend

Workflow Analysis: Frontend → Backend → LangChain Agent → Multiple Tools (DB/RAG queries) → Gemini (reasoning) → Database (save) → Backend → Frontend

Security Layers

Layer 1: Network - HTTPS/TLS encryption - CORS protection - Rate limiting

Layer 2: Authentication - JWT token verification - Token expiry enforcement - Refresh token rotation

Layer 3: Authorization - Role-based access control - Resource-level permissions - Endpoint protection

Layer 4: Data - Input validation (Pydantic) - SQL injection prevention (ORM) - XSS protection (output sanitization) - Prompt injection prevention

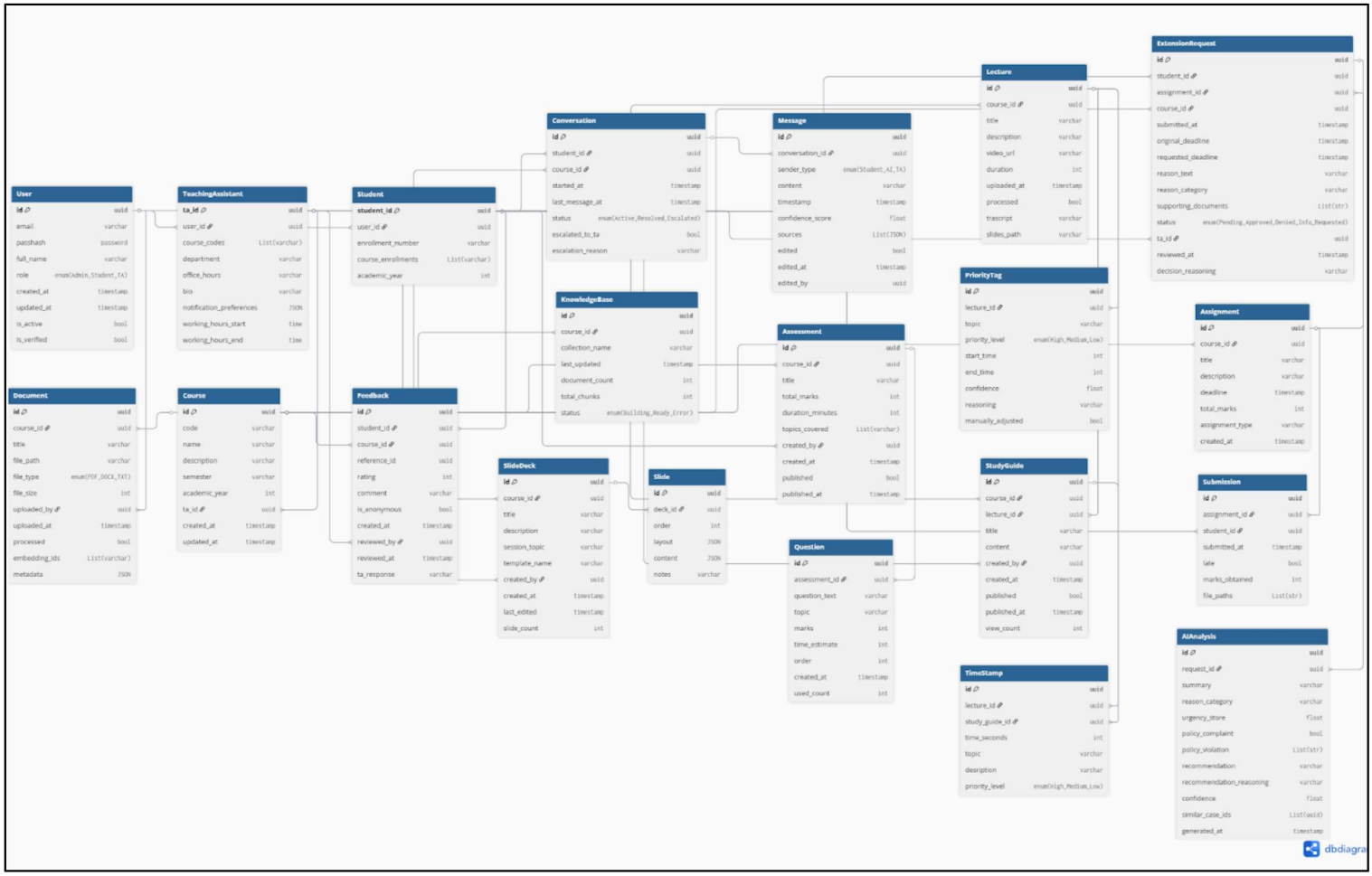
Scalability Considerations

Horizontal Scaling: - Stateless backend (JWT tokens) - Database connection pooling - Async operations for concurrency

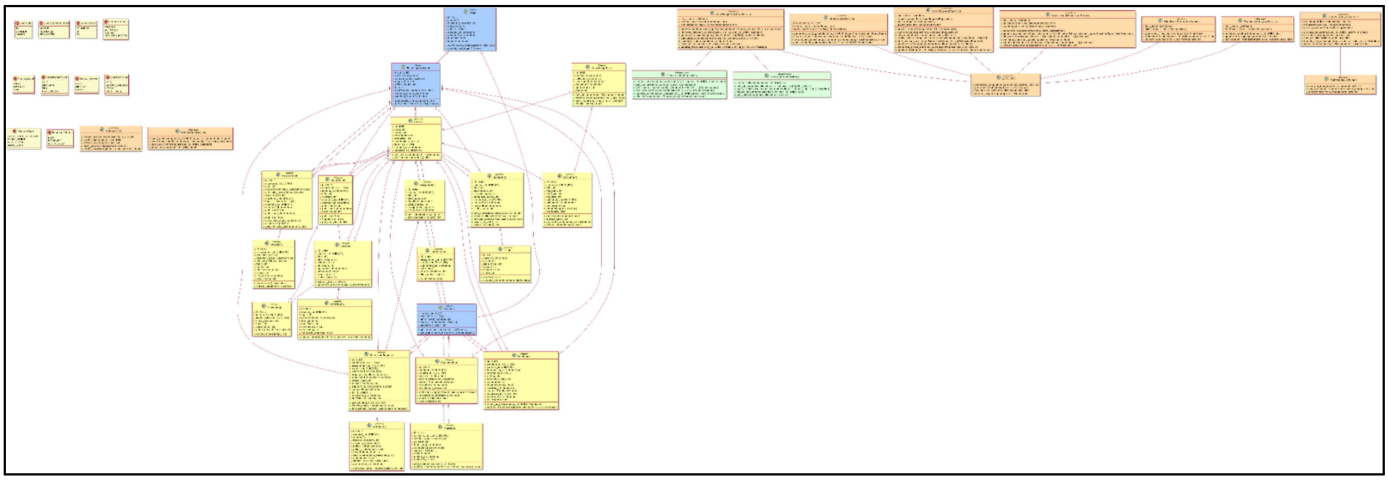
Performance Optimization: - Response caching (future: Redis) - Database query optimization - Vector search indexing - Lazy loading of relationships

Monitoring & Observability: - Structured logging (JSON format) - Error tracking and alerts - Performance metrics - API usage analytics

4.4 ER Diagram *(click here and zoom for clear view)*



4.5 Class(UML) Diagram *(click here and zoom for clear view)*



Milestone - 4

APIs | YAML

5. API Endpoints

Routers per feature: /knowledge-assistant, /study-guides, /assessments, /slide-decks, /admin-workflow, /feedback, /auth

- CRUD endpoints for create/read/update/delete objects and requests

6. YAML

Original yaml file

Milestone - 5

Testing

1. **Assessment Generator**
   Test File: test_assessment.py
   APIs Tested
   - GET /assessments/{assessment_id}

   1. **Test Case: Get Assessment – Success API:**
      GET /assessments/1
      - Preconditions
        ● Assessment id = 1 exists
        ● created_by = 10 (current instructor user)
      - Input:
        No body parameters (path only)
      - Expected Output
        Status: 200 OK
        {
        "id": 1,
        "title": "Unit Test Assessment",
        "created_by": 10
        }
      - Behavior Validated
        ● Returns full assessment details
        ● Ensures authenticated instructor can access their own assessment

   2. **Test Case: Get Assessment – Not Found (FAILURE)**
      GET /assessments/9999
      - Preconditions
        ● Assessment with ID 9999 does not exist
      - Expected Output
        Status: 404
        {
        "detail": "Assessment not found"
        }
      - Behavior Validated
        ● Proper 404 handling
        ● Validates missing resource error flow

2. **Knowledge Assistant – Conversation System**

Test File: test_knowledge.py

APIs Tested

- POST /knowledge/conversations
- GET /knowledge/conversations/{conversation_id}

1. **Create Conversation – Success API:**

POST /knowledge/conversations

- Input

{
"course_id": 1
}

- Preconditions
  - Logged-in student: id = 101
- Expected Output

Status: 201

{
"id": 1,
"course_id": 1,
"student_id": 101,
5

"messages": []
}

- Behavior Validated
  - Conversation gets created for the student
  - Ensures correct ownership

2. **Get Conversation – Forbidden (FAILURE) API:**

GET /knowledge/conversations/1

- Preconditions
  - Conversation belongs to student_id = 999
  - Current student = 101
- Expected Output

Status: 403

{
"detail": "Not authorized"
}

- Behavior Validated
  - Access is blocked when user does not own the conversation

3. Study Guide Generator

Test File: test_study_guide.py

APIs Tested

- POST /study-guides/
- GET /study-guides/{guide_id}

1. Create Study Guide – Success

- Input

  {
  "youtube_url": "https://youtube.com/watch?v=abc123", "title": "Test Guide",
  "course_id": 1,
  "priority_segments": []
  }
- Mocked Preconditions
  - Extracted video ID: "abc123"
  - Metadata returned: title, duration, thumbnail ● Captions available
  - CRUD returns guide id = 1
- Expected Output

  Status: 201

  {
  "id": 1,
  "title": "Test Guide"
  }
- Behavior Validated
  - Successfully creates a study guide when required metadata exists

2. Create Study Guide – No Captions (FAILURE)

- Mock
  - check_captions_available → False
- Expected Output

  Status: 400

  {
  "detail": "Video does not have captions available" }
- Behavior Validated
  - System rejects videos without captions
  - Prevents unusable study guides

4. Slide Deck Generator

Test File: test_slide_deck.py

APIs Tested

● POST /slide-decks/

● GET /slide-decks/{deck_id}

1. Create Slide Deck – Success

- Input

  {
  "course_id": 1,
  "title": "Intro Slides",
  "source_type": "DOCUMENT",
  "source_id": 5
  }
- Expected Output

  9

  Status: 201

  {
  "id": 1,
  "title": "Intro Slides"
  }
- Behavior Validated

  ● Slide deck gets created from a document source

2. Slide Deck – Not Found (FAILURE) API:

GET /slide-decks/9999

- Expected Output

  Status: 404

  {
  "detail": "Slide deck not found"
  }
- Behavior Validated

  ● Ensures proper missing-deck handling

5. Workflow Automation

Test File: test_workflow_agent.py

APIs Tested

● POST /workflow/workflow-requests/

● GET /workflow/workflow-requests/{id}

1. Create Workflow Request – Success

- Input

  {
  "course_id": 1,
  "description": "Need help setting grading rules", "priority": "MEDIUM"

}

- Expected Output
  Status: 201
  {
  "id": 1,
  "course_id": 1,
  "status": "PENDING"
  }
- Behavior Validated
  ● New workflow request is created and stored

2. Workflow Request Not Found – FAILURE:

GET /workflow/workflow-requests/9999

- Expected Output
  Status: 404
  {
  "detail": "Workflow request not found"
  }

6. Feedback System

Test File: test_feedback.py
APIs Tested
● POST /feedback/
● GET /feedback/

1. Create Feedback – Success

- Input
  {
  "rating": 4,
  "comment": "Helpful answer",
  "context": {
  "type": "chat",
  "id": 10
  }
  }
- Expected Output
  Status: 201
  {
  "id": 1,
  "rating": 4,
  "comment": "Helpful answer"
  }

2. Create Feedback – Invalid Rating (FAILURE)

- Input
  {
  13
  "rating": 11,
  "comment": "Too high",
  "context": { "type": "chat", "id": 10 }
  }
- Expected Output
  ● Status: 400 or 422 (Pydantic validation error)
- Behavior Validated
  ● Ensures rating boundaries

7. Knowledge Assistant – Chat Flow

Test File: test_knowledge_chat.py
API Tested
● POST /knowledge/chat

1. Chat – New Conversation Success

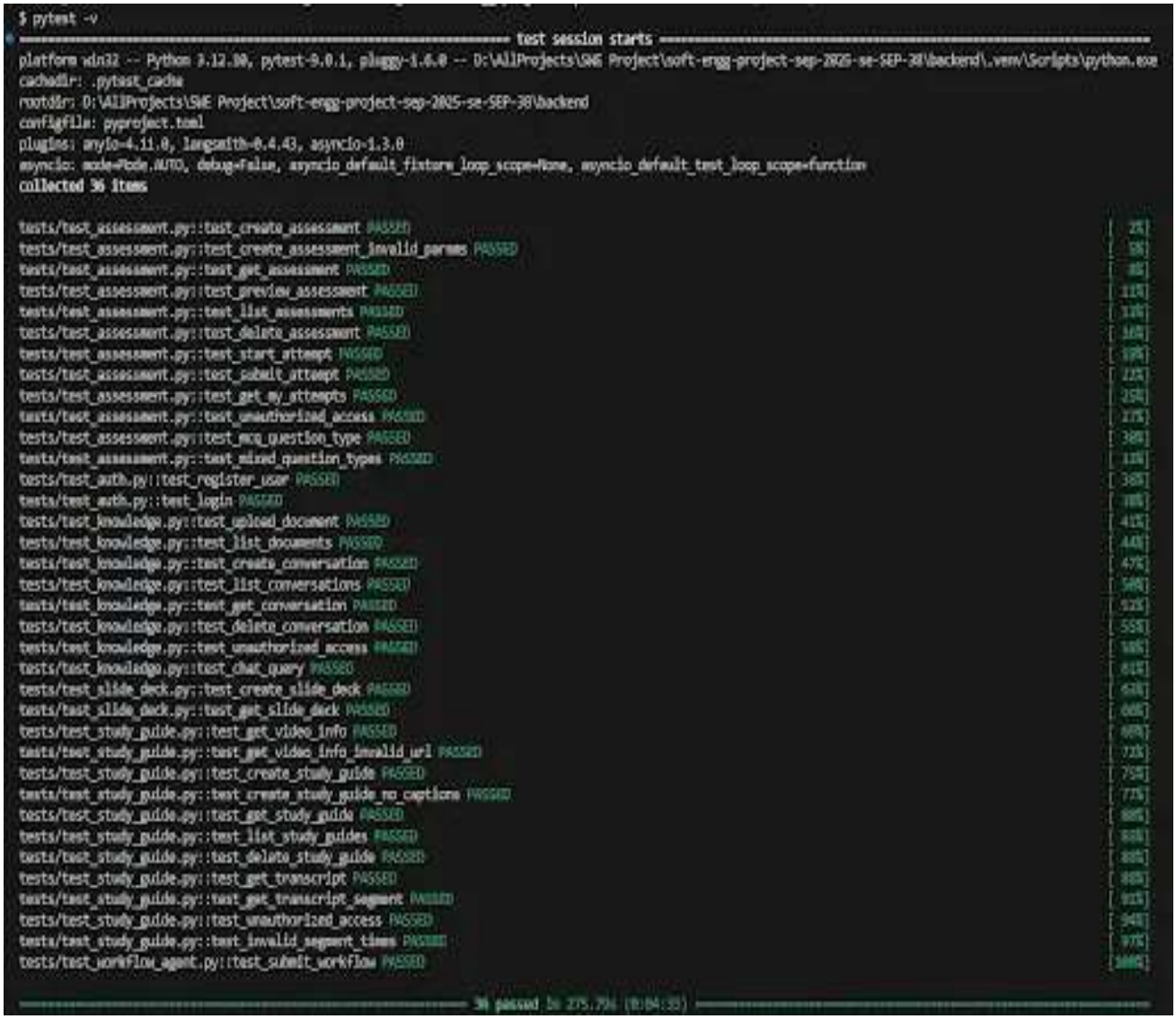
- Input
  {
  "course_id": 1,
  "query": "What is RAG?",
  "conversation_id": null
  }
- Expected Response
  Status: 200
  {
  "message": { "content": "AI answer" },
  "confidence_score": <0-1>,
  "should_escalate": true/false
  }

2. Conversation Not Found – FAILURE

- Input
  {
  "course_id": 1,
  "query": "Hello",
  "conversation_id": 9999
  }
- Expected Output
  Status: 404

{
"detail": "Conversation not found"
}

Sample Tests



Milestone - 6

Steps to run app | Tech Stack | Issues & Pull Requests

7. How to Run the Application

This section explains how to run both the backend (FastAPI + Neon Postgres + Gemini) and the frontend (Next.js 14) together for full EduAssist functionality.

Backend Setup (FastAPI)

Follow these steps inside the backend directory.

1. Create your environment file

- `cp .env.example .env`

2. Configure Environment Variables

Database (PostgreSQL – Neon)

Create a free database at:
https://neon.tech

Then set:

- `DATABASE_URL=postgresql://user:password@host/dbname`

Gemini API Key

- `GEMINI_API_KEY=your_key_here`

App Secret Key

Generate a secure key:

- `openssl rand -hex 32`

Then set:

- `SECRET_KEY=your_generated_key`

---

3. Ensure uv is installed

EduAssist uses Astral's ultra-fast Python runner (`uv`) instead of pip.

Check installation:

- `command -v uv`
- `uv self version`

If not installed, follow installation instructions:
https://docs.astral.sh/uv/getting-started/installation/

---

4. Start the Backend Server

Run:

- `source launch.sh`

This script will:

- Create a virtual environment (if missing)
- Sync dependencies (`uv sync`)
- Run the FastAPI server
- Auto-load `.env`

Backend will be available at:
http://localhost:8000

API docs:
http://localhost:8000/docs

Frontend Setup (Next.js 14)

1. Navigate to frontend

- `cd frontend`

2. Install dependencies

- `npm install`

3. Set environment variables

Create `.env.local`:

- `NEXT_PUBLIC_API_URL=http://localhost:8000`

4. Start the frontend

- `npm run dev`

Frontend will run on:
http://localhost:3000

End-to-End Usage

To use the full platform, ensure both backend and frontend are running.

Component	URL	Status
Backend API (FastAPI)	http://localhost:8000	Must be running
Swagger Docs	http://localhost:8000/docs	Optional
Frontend (Next.js)	http://localhost:3000	Must be running

After launching both:

- Login or Signup

- Use the Knowledge Assistant
- Upload documents (/knowledge/documents)
- Generate study guides, assessments, slide decks
- Submit feedback
- Use admin workflows

---

8. Tech Stack Used

Frontend

- Framework: Next.js 14 (App Router)
- Language: TypeScript
- Styling: Tailwind CSS
- Icons: Lucide React
- State: React Hooks
- Auth: JWT-based
- UI: Clean, modern, responsive layouts

Backend

- Framework: FastAPI (Python 3.10+)
- Runner: Astral uv
- Database: PostgreSQL (Neon serverless)
- ORM: SQLAlchemy
- Authentication: JWT with SECRET_KEY
- AI Integration: Gemini API + OpenAI-compatible LLMs
- API Specs: OpenAPI 3.0 auto-generated docs

- Routing: Modular routers (Auth, Knowledge, Study Guides, Assessments, Slides, Feedback)

AI / LLM Services

Used for:

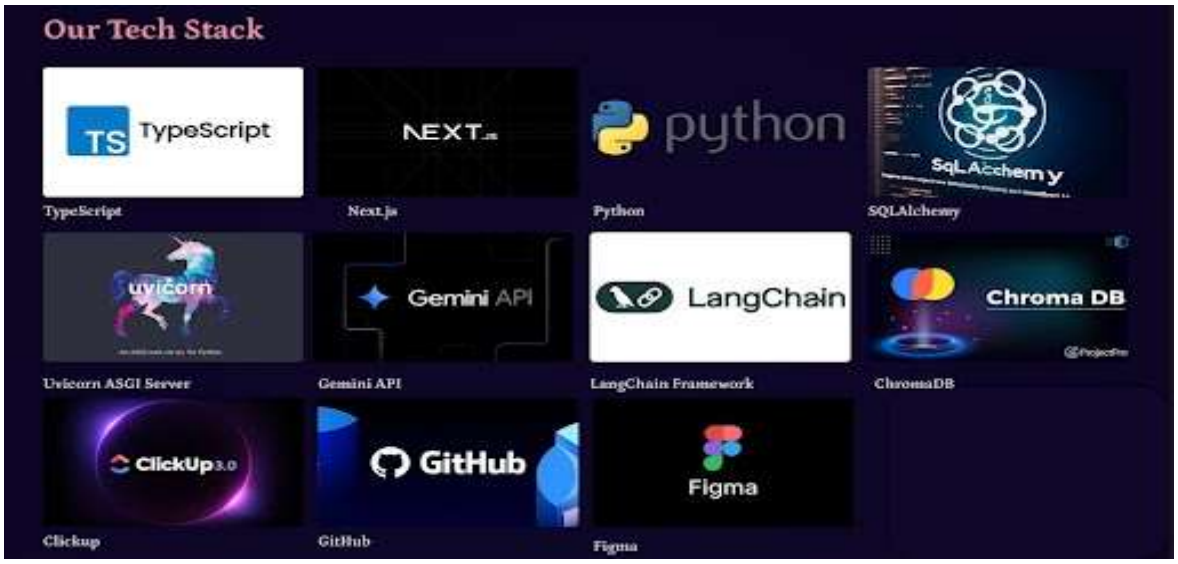
- Knowledge Assistant (RAG + LLM)
- Study guide generation
- Assessment creation
- Slide deck generation
- Content priority tagging
- Feedback insights

Security

- JWT-based session authentication
- Role-based access (Student, Instructor, Admin)
- Environment-variable-driven secrecy
- CORS configuration
- Endpoint validation and error handlers

Dev / Tooling

- uv (dependency and environment management)
- Uvicorn (ASGI server)
- Pytest (testing)
- Swagger UI (API documentation)
- Pydantic (schemas)



---

9. Issue Reporting & Pull Requests

This section summarizes the current issues and pull requests visible in the repository, based on the latest activity shown in the project dashboards.

Issues Overview

The following issues are currently visible or have recently been completed:

Backend Enhancements & API Completion

Several issues are focused on completing or refining backend API implementations. Examples include completing existing endpoints and improving API consistency.

Frontend + Backend Integration

Issues referencing UI–backend alignment, API consumption fixes, and improved communication between the two layers.

Feature Pages

Tasks related to creating new feature pages, including layout, navigation support, and adding functional components.

Deployment and CI/CD

Issues dedicated to setting up or improving deployment pipelines, ensuring smooth builds and automated workflows.

Form and Input Components

Work items relating to the creation of reusable frontend forms and input handling components.

Navigation & Layout Structure

Multiple tasks related to updating or reorganizing the UI layout, sidebar, and top-level navigation to improve UX.

Authentication & Login Pages

Dedicated issues around improving login pages, refining logic, and enhancing Tailwind styling for authentication flows.

Status

From the repository view, the project is in **active development**.
Issues are being regularly updated, completed, and closed.
Recent merges include both backend and frontend contributions.

Pull Request Summary

The following PRs reflect the recent development progress:

1. Documentation Updates

Pull requests adding or updating the project README, improving clarity and details about EduAssist.

2. Backend Improvements

PRs adding or refining backend features such as:

- Updated FastAPI routes
- Enhanced API functionality
- Backend structural improvements

3. Frontend Page Additions

Pull requests adding new pages such as:

- Auth-related pages

- Feature pages
- Enhanced layouts

4. Main Layout Enhancements

Several PRs introduce or refine the main layout, navigation, and standardized page components.

5. Styling & Tailwind Integration

Pull requests focused on improving styling:

- Tailwind-based UI upgrades
- Cleaner page templates
- UI consistency improvements

6. Fixes and Maintenance

PRs addressing minor fixes, refactoring, and structural improvements to both backend and frontend codebases.

7. UI/UX Polishing

Pull requests introducing polished UI elements, improving navigation flow, and enhancing responsiveness.

9. Issue Reporting & Pull Requests

This section summarizes the current issues and pull requests visible in the repository, based on the latest activity shown in the project dashboards.

Issues Overview

The following issues are currently visible or have recently been completed:

Backend Enhancements & API Completion

Several issues are focused on completing or refining backend API implementations. Examples include completing existing endpoints and improving API consistency.

Frontend + Backend Integration

Issues referencing UI–backend alignment, API consumption fixes, and improved communication between the two layers.

Feature Pages

Tasks related to creating new feature pages, including layout, navigation support, and adding functional components.

Deployment and CI/CD

Issues dedicated to setting up or improving deployment pipelines, ensuring smooth builds and automated workflows.

Form and Input Components

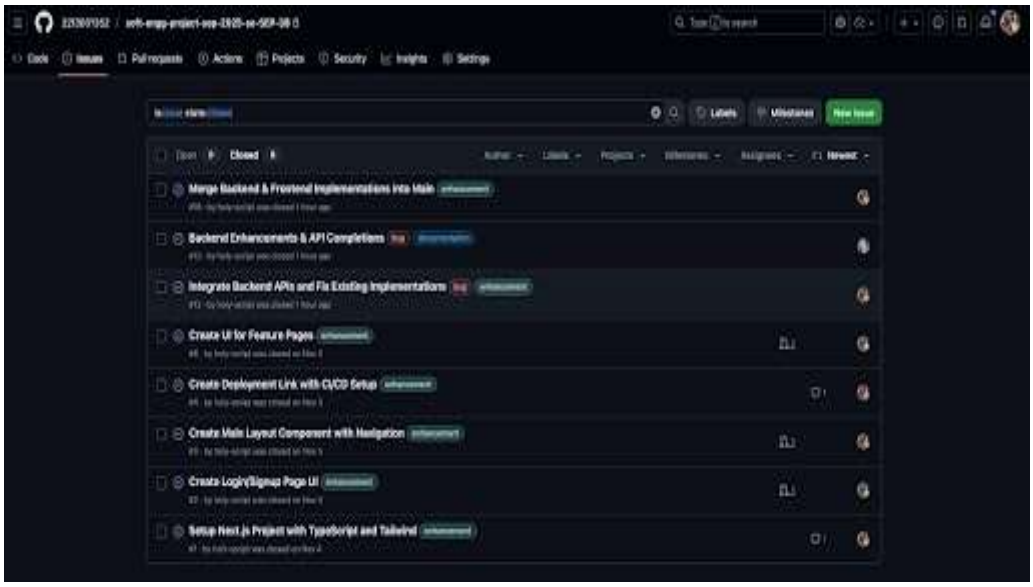
Work items relating to the creation of reusable frontend forms and input handling components.

Navigation & Layout Structure

Multiple tasks related to updating or reorganizing the UI layout, sidebar, and top-level navigation to improve UX.

Authentication & Login Pages

Dedicated issues around improving login pages, refining logic, and enhancing Tailwind styling for authentication flows.



Status

From the repository view, the project is in **active development**.
Issues are being regularly updated, completed, and closed.

Recent merges include both backend and frontend contributions.

Pull Request Summary

The following PRs reflect the recent development progress:

1. Documentation Updates

Pull requests adding or updating the project README, improving clarity and details about EduAssist.

2. Backend Improvements

PRs adding or refining backend features such as:

- Updated FastAPI routes
- Enhanced API functionality
- Backend structural improvements

3. Frontend Page Additions

Pull requests adding new pages such as:

- Auth-related pages
- Feature pages
- Enhanced layouts

4. Main Layout Enhancements

Several PRs introduce or refine the main layout, navigation, and standardized page components.

5. Styling & Tailwind Integration

Pull requests focused on improving styling:

- Tailwind-based UI upgrades
- Cleaner page templates
- UI consistency improvements

6. Fixes and Maintenance

PRs addressing minor fixes, refactoring, and structural improvements to both backend and frontend codebases.

7. UI/UX Polishing

Pull requests introducing polished UI elements, improving navigation flow, and enhancing responsiveness.