CHAPTER 1

INTRODUCTION

1.1 Project Summary
1.2 Purpose & Goals
1.3 Technology

1.1 Project Summary

Syllaby is a full-stack, AI-driven web application designed to serve as a comprehensive study plannerfor students and tutors. Its primary function is to take a user-provided course outline—in any text format—and use a Large Language Model (LLM) to generate a detailed, week-by-week study plan.
Beyond simple text generation, Syllaby integrates this plan into a suite of productivity tools. Each generated plan is accompanied by an interactive Kanban board for task management, and the system offers further AI-powered tools such as a quiz generator, flashcard creator, and an educational chatbot. The application is built on a modern technology stack, ensuring a responsive, secure, and scalable user experience.

1.2 Project Summary:

The primary purpose of Syllaby is to reduce the cognitive load of academic planning and empower students with tools that promote organized and efficient learning habits.
The key goals of the project are:

To Automate Study Planning: To eliminate the manual effort required to create a structured study schedule from a course syllabus.

To Enhance Organization: To provide users with an integrated Kanban board system for visual task management, including features like setting deadlines and tracking completion.

To Provide Personalized Learning Tools: To leverage AI to create study aids like quizzes, flashcards, and summaries tailored to the user's specific course content.

To Centralize Academic Management: To offer a single, unified platform where a user can manage study plans, tasks, notes, and self-assessment for all their courses.

To Create a Modern User Experience: To build a fast, intuitive, and responsive web application using modern technologies like React and FastAPI.

1.3 Technology

Python(FastAPI):

A modern, high-performance Python web framework used to build the application's REST API. It was chosen for its incredible speed, automatic generation of interactive API documentation (via OpenAPI and Swagger UI), and robust data validation powered by Pydantic.

REACT:

A popular JavaScript library for building dynamic user interfaces. Its component-based architecture was ideal for creating a modular and maintainable Single Page Application (SPA), enabling features like drag-and-drop Kanban boards and interactive modals.

PostgreSQL:

A powerful, open-source object-relational database system known for its reliability, feature robustness, and performance. It is used to store all persistent data, including user accounts, syllabi, Kanban boards, and tasks.

Artificial Intelligence & Machine Learning (Core):

The core intelligence of Syllaby is powered by an LLM accessed via the Ollama framework. This allows for local, private, and powerful natural language processing to generate study plans, extract tasks, create quizzes, and power the chatbot.

CHAPTER 2

PROJECT PROFILE

2.1 Project Planning and Scheduling
2.2 Risk Management
2.3 Schedule Representation

2.1 Project Planning and Scheduling

The development of Syllaby followed an **Agile development methodology**, specifically inspired by the Scrum framework. This approach was chosen for its flexibility, allowing for iterative development and the ability to adapt to new requirements and technical challenges as they arose.
The project was broken down into the following key phases:

Phase 1: Foundation & Authentication (Week 1-2):

Setup of the project environment (FastAPI, React, PostgreSQL).
Database schema design and implementation.
Implementation of user registration, login, and JWT-based authentication.

Phase 2: Core AI Feature - Syllaby Generation (Week 3-4):

Integration with the Ollama LLM.
Development of the FastAPI endpoint for syllabus creation.
Creation of the React form for users to input their course outline.

Phase 3: Kanban Board Implementation (Week 5-7):

Backend development for creating and managing boards, columns, and tasks.
Frontend implementation of the drag-and-drop interface using react-beautiful-dnd.
Integration of AI-powered task extraction during syllabus generation.

Phase 4: Advanced Features & UI Polish (Week 8-10):

Development of the interactive task detail modal.
Implementation of the "Upcoming Deadlines" dashboard widget and the Calendar view.
Development of supplementary AI tools (Quiz Generator, Chatbot).
Overall UI/UX refinement and styling.

Phase 5: Testing (Week 11-12):

End-to-end testing of all features.
Bug fixing and performance optimization.
Preparation of documentation and final report.

2.2 Risk Management

2.2.1 Risk Identification:

During the analysis and development of Syllaby, the following risks were identified:

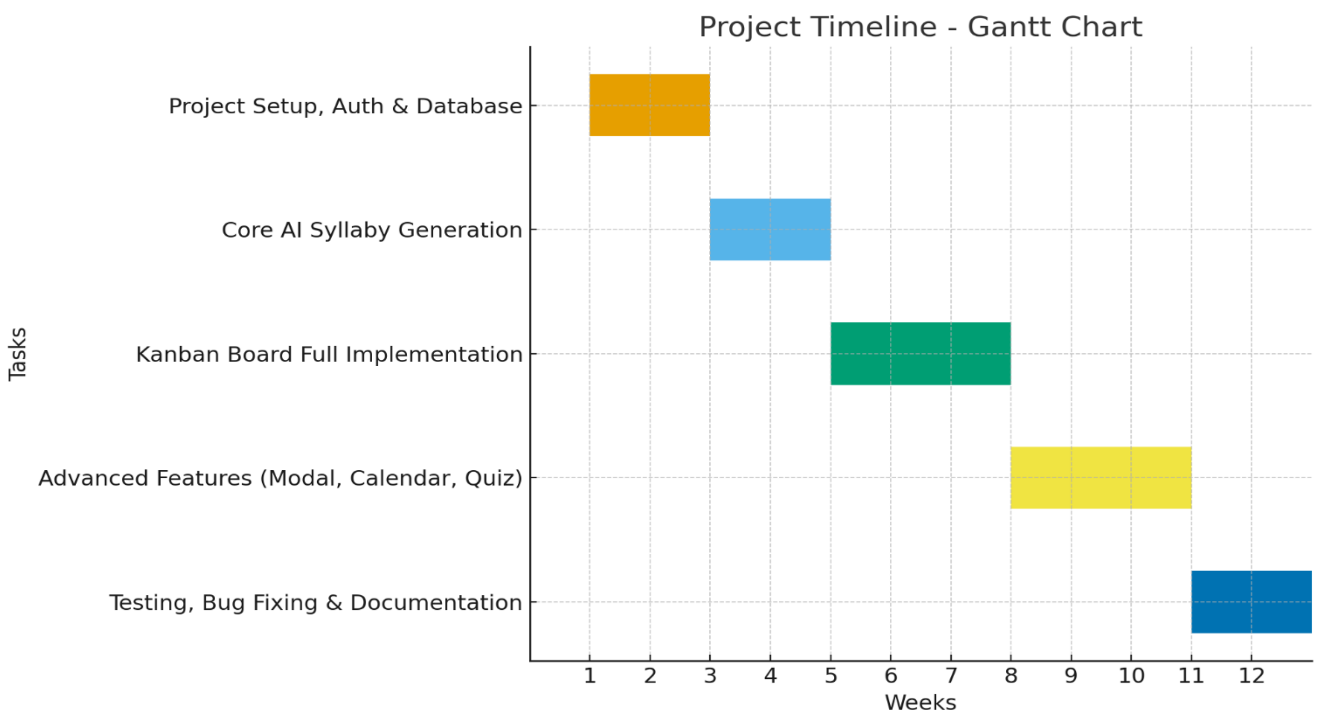
Technological Risk
Economical Risk
User Adoption Risk

2.2.2 Risk Analysis:

Several potential risks were identified at the outset of the project, with mitigation strategies planned accordingly.

Risk Category	Risk Description	Mitigation Strategy
Technical	The LLM may produce inconsistent or poorly formatted JSON, causing backend errors.	Implemented robust error handling (try-except blocks) and data validation (Pydantic schemas) to parse AI output safely. Refined AI prompts for stricter output formatting.
Technical	Integrating the frontend drag-and-drop library with React proves complex.	Researched and chose a well-documented library. Allocated specific time for prototyping and debugging this feature.
Schedule	The scope of AI features is larger than anticipated, causing delays.	Adopted an Agile approach, prioritizing the core syllabus generation and Kanban features first, with other tools as "stretch goals."
Usability	The user interface is not intuitive, leading to poor user adoption.	Focused on a clean, minimal design. Conducted informal usability tests with peers to gather feedback and refine the user flow.

Schedule Representation



CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 User Characteristics

3.2 Software and Hardware Requirement

3.3 Constraint

3.1 User Characteristics

Syllaby is designed for two primary types of users:

Students: The primary audience. They are typically tech-savvy individuals enrolled in high school, college, or university. Their main goal is to improve their personal organization, manage multiple courses simultaneously, and find more efficient ways to study. They require an intuitive interface that is quick to learn and provides immediate value.

Tutors/Educators: A secondary audience. They can use Syllaby to quickly generate structured learning plans for their students. They require a tool that is reliable and produces clear, actionable outputs that can be easily shared or referenced.

3.2 Hardware and Software Requirements

Hardware:

A server (cloud or physical) with a multi-core CPU.
At least 8GB of RAM (16GB+ recommended for running LLMs).
Sufficient disk space for the OS, database, and application code.
A GPU is highly recommended for running the Ollama LLM efficiently.

Software:

A Linux-based operating system (e.g., Ubuntu 20.04+).
Python 3.10 or newer.
PostgreSQL Server 13+.
Node.js 16+ (for building the React frontend).
Ollama framework installed and running with a suitable LLM (e.g., Llama 3, Mistral).

3.3 Constraint

Dependency on Ollama: The application's core AI features are entirely dependent on a running instance of the Ollama server. If the Ollama service is down, these features will be unavailable.

Quality of AI Output: The quality and accuracy of the generated study plans and extracted tasks are directly tied to the capability of the underlying LLM.

Stateless Authentication: The system uses JWT for authentication, which is stateless. This means the server does not store session information, requiring the token to be sent with every secure API request.

No Real-Time Collaboration: The application is designed as a single-user tool. It does not currently support real-time collaboration on Kanban boards or study plans.

CHAPTER 4

ANALYSIS AND DESIGN

4.1 System Requirements

4.2 Feasibility Analysis

4.3 Requirement Validation

4.4 Requirement Gathering

4.5 Process Model

4.6 System Design

4.1 System Requirements:

Functional Requirements:

1.User Management:
The system shall allow new users to register for an account.
The system shall allow existing users to log in securely.
The system shall protect user-specific data, ensuring a user can only access their own content.

2.Syllabus Management:
Users must be able to create a new study plan by submitting a title and a raw course outline.
The system shall use an AI model to process the raw outline and generate a structured study plan.
Users must be able to view a list of all their created syllabi.
Users must be able to view, edit, and delete a specific syllabus.

3.Kanban Task Management:
Upon syllabus creation, the system shall automatically generate a corresponding Kanban board.
The system shall use an AI model to extract tasks from the generated plan and populate the "To Do" column.
Users must be able to create, view, edit, and delete tasks within a Kanban board.
Users must be able to drag and drop tasks between columns to update their status.
Tasks must support details such as a description, priority level, due date, and a completion status.

4.Dashboard & Calendar:
The user's homepage shall display a dashboard widget showing upcoming task deadlines.
The system shall provide a full calendar view displaying all tasks with set due dates.

Non-Functional Requirements:

Performance: The application's API should respond to most requests in under 500ms. The frontend should load initial content in under 3 seconds on a standard internet connection.

Security: User authentication must be secure, using hashed passwords and JSON Web Tokens (JWT). All user data must be segregated and accessible only by the owner.

Usability: The user interface must be intuitive, responsive, and accessible on both desktop and mobile web browsers.

Scalability: The backend architecture should be able to handle an increasing number of users and data without significant degradation in performance.

Reliability: The application should handle errors gracefully (e.g., API failures, AI model unavailability) and provide clear feedback to the user.

4.2 Feasibility Analysis:

Technical Feasibility: The chosen technology stack (FastAPI, React, PostgreSQL, Ollama) is mature, well-documented, and widely used, making the project technically feasible. The primary technical challenge was the integration with the Ollama LLM and ensuring its output could be reliably parsed, which was addressed through careful prompt engineering and robust backend validation.

Economic Feasibility: The project was developed using open-source software (Python, React, PostgreSQL, Ollama), resulting in zero software licensing costs. The primary cost would be for deployment and hosting, which can be managed affordably using cloud services. The development effort was deemed manageable for a single developer within the project timeline.

Operational Feasibility: The application is designed to be a standard web service. Once deployed, it requires minimal operational maintenance beyond standard server and database management. The end-user requires no special training, as the interface is designed to be intuitive and similar to other modern productivity applications.

4.3 Requirement Validation:

Syllaby is a **dynamic AI-powered study planner**, and the following validation rules ensure correct and secure operation:

User Access Rights:

Students can **view, create, and manage their own study schedules**.
Users can **interact with AI features** such as flashcards, notes, and chatbot tutoring.
Students **cannot modify system logic, AI recommendations, or other users' data**.

System Integrity:

All updates to schedules, AI recommendations, and performance tracking are **handled automatically by the backend**.
The system ensures **data consistency, accuracy, and security** across all modules.

4.4 Requirement Gathering:

Requirement gathering is the process of **collecting, analysing, documenting, and validating the system requirements** to ensure the developed application meets user needs.

The main steps involved in requirement engineering for Syllaby are:
Finding out: Identifying the features and functionalities students need in a study planner.
Analysing: Determining which features are feasible and how AI can assist in automating study schedules.
Documenting: Recording all functional and non-functional requirements clearly.
Checking: Ensuring that the gathered requirements are complete, consistent, and achievable.

Key Requirements for Syllaby:
The system must **dynamically generate and update study schedules** based on user input and syllabus deadlines.

The platform must provide **search and filter functionality** for notes, flashcards, and study materials.
It must offer **AI-powered guidance**, including study plan recommendations, reminders, and progress tracking.
The system should have a **user-friendly interface** accessible via modern web browsers on desktops and mobile devices.

4.5 Process Model:

The **Agile (Iterative) Process Model** was selected for the development of Syllaby. This methodology was chosen for its flexibility in handling evolving requirements, especially concerning the experimental nature of AI integration. It allowed for the project to be built in small, manageable, and testable increments.

The development was broken down into sprints:
Sprint 1: Foundation & Authentication: Setup project structure, database schema, and user login/registration.

Sprint 2: Core Feature: Implemented AI-powered syllabus generation and automatic Kanban board creation.

Sprint 3: Task Management: Developed the drag-and-drop Kanban UI, task creation, and task detail modals.

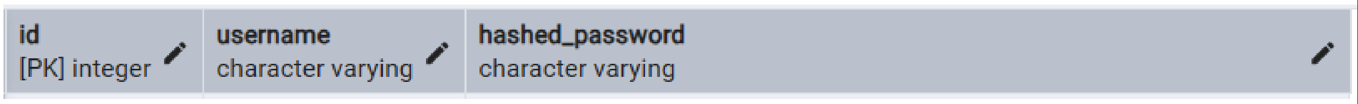
Sprint 4: Dashboarding & Visualization: Built the "Upcoming Deadlines" widget and the full Calendar page.

Sprint 5: Testing & Refinement: Conducted end-to-end testing, fixed bugs, and prepared documentation.

4.6 System Design:

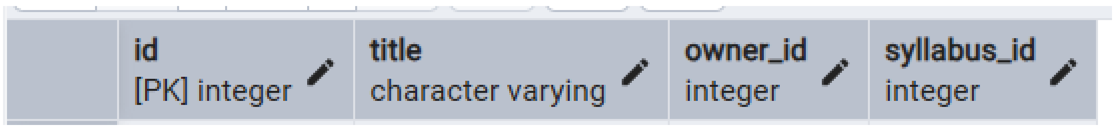
4.6.1 Data Dictionary

This describes the main tables in the PostgreSQL database.
Table: users



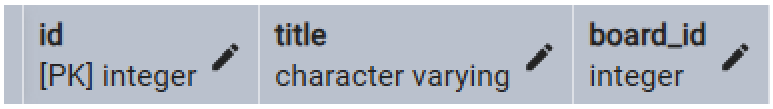
id	username	hashed_password
[PK] integer	character varying	character varying

Table: kanban_boards



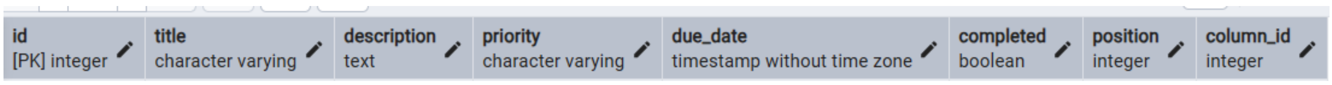
	id	title	owner_id	syllabus_id
	[PK] integer	character varying	integer	integer

Table: kanban_columns



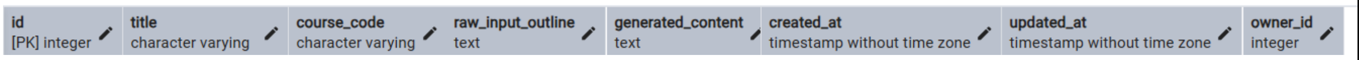
id	title	board_id
[PK] integer	character varying	integer

Table: kanban_tasks



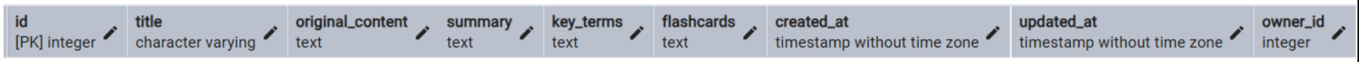
id	title	description	priority	due_date	completed	position	column_id
[PK] integer	character varying	text	character varying	timestamp without time zone	boolean	integer	integer

Table: syllaby



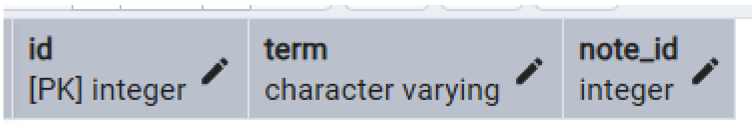
id	title	course_code	raw_input_outline	generated_content	created_at	updated_at	owner_id
[PK] integer	character varying	character varying	text	text	timestamp without time zone	timestamp without time zone	integer

Table: notes



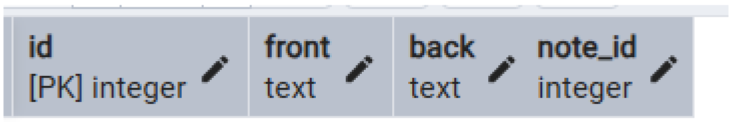
id	title	original_content	summary	key_terms	flashcards	created_at	updated_at	owner_id
[PK] integer	character varying	text	text	text	text	timestamp without time zone	timestamp without time zone	integer

Table: key_term



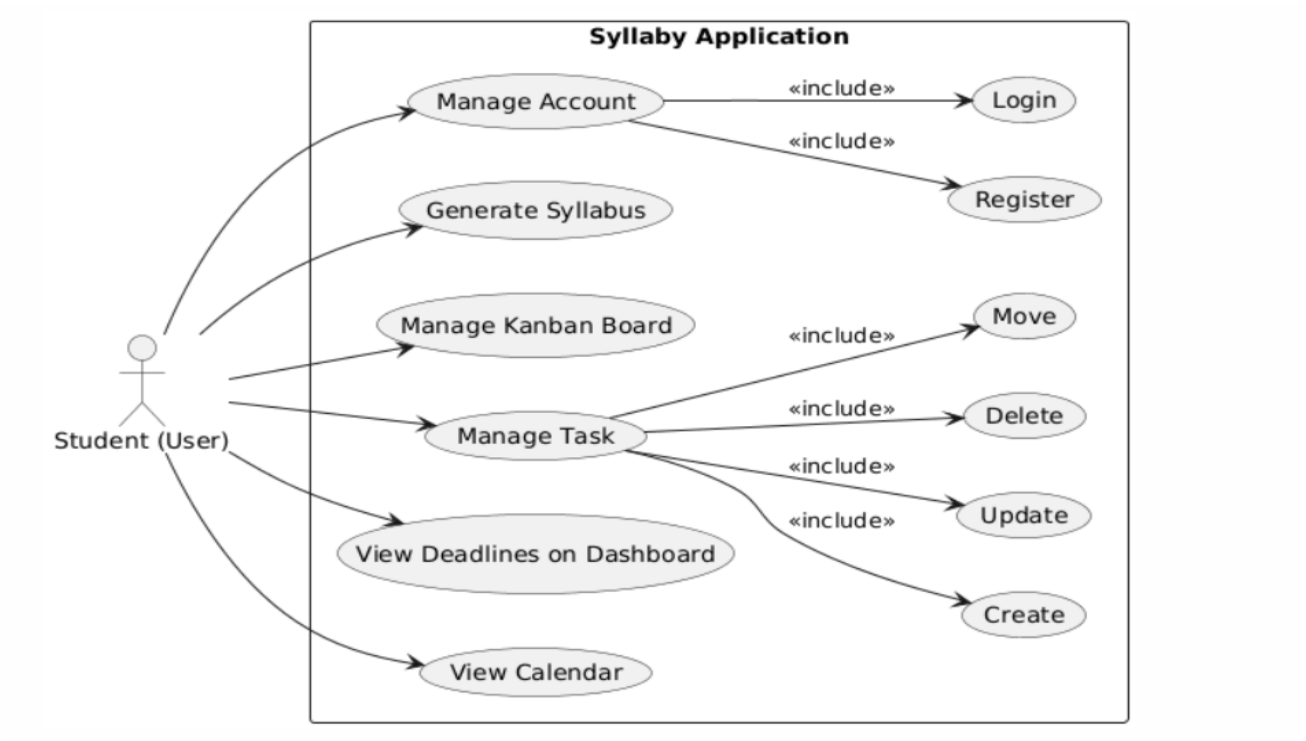
id	term	note_id
[PK] integer	character varying	integer

Table: flashcard

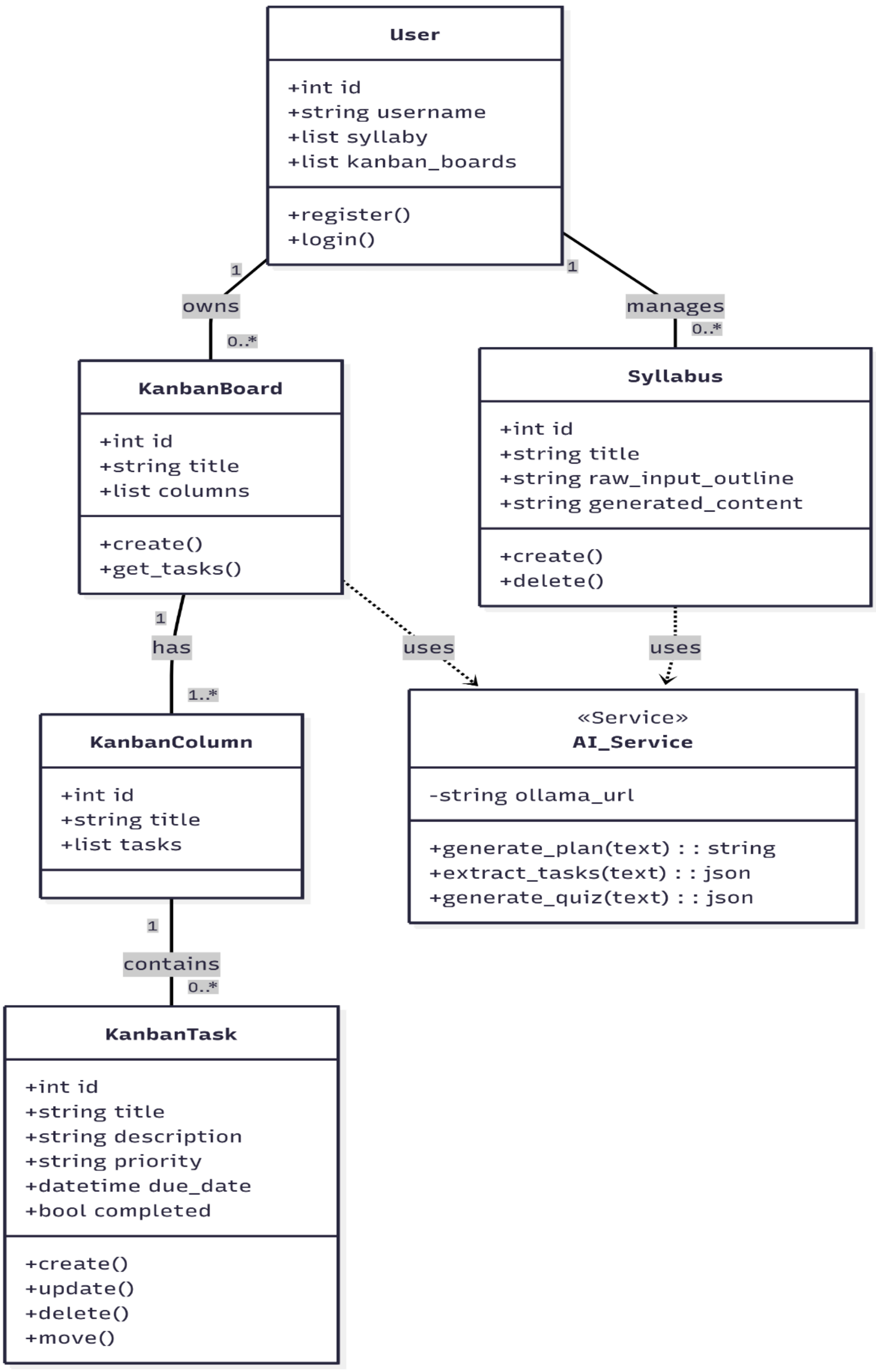


id	front	back	note_id
[PK] integer	text	text	integer

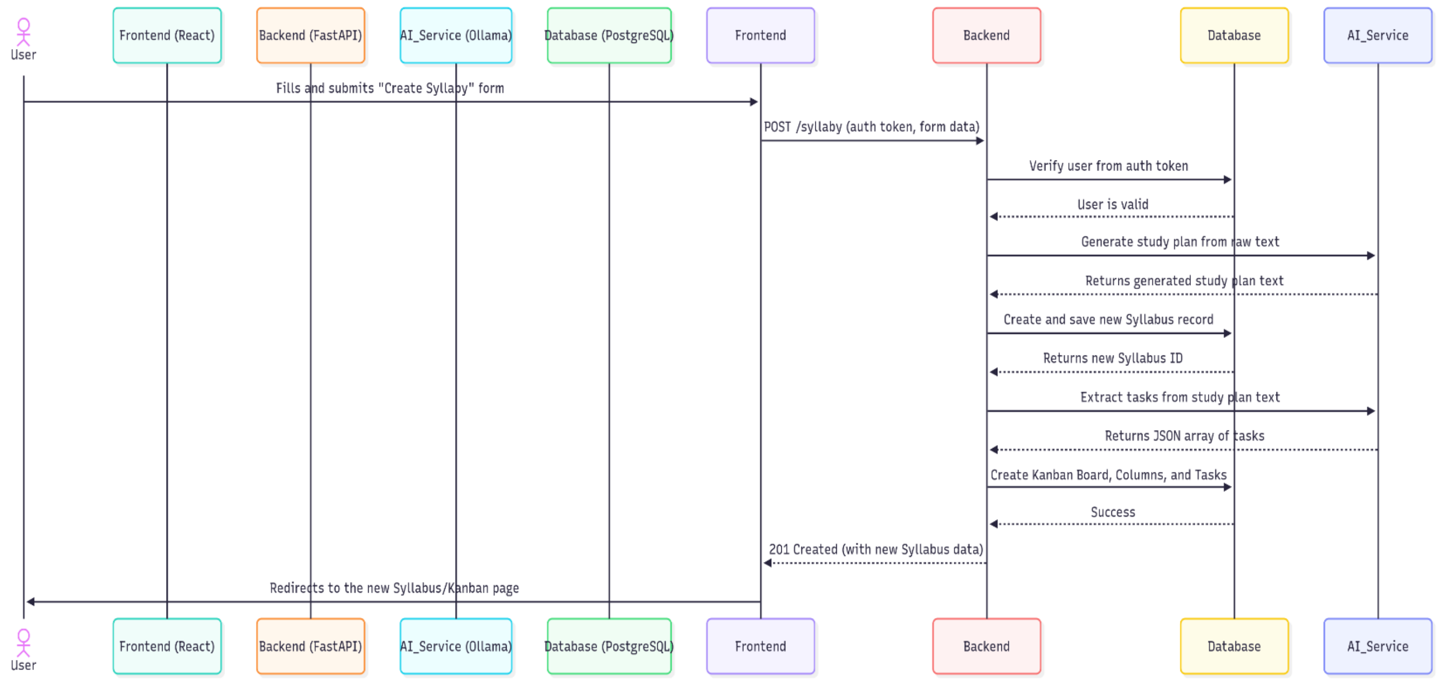
4.6.2 Use case Diagram:



4.6.3 Class Diagram:

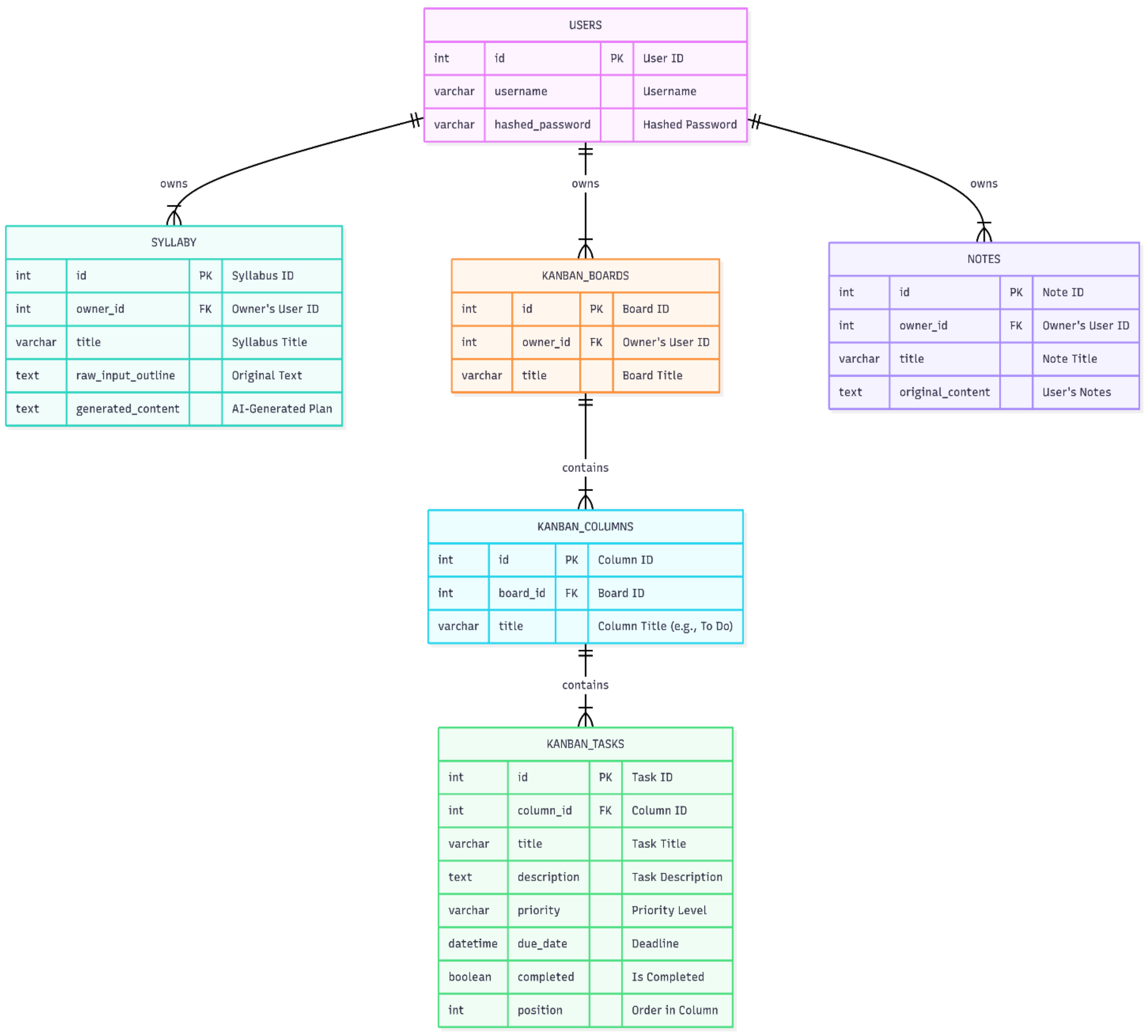


4.6.4 Sequence Diagram:


User
Frontend (React)
Backend (FastAPI)
AI_Service (Ollama)
Database (PostgreSQL)
Frontend
Backend
Database
AI_Service
Fills and submits "Create Syllaby" form
POST /syllaby (auth token, form data)
Verify user from auth token
User is valid
Generate study plan from raw text
Returns generated study plan text
Create and save new Syllabus record
Returns new Syllabus ID
Extract tasks from study plan text
Returns JSON array of tasks
Create Kanban Board, Columns, and Tasks
Success
201 Created (with new Syllabus data)
Redirects to the new Syllabus/Kanban page

4.6.5 E-R Diagram:

User E-R Diagram



USERS

int	id	PK	User ID
varchar	username		Username
varchar	hashed_password		Hashed Password

SYLLABY

int	id	PK	Syllabus ID
int	owner_id	FK	Owner's User ID
varchar	title		Syllabus Title
text	raw_input_outline		Original Text
text	generated_content		AI-Generated Plan

KANBAN_BOARDS

int	id	PK	Board ID
int	owner_id	FK	Owner's User ID
varchar	title		Board Title

NOTES

int	id	PK	Note ID
int	owner_id	FK	Owner's User ID
varchar	title		Note Title
text	original_content		User's Notes

KANBAN_COLUMNS

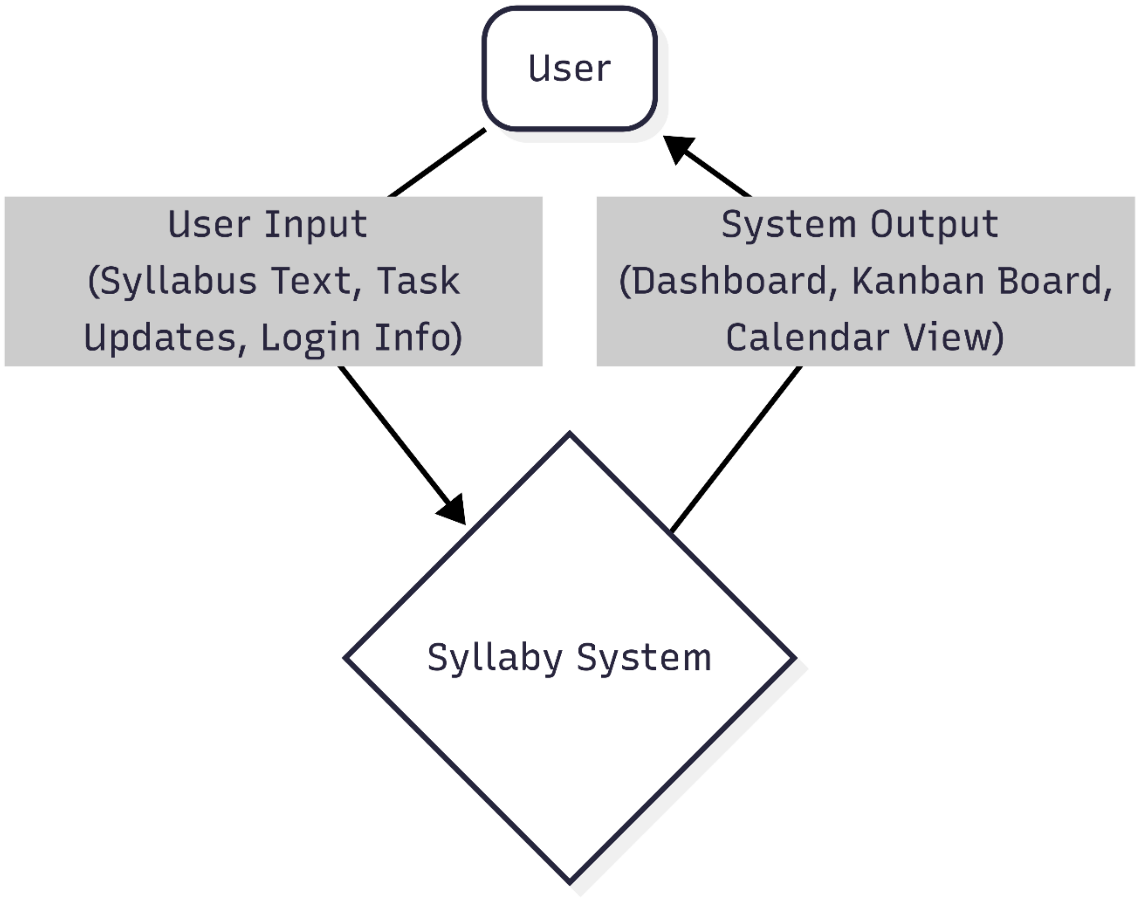
int	id	PK	Column ID
int	board_id	FK	Board ID
varchar	title		Column Title (e.g., To Do)

KANBAN_TASKS

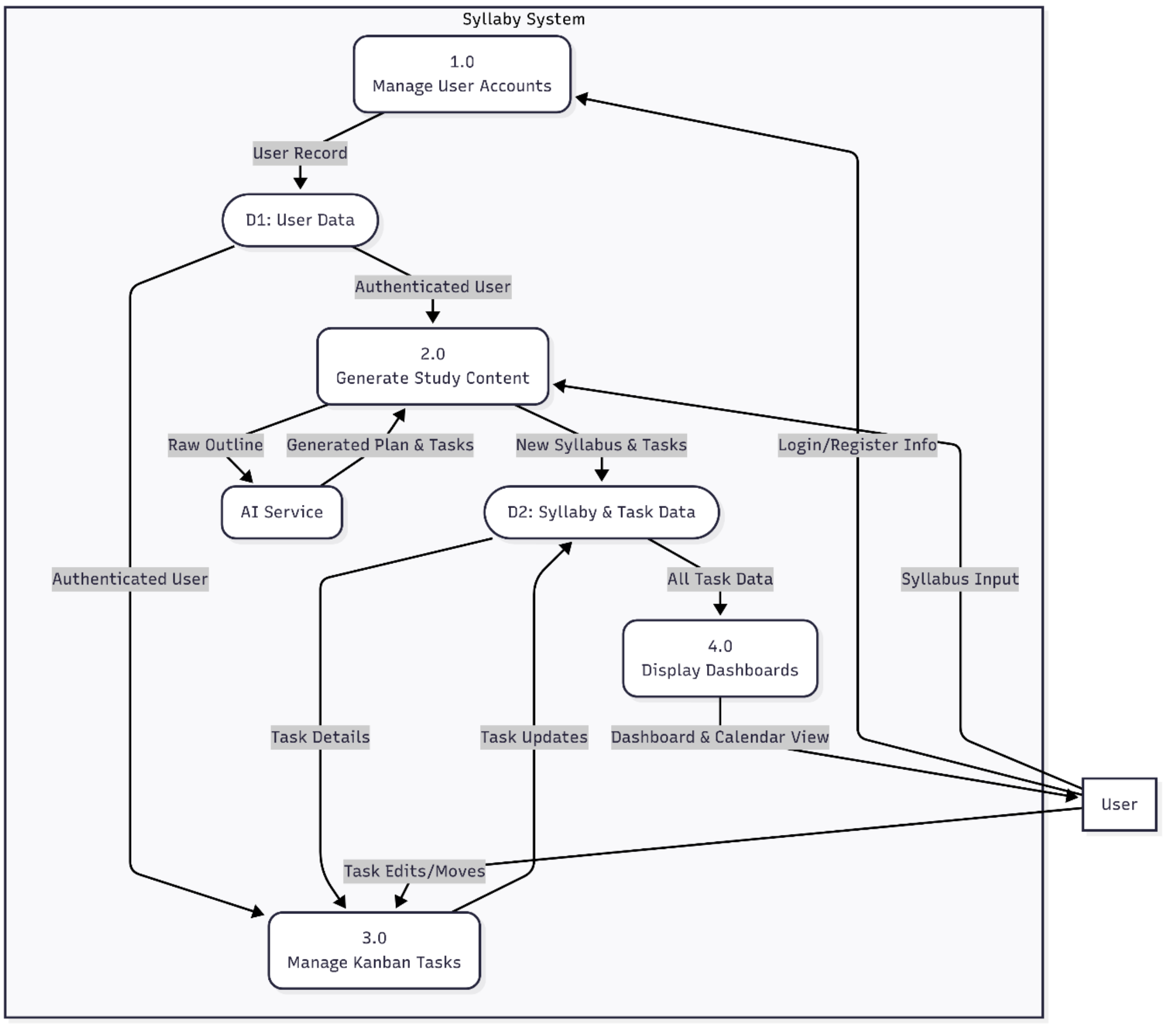
int	id	PK	Task ID
int	column_id	FK	Column ID
varchar	title		Task Title
text	description		Task Description
varchar	priority		Priority Level
datetime	due_date		Deadline
boolean	completed		Is Completed
int	position		Order in Column

4.6.6 Data Flow Diagram:

DFD 0 LEVEL


User
User Input (Syllabus Text, Task Updates, Login Info)
System Output (Dashboard, Kanban Board, Calendar View)
Syllaby System

DFD 1 LEVEL


Syllaby System
1.0
Manage User Accounts
User Record
D1: User Data
Authenticated User
2.0
Generate Study Content
Raw Outline
Generated Plan & Tasks
New Syllabus & Tasks
Login/Register Info
AI Service
D2: Syllaby & Task Data
Authenticated User
All Task Data
Syllabus Input
4.0
Display Dashboards
Task Details
Task Updates
Dashboard & Calendar View
User
Task Edits/Moves
3.0
Manage Kanban Tasks

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 Implementation Environment
5.2 User Side Screen Shot
5.3 Admin Side Screen Shot
5.4 Sample Coding

5.1 Implementation Environment

Multiuser & GUI Environment:

Syllaby is a **multiuser system**, designed as a web application accessible from any modern browser on any device with an internet connection. Each user's data is securely isolated.
The application provides a fully **Graphical User Interface (GUI)** built with React. This component-based architecture ensures a dynamic, responsive, and intuitive user experience, which is crucial for a productivity tool.

Technology Stack:

Backend: Python 3.11, FastAPI, SQLAlchemy (ORM), Pydantic
Frontend: React 18, JavaScript (ES6+), HTML5, CSS3, Tailwind CSS
Database: PostgreSQL 14
AI Service: Ollama framework with Llama 3 8B model
Key Libraries: Axios, React Router, @hello-pangea/dnd, react-big-calendar, react-hot-toast

5.2 User Side Screen Shot

User Module:

The primary users of the system are students and educators.
Users can register, log in, and manage their own study plans, notes, and tasks.
All user data is protected and accessible only after successful authentication. Syllaby does not have a separate "Admin" module for public users; system administration is handled at the database and server level.

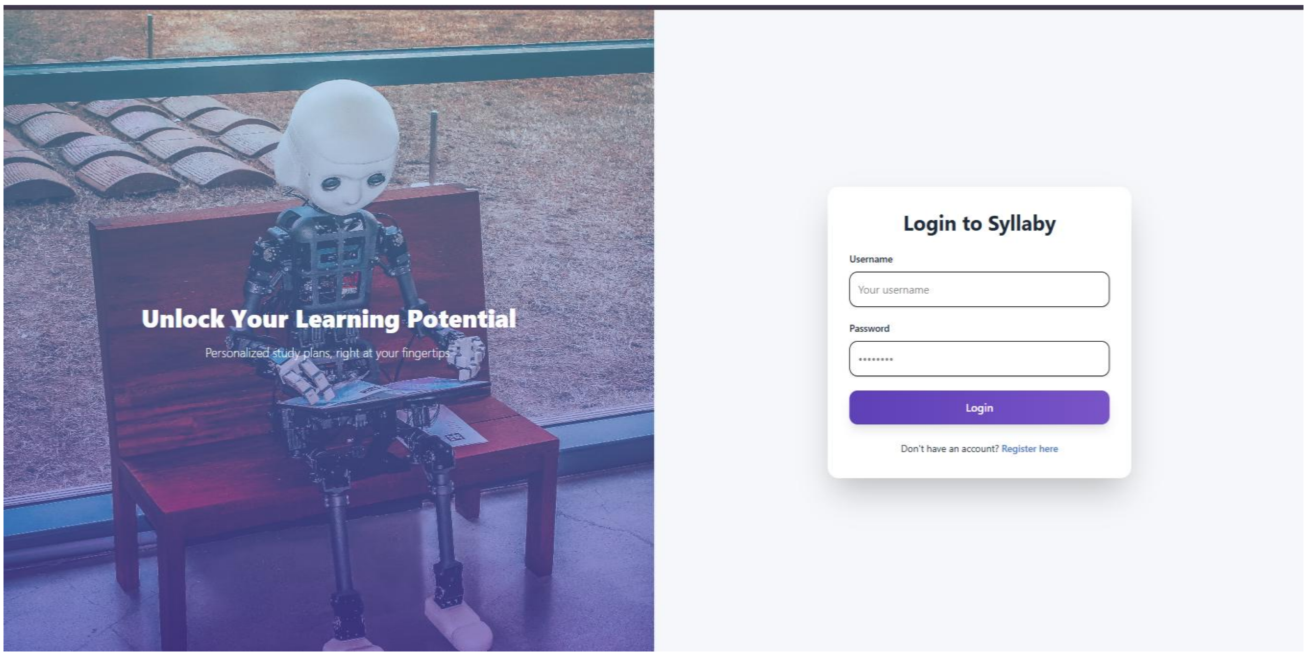
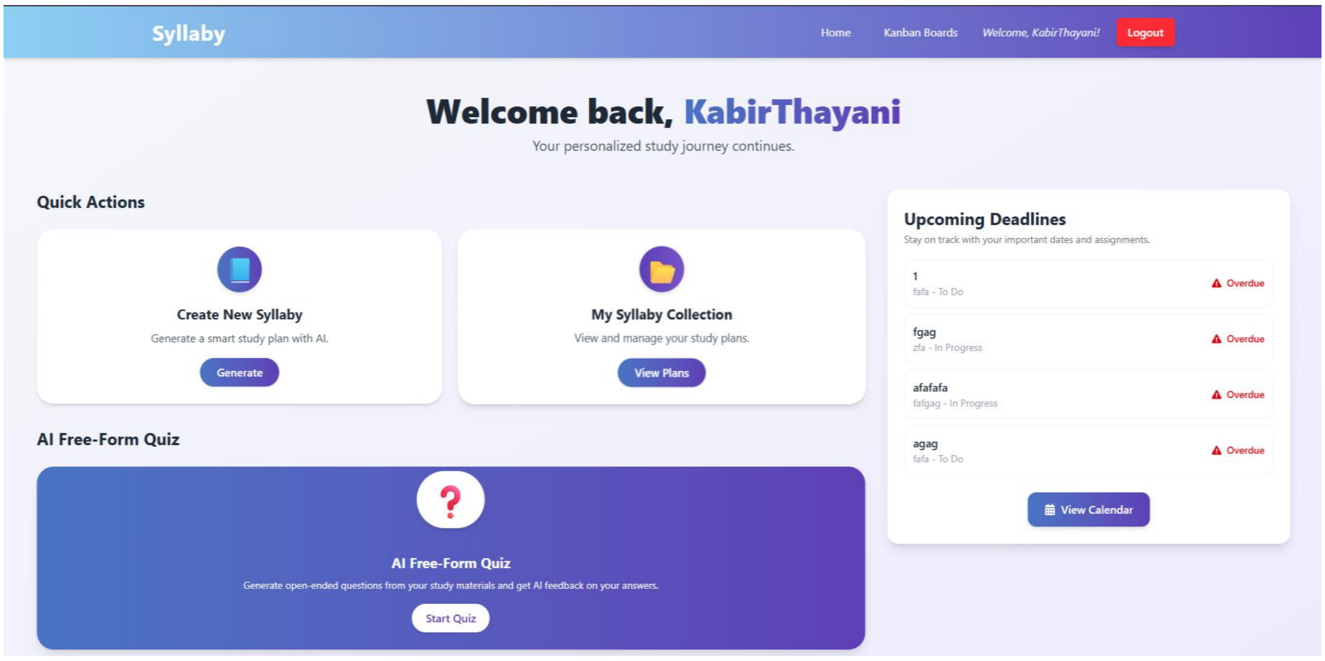

Unlock Your Learning Potential
Personalized study plans, right at your fingertips
Login to Syllaby
Username
Your username
Password
••••••••
Login
Don't have an account? Register here

Figure 5.1 - User Login Page



Welcome back, KabirThayani

Your personalized study journey continues.

Quick Actions

Create New Syllaby

Generate a smart study plan with AI.

My Syllaby Collection

View and manage your study plans.

AI Free-Form Quiz

AI Free-Form Quiz

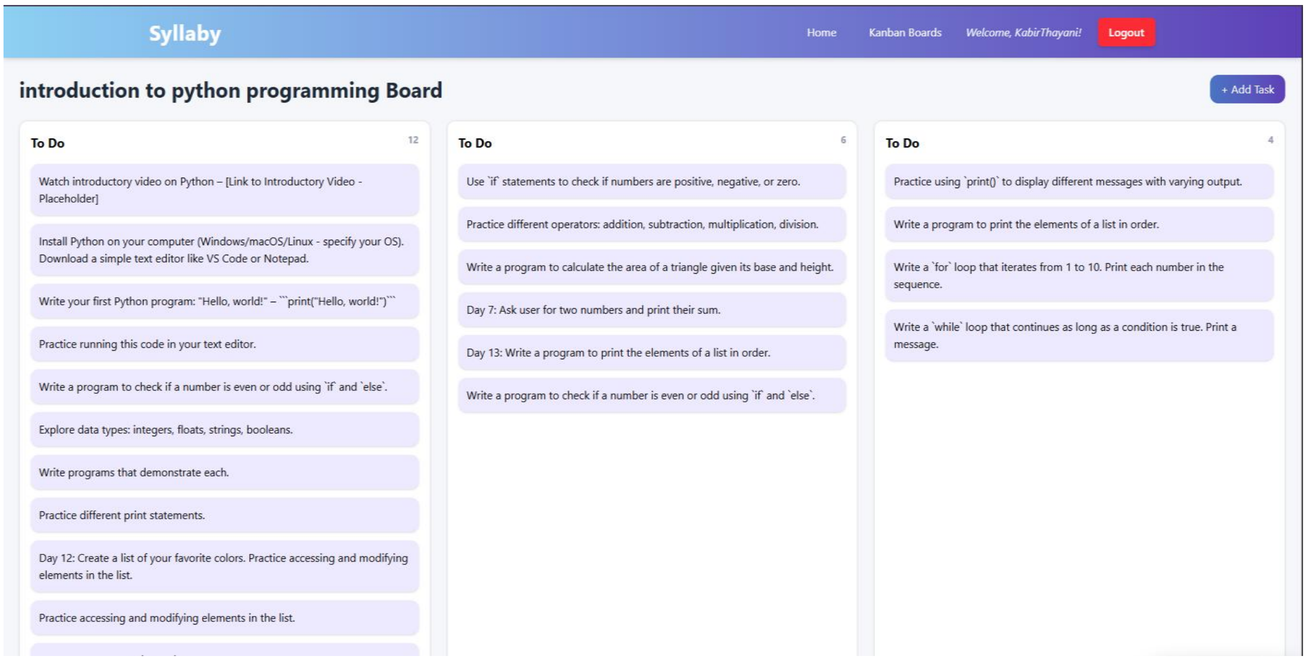
Generate open-ended questions from your study materials and get AI feedback on your answers.

Upcoming Deadlines

Stay on track with your important dates and assignments.

1 fafa - To Do	Overdue
fgag zfa - In Progress	Overdue
afafafa fafgag - In Progress	Overdue
agag fafa - To Do	Overdue

Figure 5.2 - Main User Dashboard



introduction to python programming Board

To Do 12

- Watch introductory video on Python – [Link to Introductory Video - Placeholder]
- Install Python on your computer (Windows/macOS/Linux - specify your OS). Download a simple text editor like VS Code or Notepad.
- Write your first Python program: "Hello, world!" – ```print("Hello, world!")```
- Practice running this code in your text editor.
- Write a program to check if a number is even or odd using `if` and `else`.
- Explore data types: integers, floats, strings, booleans.
- Write programs that demonstrate each.
- Practice different print statements.
- Day 12: Create a list of your favorite colors. Practice accessing and modifying elements in the list.
- Practice accessing and modifying elements in the list.

To Do 6

- Use `if` statements to check if numbers are positive, negative, or zero.
- Practice different operators: addition, subtraction, multiplication, division.
- Write a program to calculate the area of a triangle given its base and height.
- Day 7: Ask user for two numbers and print their sum.
- Day 13: Write a program to print the elements of a list in order.
- Write a program to check if a number is even or odd using `if` and `else`.

To Do 4

- Practice using `print()` to display different messages with varying output.
- Write a program to print the elements of a list in order.
- Write a `for` loop that iterates from 1 to 10. Print each number in the sequence.
- Write a `while` loop that continues as long as a condition is true. Print a message.

Figure 5.3 - Kanban Board View

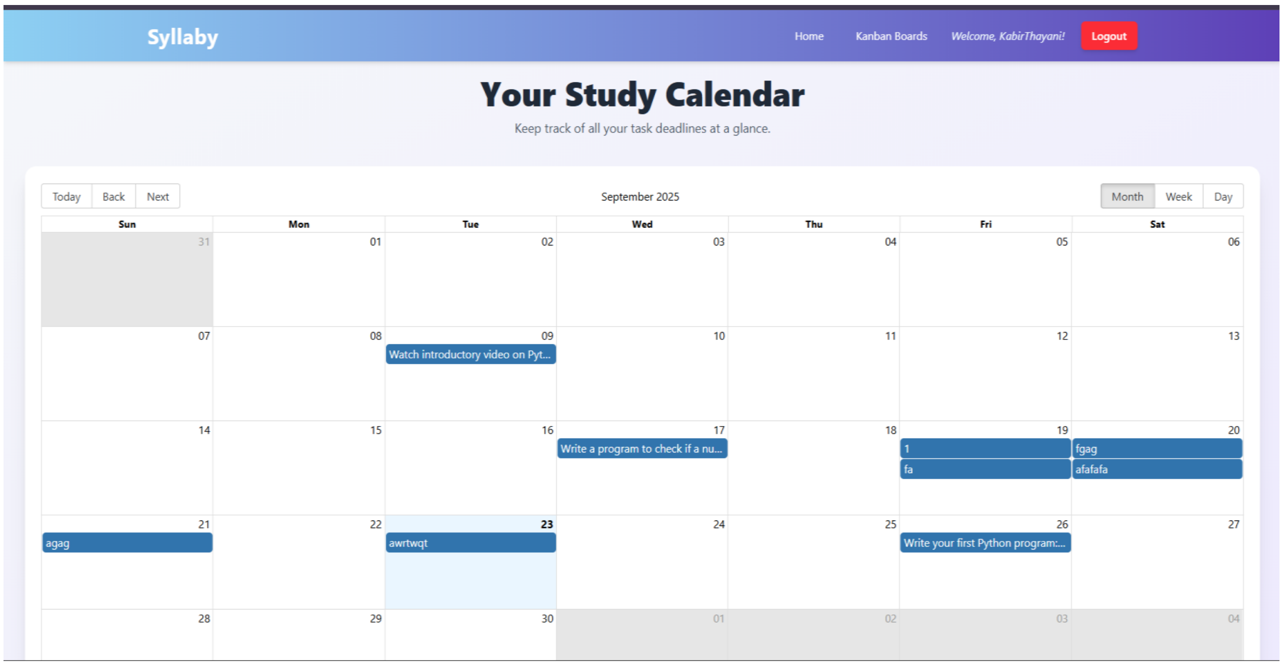
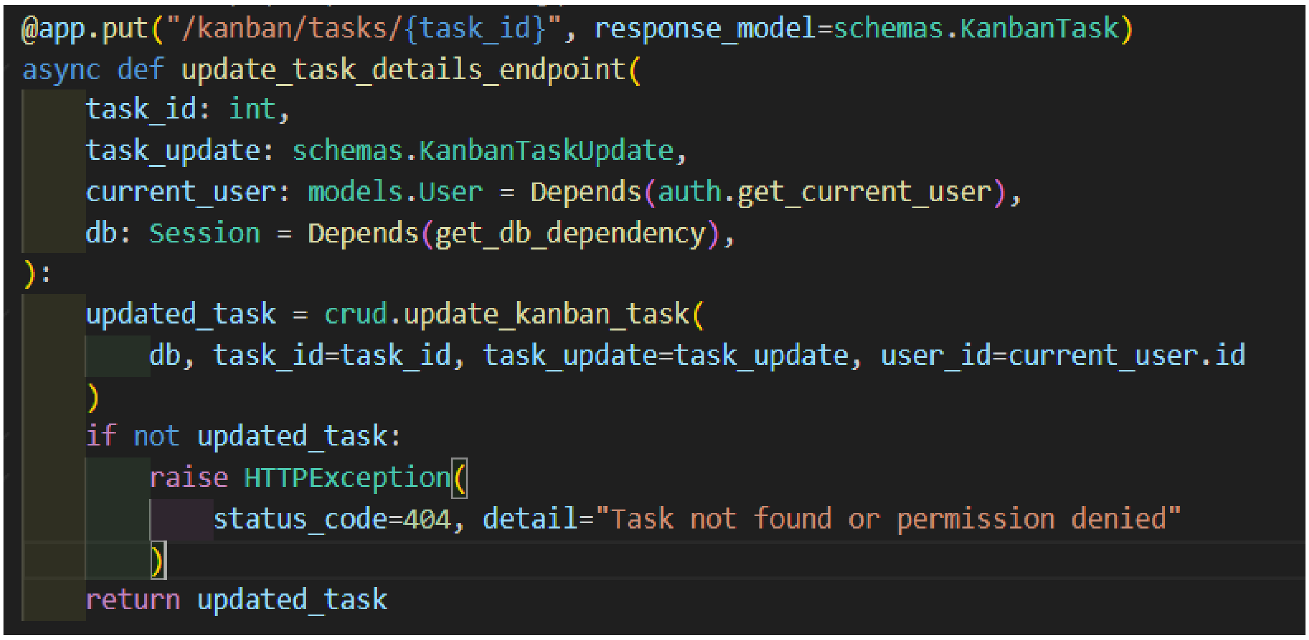


Figure 5.4 - Calendar View of Deadlines

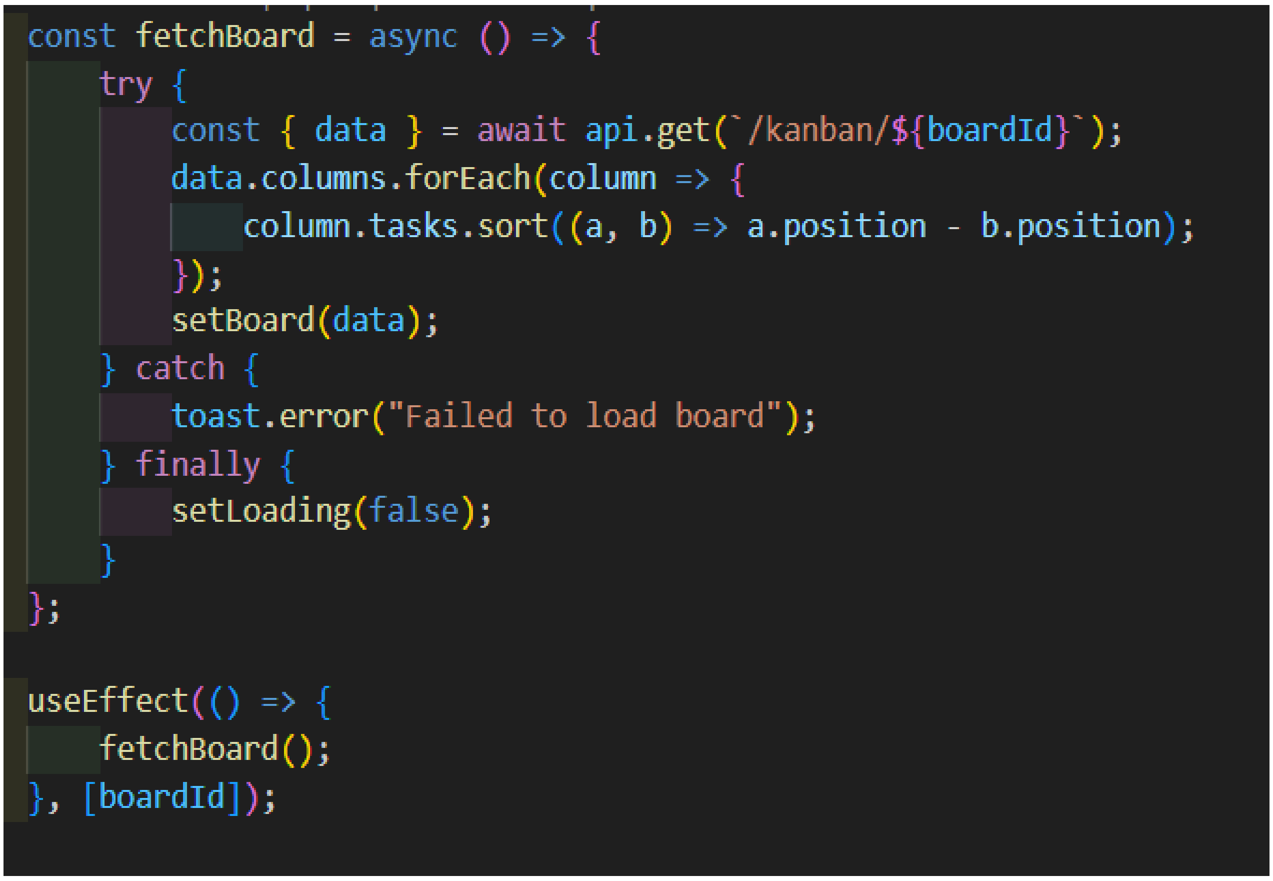
5.4 Sample Coding

Backend: API Endpoint for Updating a Task (main.py)



```python
@app.put("/kanban/tasks/{task_id}", response_model=schemas.KanbanTask)
async def update_task_details_endpoint(
    task_id: int,
    task_update: schemas.KanbanTaskUpdate,
    current_user: models.User = Depends(auth.get_current_user),
    db: Session = Depends(get_db_dependency),
):
    updated_task = crud.update_kanban_task(
        db, task_id=task_id, task_update=task_update, user_id=current_user.id
    )
    if not updated_task:
        raise HTTPException(
            status_code=404, detail="Task not found or permission denied"
        )
    return updated_task
```

React Hook for Fetching Board Data (KanbanBoardPage.jsx)



```
const fetchBoard = async () => {
    try {
        const { data } = await api.get(`/kanban/${boardId}`);
        data.columns.forEach(column => {
            column.tasks.sort((a, b) => a.position - b.position);
        });
        setBoard(data);
    } catch {
        toast.error("Failed to load board");
    } finally {
        setLoading(false);
    }
};

useEffect(() => {
    fetchBoard();
}, [boardId]);
```

CHAPTER 6

TESTING

6.1 Testing Plan

6.2 Test case Modules

6.1 Testing Plan:

Software testing is a critical element of quality assurance, representing the ultimate review of specification, design, and coding. The testing plan for Syllaby was designed to be comprehensive, covering functionality, security, and usability. Key principles included:

Traceability: All tests were designed to be traceable to the functional and non-functional requirements outlined in Chapter 4.

Early Testing: Testing was not a final phase but an integrated part of the Agile development process, with each new feature being tested as it was completed.

Manual & Automated Approaches: A combination of manual testing (for user workflows and UI) and automated principles (using FastAPI's interactive docs for API validation) was used.

6.2 Test Case Module:

User Management Module:

This module handles user registration, login, and secure access to data.
Test Case ID: TC_AUTH_001
Scenario: Verify a new user can successfully register.
Steps: 1. Navigate to the register page. 2. Enter a unique username and a valid password. 3. Click "Register".
Expected Result: User is successfully created in the database, and they are redirected to the login page.
Test Case ID: TC_AUTH_002
Scenario: Verify a user cannot register with an existing username.
Steps: 1. Navigate to the register page. 2. Enter a username that already exists. 3. Click "Register".
Expected Result: An error message is displayed indicating the username is already taken. The API returns a 400 Bad Request status.

Kanban & Task Management Module

This module handles all task-related functionalities.
Test Case ID: TC_KANBAN_001
Scenario: Verify a user can update a task's status via drag-and-drop.
Steps: 1. Navigate to a Kanban board. 2. Drag a task from the "To Do" column. 3. Drop it into the "In Progress" column. 4. Refresh the page.
Expected Result: The task's position is visually updated instantly. After refresh, the task remains in the "In Progress" column, confirming the change was saved to the database.
Test Case ID: TC_KANBAN_002
Scenario: Verify a user can mark a task as complete.
Steps: 1. Click on a task to open the detail modal. 2. Click the "Mark as Complete" button. 3. Close the modal.
Expected Result: The task card on the board is visually updated (e.g., greyed out with a strikethrough) to indicate its completed status.

CHAPTER 7

Limitation of Project

7.1 Limitation of Project

While Syllaby is a robust and functional application, there are several limitations inherent in its current design that offer opportunities for future work.

Dependency on Local AI: The application requires a locally running instance of Ollama, making public deployment and scalability challenging. A production version would need to use a cloud-based API for the LLM.

No Real-Time Collaboration: The system is built for a single user and does not support real-time updates. Changes made in one browser tab will not automatically reflect in another without a manual refresh.

Limited Customization: Users cannot create custom Kanban columns or modify the AI prompts used for generating study plans.

Basic User Authentication: The authentication system lacks features like "Forgot Password" functionality or social logins (e.g., Login with Google)

CHAPTER 8

CONCLUSION

8.1 Conclusion

This project successfully demonstrates the creation of a modern, interactive web application that leverages Artificial Intelligence to solve a common problem for students: academic organization. The project achieved its goal of building an intelligent study planner, Syllaby, that transforms unstructured course information into actionable tasks and schedules.

The development process provided invaluable hands-on experience with a full-stack technology set, including a Python-based FastAPI backend, a React frontend, and a PostgreSQL database. A key learning outcome was the successful integration and prompt engineering required to reliably harness the power of a Large Language Model for practical tasks like content generation and data extraction.

Syllaby, in its current form, is a powerful proof-of-concept. It validates the idea that AI can significantly reduce the administrative burden of studying, allowing students to focus more on learning. The project lays a strong foundation for future enhancements, such as real-time collaboration, deeper AI integration, and greater user customization

Chapter 9

Reference

9.1 Reference

Backend Technologies:

FastAPI Documentation: https://fastapi.tiangolo.com/
Python Official Website: https://www.python.org/
SQLAlchemy ORM Documentation: https://www.sqlalchemy.org/
Pydantic Documentation: https://pydantic-docs.helpmanual.io/
Ollama - Large Language Models: https://ollama.com/

Frontend Technologies:

React Official Documentation: https://react.dev/
React Router Documentation: https://reactrouter.com/
Axios (HTTP Client): https://axios-http.com/
@hello-pangea/dnd (Drag and Drop Library): https://github.com/hello-pangea/dnd
react-big-calendar Documentation: http://jquense.github.io/react-big-calendar/
Tailwind CSS: https://tailwindcss.com/

Database:

PostgreSQL Official Website: https://www.postgresql.org/