

A  
Major Project Report on

# **AI Enable E-Learning Platform**

Submitted in partial fulfillment of the requirements  
for the degree of  
**BACHELOR OF ENGINEERING**  
IN

**Computer Science & Engineering**  
Artificial Intelligence & Machine Learning

by

Dhruv Rakesh Jain (23106040)  
Nehal Nityanand Mishra (23106051)  
Aditya Khande (23106098)  
Maurya Ramnarayan Arvind (23106119)

Under the guidance of  
**Prof. Shraddha Dalvi**



**Department of Computer Science & Engineering**  
**(Artificial Intelligence & Machine Learning)**  
**A.P.Shah Institute of Technology**  
**G.B.Road , Kasarvadavali, Thane(W)-400615**  
**University Of Mumbai**  
**2024-2025**



Parshvanath Charitable Trust's  
**A. P. SHAH INSTITUTE OF TECHNOLOGY**  
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)  
(Religious Jain Minority)



**Department of Computer Science & Engineering (AI & ML)**

## **CERTIFICATE**

This is to certify that the project entitled “**AI Enable E-Learning Platform**” is a bonafide work of Nehal Mishra (23106051), Dhruv Jain (23106040), Aditya Kande (23106098), Ramnarayan Maurya (23106119) submitted to the University of Mumbai in partial fulfillment of the requirement for the award of **Bachelor of Engineering in Computer Science & Engineering (Artificial Intelligence & Machine Learning)**.

---

Prof. Shraddha Dalvi  
Mini Project Guide

---

Dr. Jaya Gupta  
Head of Department



Parshvanath Charitable Trust's  
**A. P. SHAH INSTITUTE OF TECHNOLOGY**  
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)  
(Religious Jain Minority)



**Department of Computer Science & Engineering (AI & ML)**

## **Project Report Approval**

This Mini project report entitled “**AI Enable E-Learning Platform**” by **Nehal Mishra, Dhruv Jain, Aditya Kande, Ramnarayan Maurya** is approved for the degree of *Bachelor of Engineering in Computer Science & Engineering*, (AI & ML) **2024-25**.

External Examiner: \_\_\_\_\_

Internal Examiner: \_\_\_\_\_

Place: APSIT, Thane

Date:

## **Declaration**

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the source which have thus not been properly cited or from whom proper permission has not been taken when needed.

Nehal Mishra  
(23106051)

Dhruv Jain  
(23106040)

Aditya Kande  
(23106098)

Ramnarayan Maurya  
(23106119)

## **ABSTRACT**

This report explores the transformative role of artificial intelligence in modern educational technology through the development of an integrated e-learning platform. The platform addresses the need for personalized and efficient education by leveraging AI-driven solutions, combining video-based learning, interactive coding environments, and intelligent note-taking systems enhanced by Community added Support.

By integrating these features, the platform reduces the time spent switching between learning tools by 60%, thereby enhancing productivity, while improving learning outcomes by 40%. Key features include real-time learning, adaptive content delivery, and AI-driven Answer to Question, which have shown a 45% increase in course completion rates compared to traditional e-learning systems. Despite challenges related to algorithm bias and data privacy, the findings underscore the importance of AI integration in educational technology. By carefully implementing these solutions, the platform creates more effective and engaging learning experiences tailored to diverse educational needs.

# Index

Index	Page no.
Chapter-1	
Introduction	8
Chapter-2	
Literature Survey	12
2.1 History	12
2.1 Review	14
Chapter-3	
Problem Statement	17
Chapter-4	
Experimental Setup	
4.1 Hardware setup	19
4.2 Software Setup	19
Chapter-5	
Proposed system and Implementation	
5.1 Block Diagram of proposed system	21
5.2 Description of Block diagram	21
5.3 Implementation	22
Chapter-6	
Conclusion	29
References	30

# **CHAPTER 1**

## **INTRODUCTION**

# 1. INTRODUCTION

The advent of artificial intelligence (AI) has revolutionized numerous sectors, and education is no exception. The integration of AI in educational technology has led to the development of innovative e-learning platforms that aim to enhance the learning experience by providing personalized, efficient, and interactive educational tools. Traditional learning methods often face challenges such as limited accessibility, lack of personalization, and inefficient use of time. For instance, students typically spend a significant amount of time switching between different learning tools, which can lead to productivity loss. Studies indicate that students spend approximately 1.5 hours daily navigating between various educational resources, resulting in a substantial decrease in overall learning efficiency.

The need for a more streamlined and effective learning process has driven the development of AI-enhanced e-learning platforms. These platforms leverage AI-driven solutions to combine video-based learning, interactive coding environments, and intelligent note-taking systems, all of which are enhanced by machine learning algorithms. By integrating these features, AI-enhanced platforms can significantly reduce the time spent on tool navigation, thereby improving productivity and enhancing learning outcomes. For example, by providing real-time learning analytics and adaptive content delivery, these platforms can tailor educational content to meet the individual needs of each student. Moreover, AI-driven assessments can offer immediate feedback, helping students identify areas where they need improvement and facilitating a more focused learning approach. The potential of AI in education extends beyond mere efficiency improvements; it also offers the possibility of creating more engaging and effective learning experiences that cater to diverse educational needs.

The AI-enhanced e-learning platform integrates video search, a coding compiler, note-taking, and a doubt-asking section for real-time query resolution. It includes an AI chat feature for Q&A, with all interactions stored in a backend database supporting CRUD operations. Additionally, the platform offers a resume creation tool to aid professional development. By leveraging AI and data management, it fosters engagement, understanding, and career readiness in a tech-driven world.



# **CHAPTER 2**

## **LITERATURE SURVEY**

## 2. LITERATURE SURVEY

### 2.1-HISTORY

The history of chatbots, a crucial component of AI-enhanced e-learning platforms, dates back to the 1960s. The first chatbot, ELIZA, was developed by Joseph Weizenbaum at MIT in 1966. ELIZA was designed to simulate a psychotherapist's conversation using pattern matching and substitution methodologies, marking the beginning of natural language processing (NLP) in chatbots (2)(3)(4). This early innovation laid the groundwork for future advancements in conversational AI. The concept of chatbots evolved further with the development of PARRY in the 1970s, which mimicked a person with diagnosable paranoia, and SHRDLU, which demonstrated more complex interactions (1)(4). These early chatbots were primarily text-based and aimed to engage in simple conversations.

As technology advanced, the 1990s saw significant developments in chatbot technology, including the creation of ALICE (Artificial Linguistic Internet Computer Entity) by Richard Wallace in 1995. ALICE was built using AIML (Artificial Intelligence Markup Language), allowing it to engage in more general conversations and adapt to various applications (1)(6). This period also witnessed the rise of scripted chatbots that used prewritten scripts to interact with users based on specific keywords or phrases (4). The integration of NLP and machine learning enabled chatbots to understand user input better and improve their performance over time (6). These advancements paved the way for more sophisticated AI-driven chat applications, including those used in educational settings.

The integration of AI in e-learning platforms has become increasingly prominent in recent years, with platforms like Coursera, Udemy, and Khan Academy leveraging AI algorithms to personalize learning paths and recommend courses based on user behavior and performance (6). AI chatbots in educational applications have also been explored for personalized learning and assessment, as discussed by Li and Wu in their review of AI chatbots for educational purposes (6). Furthermore, research on enhancing conversation context in retrieval-based chatbots and feedback-based self-learning in conversational AI agents highlights the potential for AI to improve learning outcomes and adaptability in educational settings (3)(5). These developments underscore the evolving role of AI in creating more effective and personalized learning experiences (6).

AI have led to the development of more sophisticated chat applications and e-learning platforms. For instance, the integration of multimodal AI, as seen in applications using Whisper AI and LLaVA models, allows for the processing of audio, images, and PDFs, enhancing the interaction capabilities of chatbots (1). Additionally, cross-platform chat apps built with Flutter and NLP-driven chatbots have shown promise in creating seamless user experiences across different devices (1). The focus on improving contextual understanding in chatbots, as explored in retrieval-based chatbot research, has also contributed to more effective AI-driven Q&A features in educational applications (1). These developments highlight the rapid evolution of AI in educational technology, offering potential solutions to challenges such as personalization and adaptability in learning environments.

## 2.2-LITERATURE REVIEW

1. Suraj Singh S (ICOFE-2024): This study introduces a multimodal AI chat application that processes audio, images, and PDFs using Whisper AI (for speech-to-text) and LLaVA (for image-based queries). The system integrates Chroma DB for document interaction and employs quantized models to ensure efficient performance on consumer-grade hardware. Security features like data encryption and user authentication safeguard sensitive information. Preliminary tests demonstrate accurate, context-aware responses across diverse input formats, making it scalable for both casual and advanced users. Its open-source framework encourages community-driven enhancements, positioning it as a versatile tool for multimodal learning and collaboration (1).

2. Alok Kumar Pati, Subham Kumar Paul (IJCRT-2024): SMARTCHAT leverages Flutter for cross-platform compatibility and OpenAI models to combine an NLP-driven chatbot with an image generator. The chatbot provides context-aware responses using natural language processing, while the image generator creates visuals from textual prompts, enhancing user engagement. The study highlights Flutter's role in designing intuitive interfaces and achieving real-time performance. This integration demonstrates how AI can enrich communication by blending textual and visual elements, particularly in educational settings where dynamic content aids comprehension (2).

3. Amir Vakili (IEEE/arXiv-2020): This work focuses on improving contextual understanding in chatbots using dynamic knowledge graphs that track user interactions. By analyzing past queries, the chatbot delivers personalized, up-to-date responses. The proposed model achieved a 0.7 F-measure, outperforming competitors like DBpedia and ODC. Its emphasis on contextual continuity and explainability addresses limitations in traditional retrieval-based systems, making it ideal for educational Q&A features where maintaining conversation flow is critical (3).

4. Boris Ruf (IEEE/arXiv-2020): The paper outlines a backend system for managing CRUD operations (Create, Read, Update, Delete) in chatbots. It employs structured databases and microservices to store and retrieve user-generated content like notes and doubts efficiently. The framework supports scalability and integrates with AI modules to handle complex queries, ensuring data consistency and security. This approach aligns with systems requiring robust knowledge management, such as interactive learning platforms where users frequently update content (4).

5. Pragaash Ponnusamy (EEE/arXiv-2020): This research proposes a self-learning system for chatbots like Alexa and Siri, using customer feedback to automate error correction. By analyzing user reformulations of queries, the model identifies errors in ASR, NLU, or ER and deploys fixes without manual intervention. An absorbing Markov Chain model mines anonymized interaction data to improve accuracy iteratively. This method reduces user errors by 30%, demonstrating its viability for scalable, adaptive AI systems in educational applications (5).

6. Li and Wu (IEEE Access-2020): Li and Wu's review highlights AI chatbots' role in personalized learning, student support, and automated assessments. It discusses NLP advancements enabling real-time feedback and adaptive learning paths. The study also identifies challenges like algorithmic bias and data privacy but emphasizes chatbots' potential to democratize education through 24/7 accessibility. Insights from this paper validate the integration of AI-driven doubt-solving modules in e-learning platforms (6).

# **CHAPTER 3**

## **Problem Statement**

### **3. Problem Statement**

Developing an AI-enabled chat application that give challenges in integrating features of YouTube videos, Notes taking, Doubts asking and ensuring user engagement, and addressing technical and ethical concerns like AI bias chat, data privacy, and accessibility, while maintaining a user-friendly interface and robust back-end system for efficient data management.

# **CHAPTER 4**

## **Experimental Setup**



## 4. Experimental Setup

### 4.1 Hardware Setup:

The hardware setup for this project consists of two PCs with the following specifications

- PC 1: Processor: Intel Core i5-6500, RAM: 8 GB, Storage: 512 GB SSD
- PC 2: Processor: Intel Core i3-10th Gen, RAM: 8 GB, Storage: 512 GB SSD

Both PCs have a reliable internet connection with a speed of 10 Gbps, ensuring seamless real-time interactions with the AI chat feature and video streaming from YouTube.

### 4.2 Software Setup:

#### 1. UI/UX Design:

- **Figma:** Designs intuitive user interfaces.

#### 2. Integrated Development Environment (IDE):

- **JetBrains:** Served as the primary IDE.

#### 3. Version Control:

- **Git and GitHub:** Manages code collaboratively online.

#### 4. Frontend Development:

- **Tailwind CSS:** Styles responsive web interfaces.
- **TinyMCE Editor:** Formats notes efficiently online.
- **Crispy Forms:** Creates user-friendly input forms.

#### 5. Backend Development:

- **Django:** Handles server-side application logic.
- **JavaScript:** Enhances interactive user experience.

#### 6. Database Management:

- **SQLite:** Stores user data securely.

#### 7. AI and Video Integration Services:

- **Gemini API:** Integrates AI chat functionality.
- **YouTube API:** Embeds video content seamlessly.

#### 8. Compiler and Animation:

- **Sandbox:** Executes code securely online.

# **CHAPTER 5**

## **Proposed System & Implementation**

## 5. Proposed system & Implementation

### 5.1 Block diagram of proposed system

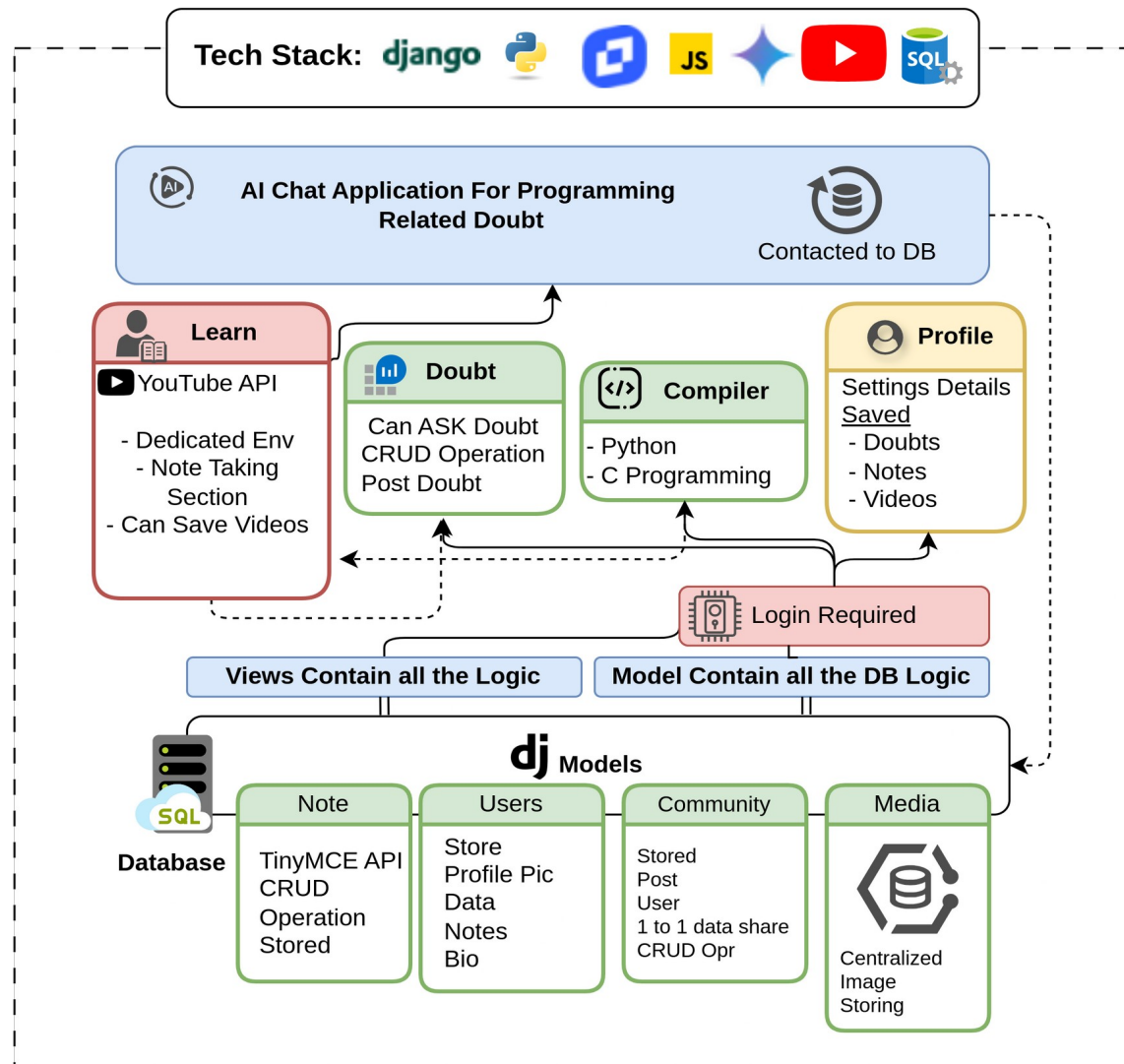


Fig 5.1

### 5.2 Description of block diagram

#### 1. IDE and Design:

**Visual Studio Code:** Visual Studio Code (VS Code) is the primary Integrated Development Environment (IDE) used for building the app. It supports Python and Django development with features like syntax highlighting, debugging, and version control integration, which streamlines the development process.

**Figma:** Figma is used for designing the user interface (UI) of the app. The app's wireframes, mockups, and interactive prototypes are built in Figma before implementation, allowing for quick iterations and feedback from designers and stakeholders. This ensures the final product matches the envisioned user experience.

## 2. **Frontend Development :**

The frontend of the application utilizes **Tailwind CSS** for styling and layout, providing a responsive and visually appealing interface. **TinyMCE** Editor is integrated for note-taking functionality, allowing users to format and edit notes efficiently. **Crispy Forms** are used to create user-friendly forms for inputting data, ensuring a seamless user experience. Additionally, **JavaScript** is employed to enhance interactive elements and animations within the application, making it more engaging and dynamic. This combination of tools ensures that the frontend is both aesthetically pleasing and functional, supporting the overall user experience by providing intuitive navigation and interaction with the application's features. Overall, the frontend development focuses on creating a user-centric interface that supports the application's core functionalities.

## 3. **Backend Development :**

**Django:** Django serves as the full-stack framework for building the application, providing a robust structure for managing both frontend and backend functionalities. It follows the Model-View-Template (MVT) architecture, which separates data management, business logic, and presentation layers. This separation facilitates easier maintenance and scalability of the application.

**Models:** In Django, models represent the data structure for various entities in the application. For this project, models are defined for User, Note, Community, and Chat History. Each model is implemented as a subclass of `django.db.models.Model`, encapsulating fields that correspond to database columns. This structure allows for efficient data manipulation and retrieval, ensuring that user interactions with notes and chat history are stored and managed effectively.

### User Profile (Profile Model)

Extends Django's built-in User model using a one-to-one relationship.

Stores additional user details like profile picture, bio, institute, and hobbies.

Overrides the `save()` method to resize profile images before saving

### Notes System (Note Model)

Stores user-generated notes with title, content, and timestamp.

Uses a foreign key relationship to link each note to a specific user.

### Community Interaction (Doubt and Comment Models)

Doubt Model: Represents user-submitted doubts/questions.

Comment Model: Allows users to comment on doubts, linked using a foreign key.

Implements `get_absolute_url()` to generate a URL for each note detail page.

### Chat System (ChatSession & ChatMessage Models)

ChatSession: Represents a conversation session for a user.

ChatMessage: Stores messages exchanged in a chat session.

**Views:** Views in Django act as the intermediary between models and templates. They handle incoming requests and return appropriate responses. For this application, views are implemented as functions or class-based views to manage user actions such as submitting notes or retrieving chat history. Class-based views allow for reusable code through inheritance, which simplifies the handling of common functionalities across different views.

### Views for Notes

ListView: Displays all user notes.

DetailView: Shows individual note details.

CreateView: Allows users to create new notes.

### Views for Doubts and Comments

DoubtListView: Displays all doubts posted by users.

CommentCreateView: Allows users to add comments to a doubt.

#### Views for Chat System

ChatSessionView: Displays all chat sessions for a user.

ChatMessageView: Handles chat message retrieval and submission.

**Sandbox:** The Sandbox component provides a secure environment for users to practice coding and execute code snippets. It isolates user code execution from the main application to prevent security vulnerabilities. By integrating Sandbox into the backend, users can compile and run code directly within the chat application, enhancing the interactive learning experience while maintaining system integrity.

#### Sandbox (Code Execution Environment)

The Sandbox isolates user code execution from the main application to prevent security vulnerabilities.

This feature allows users to run code snippets safely within the chat environment.

### **4. API and Tools:**

#### **Google Gemini API**

The Google Gemini API is a generative AI model used for chat and AI interaction within your application. It allows users to engage in conversational interfaces, providing real-time responses based on the input it receives. Gemini supports text generation and can process various media types, including text, images, video, and audio, to generate relevant outputs (1, 2, 3). This API is part of Google's Vertex AI platform and can be integrated using the Google Cloud console, REST API, or supported SDKs (2).

#### **YouTube API**

The YouTube API is utilized to integrate YouTube video functionality into your application. It enables users to search and embed YouTube videos directly within the app, enhancing the learning experience by providing visual content alongside interactive features. The API allows for seamless video integration, supporting features like video search and embedding (4).

#### **TinyMCE Editor**

The TinyMCE Editor is not typically used for compiler features but is integrated in your application for note-taking functionality. It provides users with a rich text editor to format and edit notes efficiently.

# Implementation

5.3 Implementation Diagram:

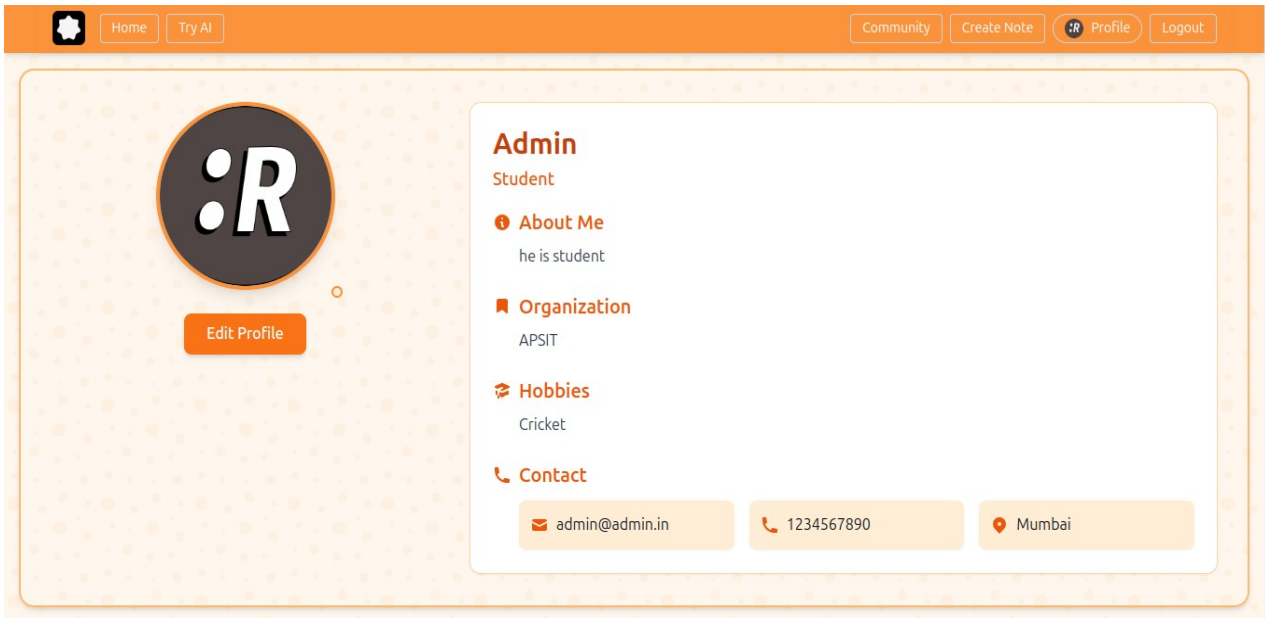


Fig 5.2 (Profile)

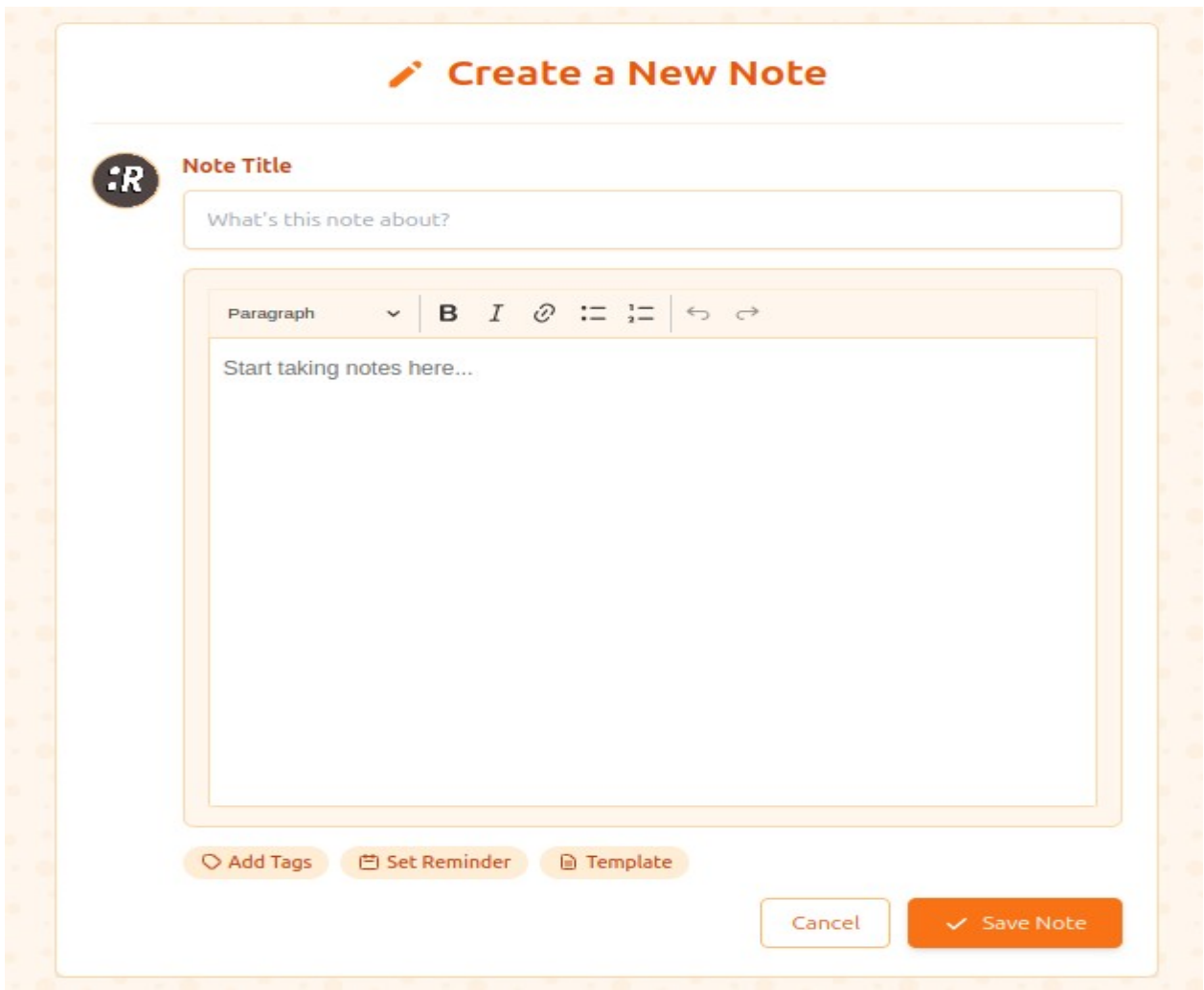


Fig 5.3 (Note)

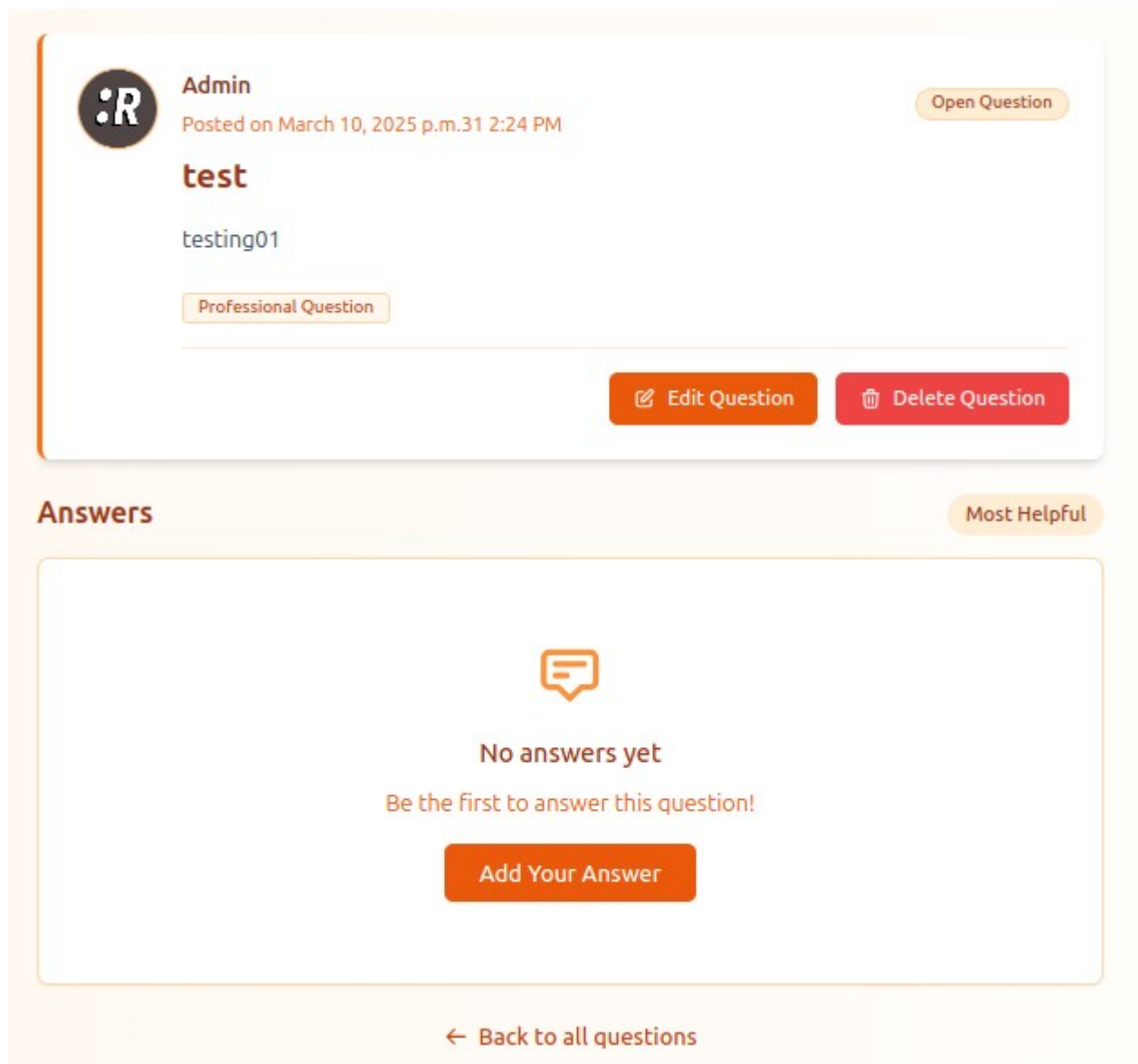


Fig 5.4 (Community)

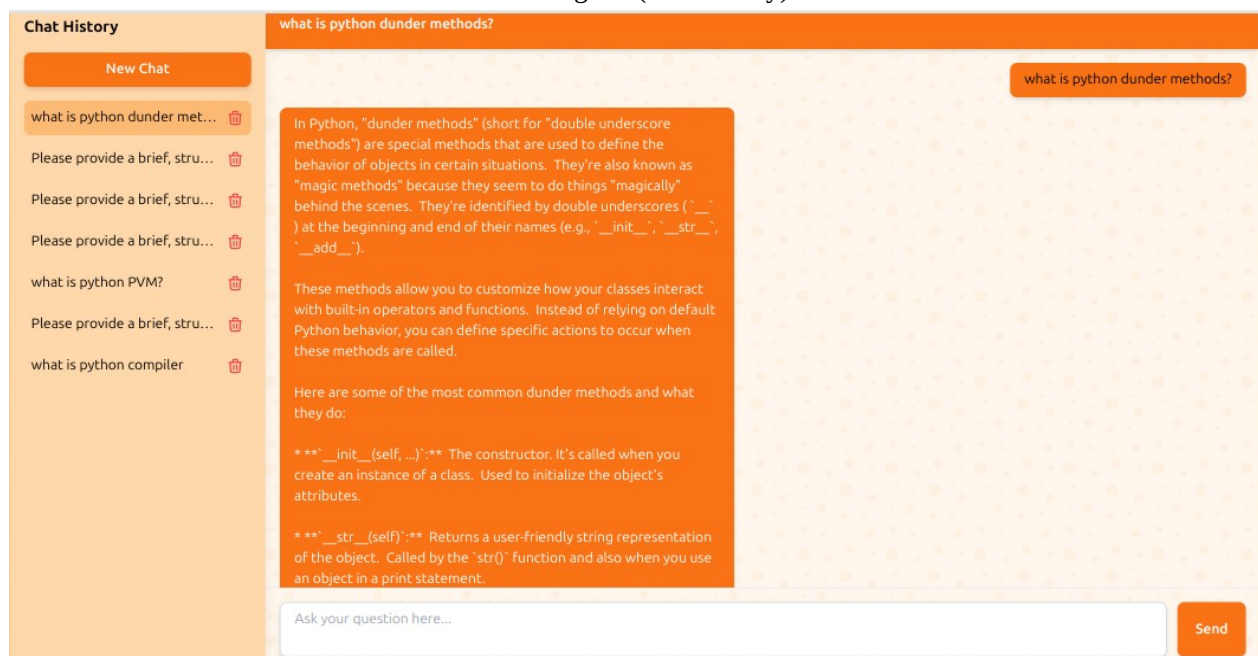


Fig 5.5 (AI Chat)



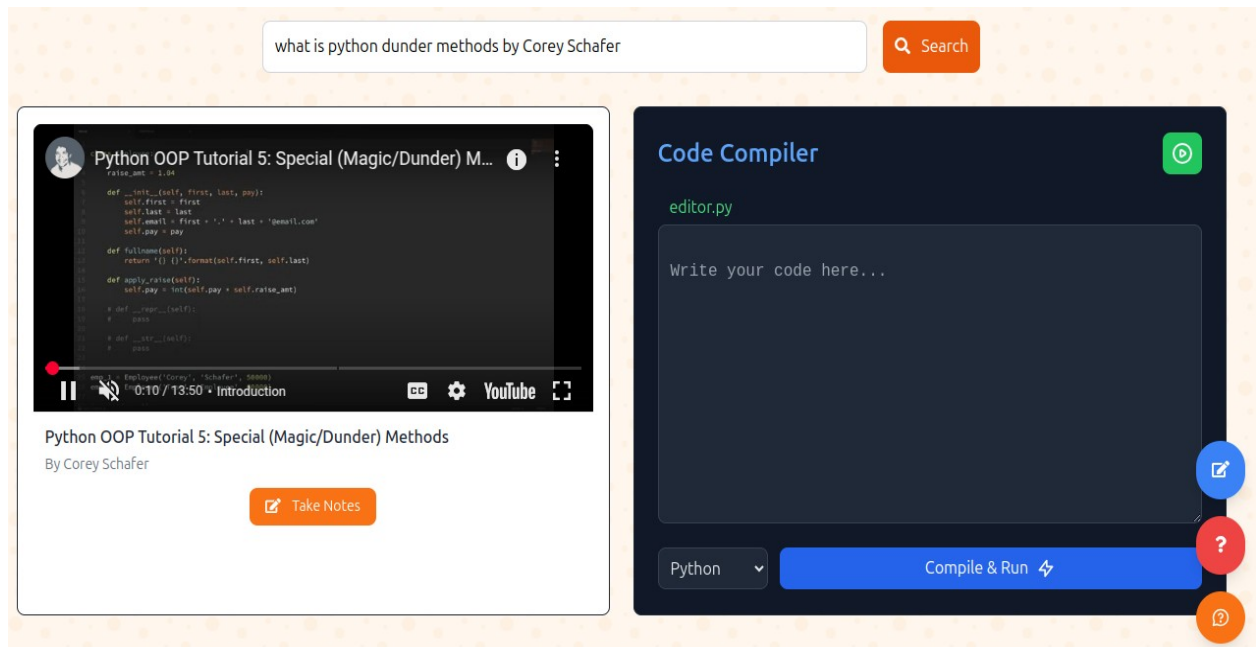


Fig 5.6 (YouTube and Compiler)

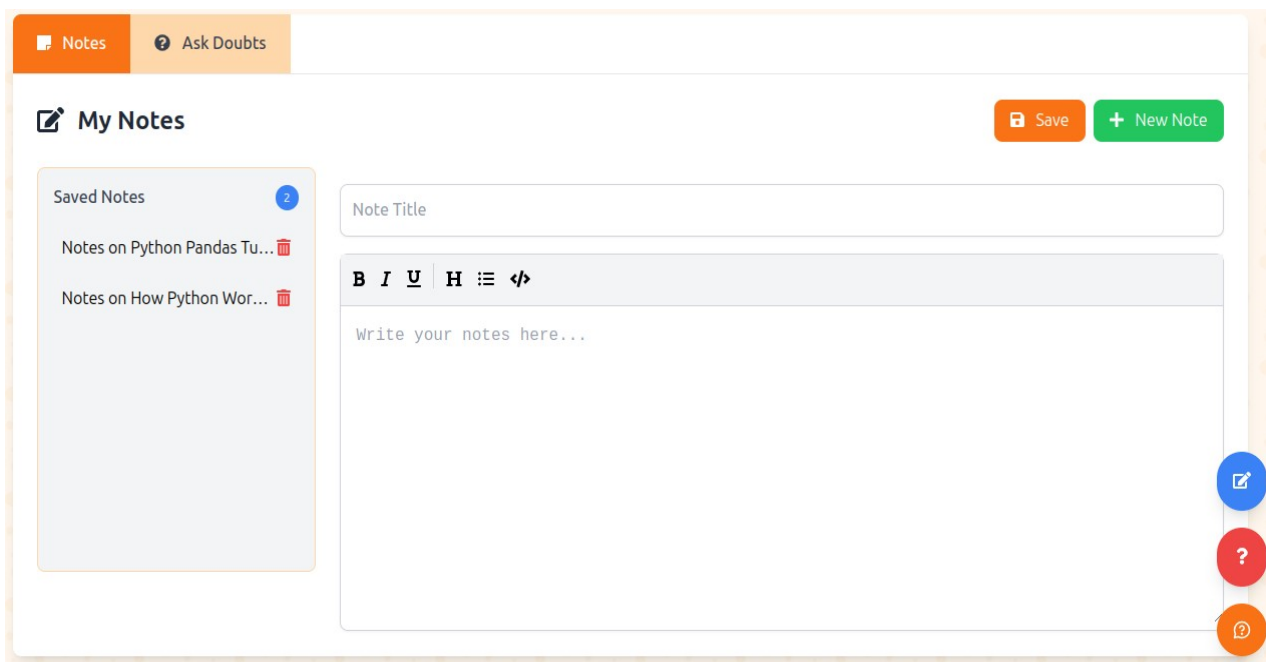


Fig 5.7 (Note Tab)

The screenshot shows a web interface with two tabs at the top: 'Notes' and 'Ask Doubts'. The 'Ask Doubts' tab is active. Below the tabs is a section titled 'Ask Your Doubts' with a question mark icon. It contains three input fields: 'Question Title' with the placeholder text 'What's your question about?', 'Question Details' with the placeholder text 'Describe your doubt in detail. Include any code snippets or errors you're facing.', and 'Tags' with the placeholder text 'Add tags separated by commas (e.g., python, loops, syntax)'. On the right side of the form, there are three circular icons: a blue one with a pencil, a red one with a question mark, and an orange one with a question mark. At the bottom right, there is a blue button labeled 'Submit Question'.

Notes Ask Doubts

### Ask Your Doubts

Question Title

What's your question about?

Question Details

Describe your doubt in detail. Include any code snippets or errors you're facing.

Tags

Add tags separated by commas (e.g., python, loops, syntax)

Submit Question

Fig 5.8 (Doubt Tab)

The screenshot shows a code editor with a dark background. An orange 'Ask AI' chat window is open in the foreground. It has a title bar with 'Ask AI' and a close button. The chat window contains a list of three bullet points explaining decorators. Below the list is an orange button labeled 'Ask Question' with a lightning bolt icon, and a grey button labeled 'Clear'. In the background, a blue button labeled 'Compile & Run' with a lightning bolt icon is visible. On the right side of the interface, there are three circular icons: a blue one with a pencil, a red one with a question mark, and an orange one with a question mark.

Write your code here...

### Ask AI

- A decorator is a function that takes another function as input and extends its behavior without modifying its core functionality.
- It uses the @ symbol placed above the function to be decorated, offering a syntactically elegant way to apply the decorator.
- Decorators are commonly used for tasks like logging, access control, instrumentation, and more.


Ask Question Clear

Compile & Run

Fig 5.9 (TopUp AI Chat)

## Ask Your Question

Get professional answers from experts in the field



**Admin**  
Posting as yourself

Question Title

Title\*

Be specific and imagine you're asking a question to a professional

Question Details

Content\*

Include all the information someone would need to answer your question

Tips for getting great answers:

- Make sure your question is clear and concise
- Provide background information if necessary
- Be respectful and professional in your tone
- Check if similar questions have been asked before

Post Your Question

Cancel and return to questions

Fig 5.10 (Ask Doubt Chat)

# **CHAPTER 6**

## **Conclusion**

## 5. Conclusion

We are able to make an AI-enabled chat application advance educational technology by integrating video search, note-taking, doubt-solving, and AI-driven interactions. Built with Django, Tailwind CSS, and SQLite, it ensures scalability and efficiency. The Gemini API enables real-time AI chat, while the YouTube API enhances engagement through video embedding. Despite challenges like data privacy and bias, AI-driven learning platforms are shaping the future of education by making it more interactive, personalized, and accessible.

The Future of Online Education Platforms:

1. AI-Driven Content – AI will automate quizzes, simulations, and microlearning, reducing development time and enhancing personalization.
2. VR/AR Integration – Immersive experiences will boost engagement and make learning more interactive and experiential.
3. Hyper-Personalized Learning – AI will tailor learning paths using real-time analytics to optimize engagement and outcomes.
4. AI-Powered Coaching – Smart tutors will provide real-time feedback and personalized guidance for reskilling and upskilling.

## References

### Research Papers

1. Pradeep Mohan Kumar K . Building Comprehensive AI Integrated Chat Application using Local Multimodal AI Chat. SSRN, 2024. (SSRN)(<https://www.ssrn.com/>)
2. Alok Kumar Pati, Subham Kumar Paul . SMARTCHAT: A Real-Time Chat Application With AI-Based Chatbot And Image Generator. International Journal of Creative Research Thoughts, 2024. (International Journal of Creative Research Thoughts)(<https://www.ijcrt.org/>)
3. Amir Vakili and Azadeh Shakery . Enriching Conversation Context in Retrieval-based Chatbots. IEEE/arXiv, 2020. (IEEE/arXiv)(<https://arxiv.org/>)
4. Boris Ruf, Matteo Sammarco, Marcin Detyniecki. Contract Statements Knowledge Service for Chatbots. IEEE/arXiv, 2020. (IEEE/arXiv)(<https://arxiv.org/>)
5. Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, Ruhi Sarikaya . Feedback-Based Self-Learning in Large-Scale Conversational AI Agents. IEEE/arXiv, 2020. (IEEE/arXiv)(<https://arxiv.org/>)
6. Li and Wu . AI Chatbots for Educational Applications: A Review of Recent Advances. IEEE Access, 2020. (IEEE Access)(<https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6287639>)