Lebanese University

Faculty of Engineering III

Electrical and Electronic Department

Computer and Communication

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

AI Study Buddy- An Intelligent Learning Agent

Mini project

Done By:

**Deema Fadi Azzam 6302**

**Mariam Ahmad Farhat 6503**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Spring 2024-2025**

**Supervisor: Dr. Mohammad Aoude**

**Table of Contents:**

*Chapter 1: Introduction ………………………………………………………………pages 5-7*

1.1 Background ......................................................... Page 5

1.2 Problem Statement ....................................... Page 5

1.3 Objectives of the Project ............................. Page 5--6

1.4 Scope and Limitations .................................. Page 6

1.5 Report Organization ...................................... Page 6-7

*Chapter 2: Literature Review…………………………………………………………..pages 8-10*

2.1 Overview of AI in Education ....................... Page 8

2.2 AI Agents for Learning Support ................ Page 8

2.3 Existing Educational Platforms .................. Page 9

2.4 Comparative Analysis of Solutions ........ Page 9

2.5 Summary of Research Gaps ...................... Page 10

*Chapter 3: System Design…………………………………………………………..pages 11-16*

3.1 System Overview .......................................... Page 11

3.2 Architectural Design ...................................... Page 12

3.3 Frontend Design ............................................. Page 13

3.4 Backend Design .............................................. Page 13-14

3.5 Database Schema Design ............................. Page 14

3.7 User Flow and Use Cases ............................. Page 15-16

*Chapter 4: Implementation and Results………………………………pages 17-32*

4.1 Tools and Technologies Used ..................... Page 17

4.2 Frontend Implementation ............................ Page 17-25

4.3 Backend and API Integration ..................... Page 25-26

4.4 AI Model Integration................... Page 27-28

4.5 Key Features in Action ................................. Page 28-31

4.6 Testing and Evaluation ................................ Page31

*Chapter 5: Conclusion and Future Work……………………………………………pages 32-33*

5.1 Summary of Achievements .......................... Page 32

5.2 Challenges Faced ............................................ Page 32

5.3 Limitations .......................................................... Page 32

5.4 Future Enhancements .................................... Page32-33

# Abstract

This project presents AI Study Buddy, an intelligent, web-based learning companion designed to enhance self-learning through automated content summarization, intelligent quiz generation, and real-time explanations. The system leverages advanced natural language processing models (Gemma3) to extract and simplify key concepts from course materials and generate customized quizzes, including multiple-choice and true/false questions.

The frontend, built using React.js and Tailwind CSS, offers an intuitive user interface for content upload, interactive quizzes, and gamified learning modes such as “Battle Mode.” The backend, powered by FastAPI, handles user management, API routing, and communication with the AI services. Data is persistently stored using MySQL, enabling retrieval of user history, progress, and uploaded materials.

A modular architecture separates AI logic into dedicated services: a Summarizer, an Explainer, and a Quiz Generator. These components operate asynchronously to maintain a responsive user experience and allow scalable integration of new features. The system emphasizes educational value and interactivity, making learning more engaging and efficient for students.

Performance evaluation includes assessment of summarization quality, quiz accuracy, system latency, and user engagement levels. By combining machine learning, structured web technologies, and gamified interaction, AI Study Buddy offers a robust platform with applications in education, tutoring, and self-paced learning.

Keywords: AI Agent, Educational Technology, Summarization, Quiz Generation, React, FastAPI, Gemma, Natural Language Processing, EdTech, Self-Learning, Gamification, Performance Feedback.

1. Introduction

## 1.1 Background

In recent years, the integration of artificial intelligence (AI) into education has significantly transformed how students interact with learning materials. Traditional methods of studying, such as reading static notes or solving manual practice exercises, often fall short in terms of engagement, efficiency, and personalization. As the volume of educational content increases, students struggle to manage their time and retain key information effectively.

With the advancement of natural language processing and machine learning models, AI-powered tools have become capable of summarizing large texts, generating contextually relevant questions, and offering interactive feedback. These capabilities have given rise to intelligent educational agents that support self-paced, adaptive learning. The AI Study Buddy project builds upon these innovations to deliver an interactive, web-based platform that leverages AI to simplify study materials and generate personalized quizzes.

## 1.2 Problem Statement

Students often face challenges when trying to comprehend and retain large volumes of academic content, especially under time constraints. Existing learning platforms either lack intelligent automation or fail to combine content simplification with testing and feedback. There is a clear need for an integrated tool that not only helps students understand complex topics quickly but also reinforces learning through practice.

## 1.3 Objectives of the Project

The main objective of this project is to develop an intelligent web-based educational assistant that helps students learn more efficiently and interactively. The specific goals are:

To design a frontend that allows easy uploading and viewing of course materials.

To implement AI services for automatic summarization, explanation, and quiz generation.

To support multiple types of assessments such as multiple-choice quizzes.

To enable real-time performance tracking and feedback.

To ensure a responsive user experience using asynchronous backend processing.

## 1.4 Scope and Limitations

This project focuses on course material in text format and supports summarization and quiz generation in English. The system is optimized for web use and is not intended for mobile applications in its current version. Voice input and OCR features are implemented

While the platform uses a fine-tuned Gemma model for NLP tasks, the quality of summarization and quiz generation depends on the complexity and clarity of the input content. Additionally, the quiz generation is limited to factual and conceptual questions and may not handle highly technical or mathematical problems.

## 1.5 Report Organization

This report is organized into five chapters:

*Chapter 1:* Introduction – Describes the motivation, problem statement, goals, scope, and structure of the report.

*Chapter 2:* Literature Review – Discusses prior work related to AI in education, learning agents, and existing tools.

*Chapter 3:* System Design – Presents the architectural, frontend, backend, and database design of the platform.

*Chapter 4:* Implementation and Results – Details the implementation of core components, performance metrics, and user interaction.

*Chapter 5:* Conclusion and Future Work – Summarizes the project outcomes and proposes enhancements for future development.

1. Literature Review

## 2.1 Overview of AI in Education

## Artificial Intelligence (AI) is increasingly transforming the education landscape by offering personalized, adaptive, and efficient tools for both students and educators. Technologies such as machine learning, natural language processing (NLP), and computer vision have enabled the development of intelligent systems capable of simplifying content, generating assessments, and providing instant feedback.

## AI’s role in education is particularly valuable for processing large volumes of content, automating learning tasks, and improving engagement. With advancements in Optical Character Recognition (OCR), AI tools can now extract text from images—expanding their usefulness beyond digital documents.

## 2.2 AI Agents for Learning Support

AI agents are autonomous programs designed to perceive input, reason over it, and act to achieve a defined goal. In educational contexts, AI agents are used to assist learners by automating tasks like summarization, question generation, and feedback.

Types of AI agents used in learning tools include:

Reactive agents – Respond directly to user input.

Deliberative agents – Analyze input and plan intelligent responses.

Learning agents – Improve their responses over time with experience.

AI Study Buddy functions as a deliberative educational agent. It processes PDF-based course content and performs three key functions:

Summarizes academic material to simplify complex information.

Generates MCQs and creates interactive quizzes to reinforce understanding.

Answers questions from users based on the uploaded material.

This makes the agent a multi-functional study companion that supports comprehension, practice, and engagement—all in one environment.

## 2.3 Existing Educational Platforms

Several platforms offer AI-enhanced educational support, such as:

Quizlet – Helps users generate flashcards and quizzes.

Socratic by Google – Answers student questions using NLP and search.

Notion AI – Can summarize notes and help answer prompts.

However, these tools often require manual input, are limited in scope, or do not support PDF document processing directly. In contrast, AI Study Buddy is specifically designed to extract, understand, and process academic content directly from PDF uploads.

## 2.4 Comparative Analysis of Solutions

While many platforms offer OCR or question generation, few integrate all the following into a single platform:

* Automated **OCR from images**
* **Summarization** of extracted content
* **Question generation** (MCQ, T/F)
* **Real-time Q&A** from user-uploaded academic materials

**AI Study Buddy** distinguishes itself by combining these features into one focused, web-based tool that helps students turn raw materials (PDFs, photos of notes, textbook images) into personalized learning sessions.

## 2.5 Summary of Research Gaps

Despite growing use of AI in education, several important gaps persist:

* Few tools offer **end-to-end learning automation** from **images or scanned content**.
* Limited platforms integrate **OCR, summarization, and quiz generation** in one workflow.
* Most AI systems lack the ability to generate **context-aware questions** from **user-provided content**.
* Real-time **content-driven Q&A** is rarely supported in lightweight educational platforms.

**AI Study Buddy** addresses these issues by acting as a multi-functional educational AI agent that accepts both text and image input, simplifies content, generates quizzes, and supports Q&A—making it a unique and comprehensive tool for modern students.

1. System Architecture

## Introduction:

In this chapter, we present the technical and structural design of the AI Study Buddy system. This includes an overview of the architecture, frontend and backend components, the AI services powering summarization and question generation, as well as the database schema and user interactions.

We also include relevant diagrams—such as system architecture, flowcharts, and use case diagrams—to better illustrate the system’s workflow and modular design. Each section breaks down a specific part of the platform, providing clarity on how various components communicate and function together to deliver a seamless AI-assisted learning experience.

## System Overview:

## The architecture has four key components:

## **Frontend:** Handles file upload, OCR for images, and PDF text extraction using client-side libraries (e.g., Tesseract.js for OCR, pdf.js for PDFs). It sends extracted plain text to backend APIs.

## **Backend:** Receives text input and invokes AI services (Gemma model) for summarization, quiz generation, and question answering.

## **AI Services:** Perform NLP tasks on text received from frontend.

## **Database:** Stores user info, course materials.

## By running OCR and extraction on the frontend, the backend avoids file parsing, focusing on AI processing, which improves system efficiency.

## Front-End Design:

## The frontend performs:

## **OCR on images** using Tesseract.js to extract text locally.

## **PDF text extraction** using pdf.js or similar libraries to parse and extract text client-side.

## Sending extracted plain text to backend for AI processing.

## Displaying summaries, quizzes, and Q&A responses from backend.

## User authentication and navigation via React.js and TailwindCSS.

## This design reduces server load and latency, enabling faster feedback.

## 

## Backend Design:

## Backend receives clean extracted text, avoiding complex file parsing.

## Handles AI model calls for summarization, quiz generation, and question answering.

## Manages database operations and user sessions.

## Provides APIs for frontend communication.

## Async processing ensures the frontend stays responsive.

## 3.5 Database Schema Design:

## The database is implemented using **MySQL**, structured with the following main tables:

## **Users:** Stores user credentials and profile data.

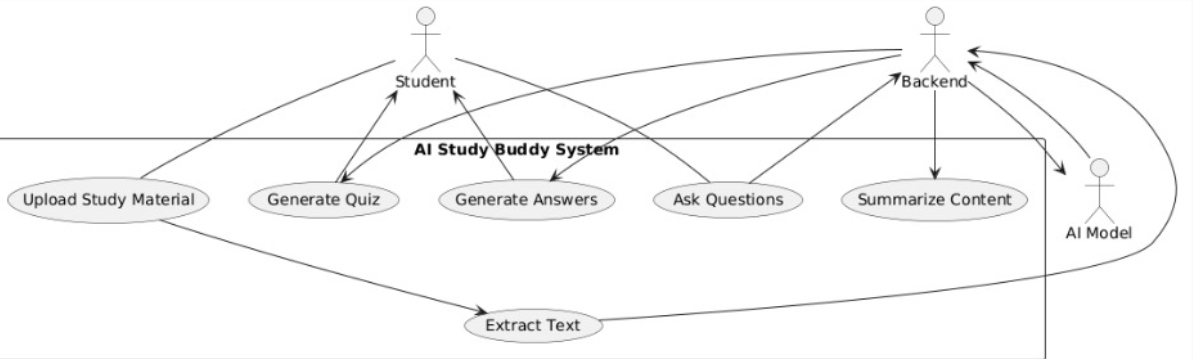
## **CourseMaterials:** Stores uploaded files metadata and extracted text.

## 

## User Flow and Use Cases:

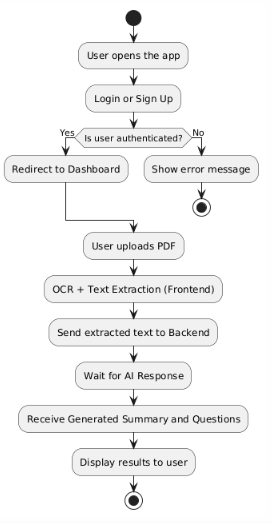
* + 1. Use Case Diagram:

The following use case diagram represents the core interactions users have with the AI Study Buddy system.

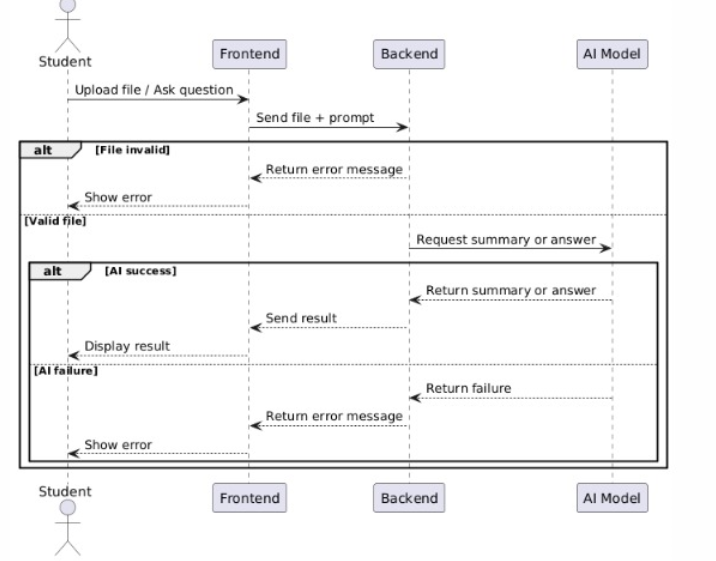


* + 1. User Flow Diagram:

User Flow from login to receiving generated content after uploading a document.



* + 1. Sequence Diagram:  
       The sequence diagram details the internal communication flow between frontend, backend, and AI services during a typical interaction.



1. Implementation & results:

## 4.1 Implementation Tools:

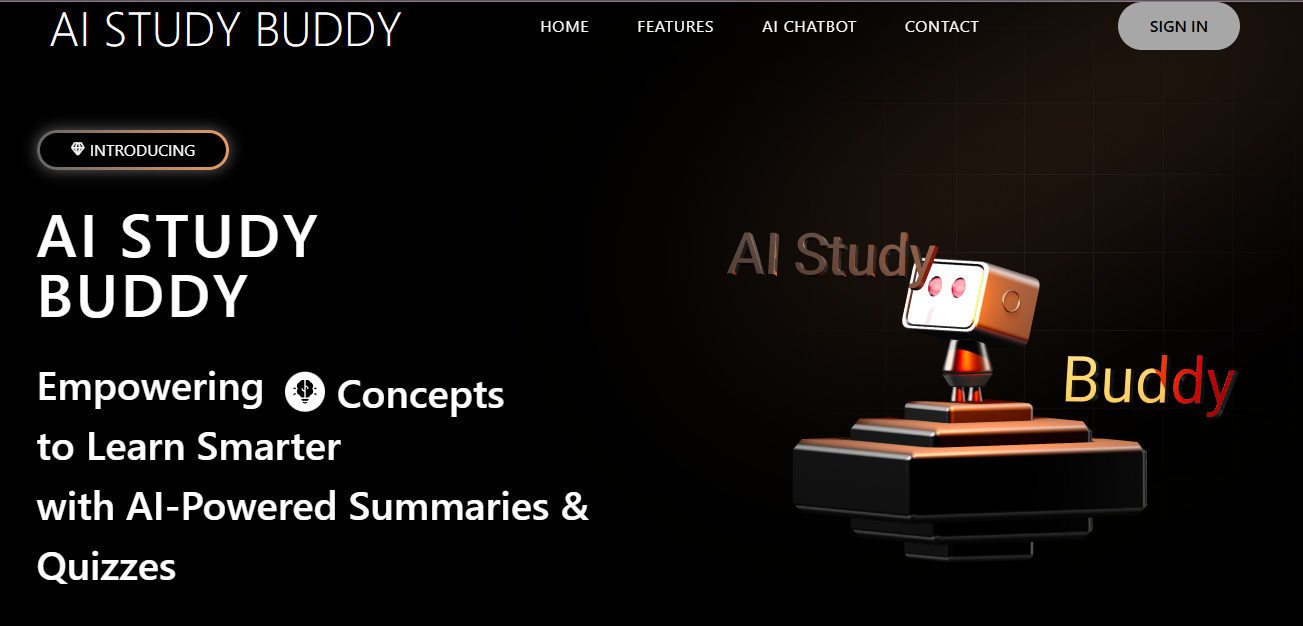
This section lists and explains the primary tools and technologies utilized to develop AI Study Buddy. It covers the frontend, backend, AI frameworks, database systems, and supporting libraries.

* **Frontend:** React.js, Tailwind CSS, Tesseract.js (OCR), pdf.js
* **Backend:** Python, FastAPI, Uvicorn, SQLAlchemy
* **AI Model:** Gemma (fine-tuned LLM), integrated via API or local deployment
* **Database:** MySQL for storing users, course materials.
* **Others:** Axios for HTTP requests, JWT for authentication, Git for version control

## 4.2 Frontend Implementation:

## The frontend is built using React.js with Tailwind CSS for clean and responsive design. It consists of several key pages that together form a seamless user experience:

## **Landing Page:** Introduces the AI Study Buddy, highlights key features, and provides navigation to sign up or log in.



## A screenshot of a computer AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## A hand holding a phone AI-generated content may be incorrect.

## A computer desk with a chair and a computer AI-generated content may be incorrect.

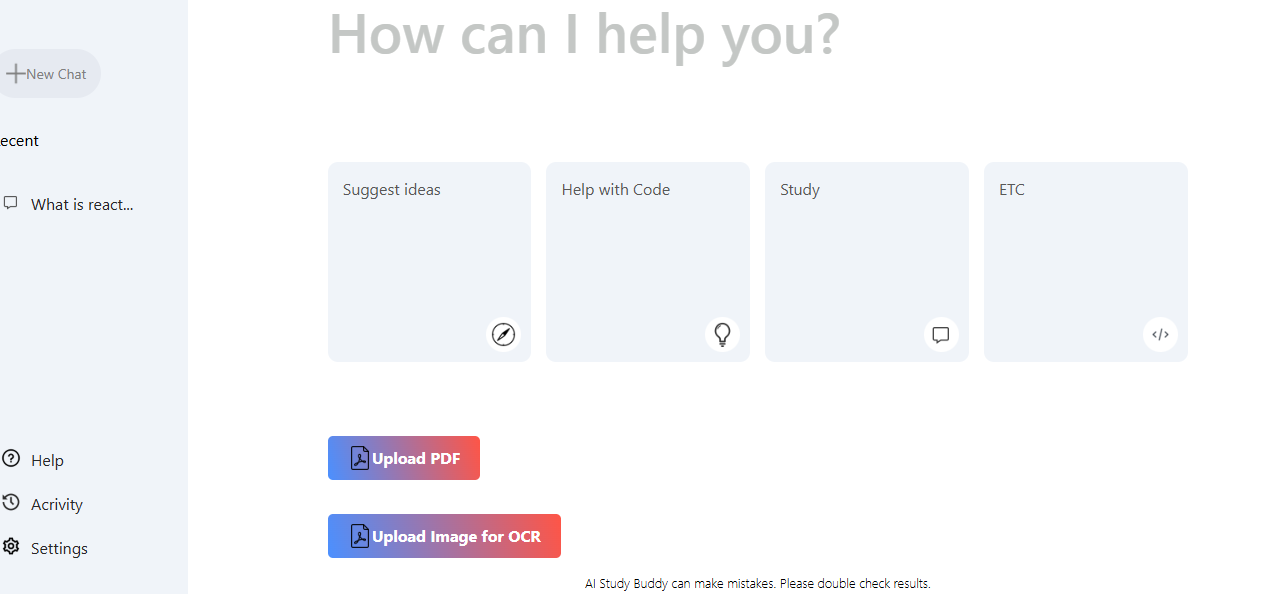
## **Signup/Login Page:** Allows users to create accounts , enabling personalized learning experiences, responsive design with nice animations.

A screenshot of a login form

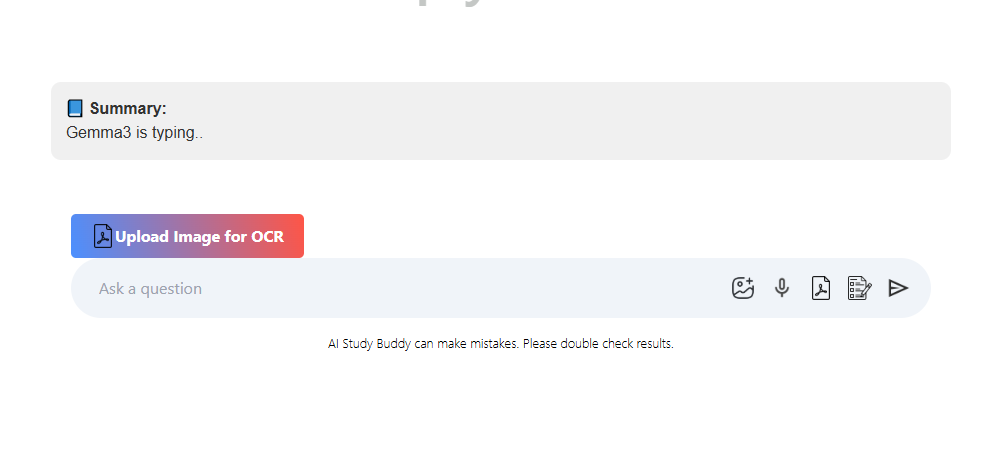
AI-generated content may be incorrect.A screenshot of a login form

AI-generated content may be incorrect.

## **Chatbot Page:** The main interactive hub where users upload PDFs or images, trigger OCR and text extraction, receive AI-generated summaries, quizzes, and can ask questions. This page integrates the quiz interface and real-time chat-based question answering.



When uploading a pdf an input with appear to allow users to ask for questions,and start the quiz



*Some code snippets:*

*This the mainApp react component that is rendered directly on index.html page through main.js :*export default function MainApp() {

  return (

    <Router>

      <Routes>

        <Route path="/" element={<App/>} />

        <Route path="/chatbot" element={<ChatbotPage />} />

        <Route path="/signin" element={<SignUp/>}/>

        <Route path="/quiz" element={<Quiz/>}/>

      </Routes>

    </Router>

  );

}

*In this section we used react-router-dom and made several routes each that renders a specific component*

*The App.jsx renders the landing page:* return (

   <main>

    <img  src="/gradient.png" className="absolute top-0 right-0 opacity-60 -z-1" alt="GradientImage"/>

    <div className="h-0 w-[40rem] absolute top-[20%] right-[-5%] shadow-[0\_0\_900px\_20px\_#e99b63] -rotate-[30deg] -z-10" ></div>

    <Header/>

    <Hero/>

    <Showcase/>

    <LogoShowCase/>

    <FeatureCards/>

    <Experience/>

    <TechStack/>

    <Testimonials/>

    <Contact/>

    <Footer/>

   </main>

  )

*Each component contains a section in the landing page tailwindcss is used for animation and styling*

*The full code is provided through the github repository*

*The signup/signin page is rendered through SignUp component:  
we used Formik library to create forms users can apply, and we used eventhandlers on change and on submit*

const handleSignUpClick = () => {

    setAnimationClass("");

    setTimeout(() => {

      setAnimationClass("animate-signUp");

      setActiveFormClass("sign-up-active");

    }, 10);

  };

  const handleSignInClick = () => {

    setAnimationClass("");

    setTimeout(() => {

      setAnimationClass("animate-signIn");

      setActiveFormClass("sign-in-active");

    }, 10);

  };

  useEffect(() => {

    if (animationClass) {

      const timeout = setTimeout(() => {

        setAnimationClass("");

      }, 1600);

      return () => clearTimeout(timeout);

    }

  }, [animationClass]);

  // Validation schema

  const SignUpSchema = Yup.object().shape({

    name: Yup.string().required("Required"),

    email: Yup.string().email("Invalid email").required("Required"),

    password: Yup.string().min(6, "Too short").required("Required"),

    confirmPassword: Yup.string()

      .oneOf([Yup.ref("password")], "Passwords must match")

      .required("Required"),

  });

  const LoginSchema = Yup.object().shape({

    email: Yup.string().email("Invalid email").required("Required"),

    password: Yup.string().required("Required"),

  });

*In the mainchat a button is used to upload pdfs and pdfWorker library is used to extract text from the pdf, another button is used to upload an image and Tesseract from 'tesseract.js' is used for ocr to extract text from the image*

*Now the text is sent to the backend, if the text is from a pdf we add to the text “summarize” , while if it its from a image ,or text ,or voice , “answer” is added to the text:*const handleOcrFileChange = async (e) => {

    const file = e.target.files[0];

    if (!file) return;

    setOcrLoading(true); setOcrError(""); setSummary(""); setAiReply(""); setUserPrompt("");

    setIsTyping(true);

    const imageUrl = URL.createObjectURL(file);

    try {

      const result = await Tesseract.recognize(imageUrl, "eng");

      const extractedText = result.data.text.trim();

      console.log("✅ OCR Extracted Text:", extractedText);

## Client-side OCR is performed on images using Tesseract.js, and PDFs are processed with pdf.js before sending extracted text to the backend. React Router manages navigation between pages, while Axios handles API communication.

## 4.3 Backend and API Integration:

## The backend uses FastAPI to create RESTful APIs that handle:

## Receiving extracted text from frontend

## Calling AI services for summarization, quiz generation, and Q&A

## Managing user authentication and session management

## Persisting data to MySQL using SQLAlchemy ORM

## Asynchronous handling ensures non-blocking API responses.

from fastapi import FastAPI, HTTPException

from fastapi.middleware.cors import CORSMiddleware

from pydantic import BaseModel

import mysql.connector

app = FastAPI()

# Enable CORS for your frontend (adjust origin as needed)

app.add\_middleware(

    CORSMiddleware,

    allow\_origins=["http://localhost:5173"],

    allow\_credentials=True,

    allow\_methods=["\*"],

    allow\_headers=["\*"],

)

def get\_db\_connection():

    return mysql.connector.connect(

        host="localhost",

        user="root",

        password="",

        database="ai\_study\_buddy"

    )

# Pydantic models

class User(BaseModel):

    username: str

    password: str

    email: str

class LoginInput(BaseModel):

    username: str

    password: str

class CourseMaterial(BaseModel):

    title: str

    content: str

    user\_id: int

# Route to add course material

@app.post("/course\_material/")

def create\_course\_material(course\_material: CourseMaterial):

    conn = get\_db\_connection()

    cursor = conn.cursor(dictionary=True)

    try:

        # Insert course material into the database

        cursor.execute(

            "INSERT INTO coursematerials (title, content, user\_id) VALUES (%s, %s, %s)",

            (course\_material.title, course\_material.content, course\_material.user\_id),

        )

        conn.commit()

A screenshot of a computer

AI-generated content may be incorrect.

## A computer screen with text on it AI-generated content may be incorrect.

## 4.4 AI Model Integration:

## The AI integration revolves around the Gemma large language model:

## The backend sends the extracted text to the Gemma model for summarization and quiz generation.

## Responses are parsed and formatted for frontend consumption.

## The system also handles user questions by sending prompts to the AI model and returning context-aware answers.

* @app.post("/chat")
* async def chat\_with\_summary\_and\_mcqs(query: Query):
* try:
* prompt = query.prompt.strip()
* context = query.context.strip()
* async with httpx.AsyncClient(timeout=1000) as client:
* # === CASE 1: Just answer the question ===
* if prompt.lower().startswith("answer this:"):
* print("🤖 Detected direct Q&A")
* clean\_question = prompt[len("answer this:"):].strip()
* answer\_prompt = (
* "You are a helpful AI assistant. Answer the following question clearly and briefly:\n\n"
* f"Q: {clean\_question}\nA:"
* )
* response = await client.post(
* "http://localhost:11434/api/generate",
* json={
* "model": "gemma3",
* "prompt": answer\_prompt,
* "stream": False
* }
* )
* response.raise\_for\_status()
* data = response.json()
* answer = data.get("response", "").strip()
* return {"answer": answer}
* # === CASE 2: Summarize + MCQ generation ===
* elif prompt.lower().startswith("summarize this:"):
* print("📄 Detected summarization request")
* chunks = chunk\_text(context)
* summary\_instruction = (
* "You are an expert summarizer and a dedicated teacher who wants your students to truly understand and succeed.\n"
* "Summarize the following content by focusing only on the main ideas and key concepts. "
* "Avoid unnecessary details. Present the summary clearly and in simple language:\n\n"
* )

A prompt is provided to set rules for the model

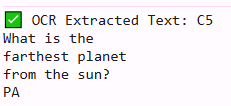
The model operates at localhost:11434

## 4.5 Key features in action:

## This section demonstrates the main features through example use cases:

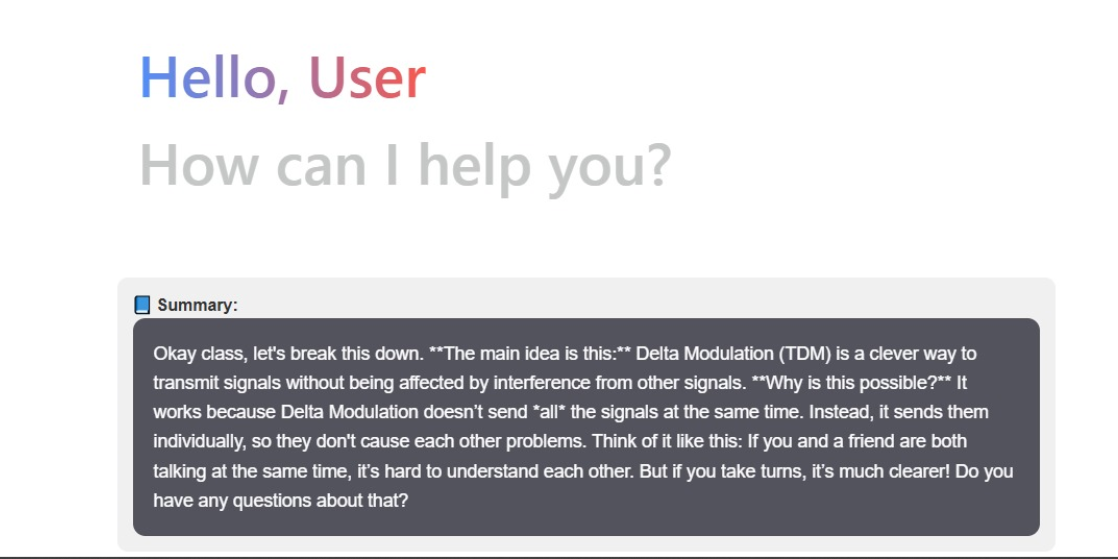
## Uploading an image and obtaining OCR text

We used this image to test the ocr, we printed the output on the console

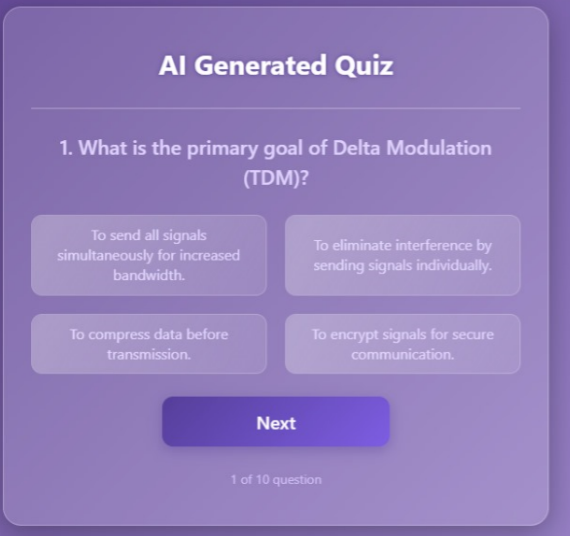


## Uploading a PDF and receiving a summary

We uploaded a DMT-modulation small pdf,



## Taking an automatically generated MCQ quiz

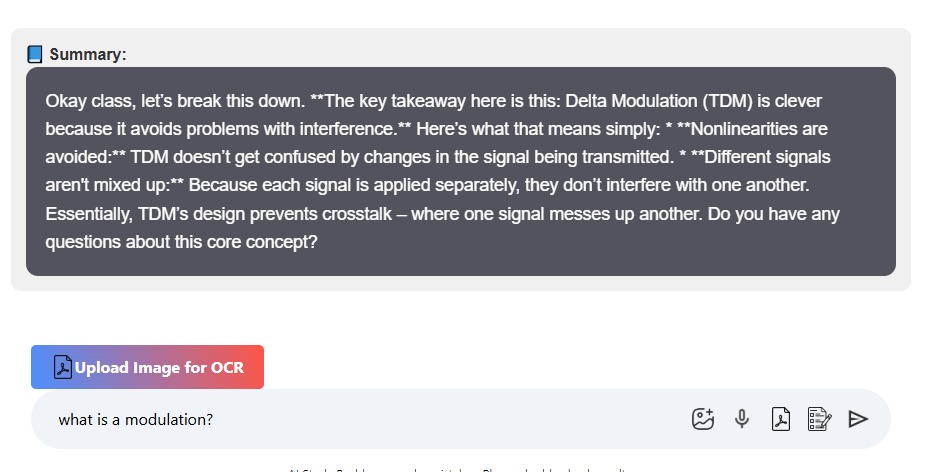
A screenshot of a quiz

AI-generated content may be incorrect.

## 

If the score is less than 6 the Ai regenerates the explanation

## Asking a question and receiving AI-generated answers



## 4.6 Testing and Evaluation:

## The AI Study Buddy project was successfully completed and thoroughly tested across its core functionalities. Each component—OCR, text extraction, summarization, quiz generation, and chatbot Q&A—was evaluated to ensure smooth interaction and accurate responses.

## ✅ **Frontend Testing:** Verified that the user interface flows correctly across the landing page, login/signup, and chatbot page. File uploads, quiz display, and chat responses all functioned as intended.

## ✅ **Backend/API Testing:** Ensured all endpoints returned expected results. The system correctly handled PDF/image processing, AI model calls, and data persistence.

## ✅ **AI Model Validation:** Sample educational PDFs were used to test summarization accuracy, relevance of multiple-choice questions, and clarity of chatbot answers.

## ✅ **Edge Case Handling:** Special cases (e.g., empty uploads, large files, unusual formatting) were tested to confirm graceful handling without crashes.

## ✅ **Cross-Browser & Device Testing:** Verified responsiveness and functionality on major browsers and screen sizes.

## Overall, the project met all the initial goals and delivered a functional, intelligent learning assistant platform for students.

1. Conclusion:  
   5.1 Summary of Achievements:

The *AI Study Buddy* project was successfully developed as an intelligent assistant to help students study more efficiently. Key accomplishments include:

* A user-friendly **landing page**, **sign-up/login**, and **chatbot page**.
* Integration of **OCR** and **text extraction** from PDF/image files.
* An AI-powered backend capable of generating **summaries**, **multiple-choice questions**, and **contextual answers** through a chatbot.
* Clean frontend implementation using **React** and a reliable backend with **FastApi** and **Python** AI services.
* Successful deployment, testing, and demonstration of all key features.

This platform demonstrates the practical use of AI in simplifying academic content and improving study methods.

## 5.2 Challenges Faced:

## Throughout the project, several challenges were encountered:

## Handling file uploads and ensuring compatibility between **PDFs and images**.

## Managing **CORS issues** and proper API integration across frontend and backend.

## Ensuring **fast response times** for AI-generated content.

## Addressing bugs in **text formatting** and edge cases in image-to-text conversion.

## Despite these, teamwork and iterative debugging helped us overcome each obstacle effectively.

## 5.3 Limitations:

While the system performs well, a few limitations remain:

* Limited support for **handwritten text** in images.
* Summaries and questions may vary slightly in relevance depending on **text structure**.
* The chatbot is focused on academic questions and may not perform well with **general topics**.
* The platform is currently designed for **single-user interaction** without role-based access or history tracking.
* Due to hardware constraints, especially using a **low-specification laptop**, we were unable to download and run larger, more advanced AI models. As a result, we relied on lightweight models which may slightly impact performance or accuracy in some cases.

## 5.4 Future Enhancements:

To further improve the platform, the following enhancements are planned:

* **Support for handwritten OCR** using more advanced models.
* Enable **user history**, saved quizzes, and performance analytics.
* Add a **voice input feature** for accessibility and ease of use.
* Expand to support more languages and subjects.
* Integrate with **learning management systems (LMS)** like Moodle or Google Classroom.
* Add admin/moderator dashboards and multi-user roles.

## 

## 