A Retrieval-Augmented based Agentic Chatbot

by

Examination Roll: 232220

A Project Report submitted to the
Institute of Information Technology
in partial fulfillment of the requirements for the degree of
Professional Masters in Information Technology

Supervisor: Dr. M Shamim Kaiser, Professor

Institute of Information Technology
Jahangirnagar University
Savar, Dhaka-1342

December 2020

DECLARATION

I hereby declare that this project is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This project, neither in whole nor in part, has been previously submitted for any degree.

Roll:232220

CERTIFICATE

The project titled “A Retrieval-Augmented based Agentic Chatbot” submitted by Student-Md. Arafat Hossain, ID: 232220, Batch: 29, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Professional Masters in Information Technology on the 17th of September 2025.

Dr. Shamim Al Mamun
Supervisor

ACKNOWLEDGEMENTS

I want to thank each and every person who has given me the opportunity to make this project successful. My thanks to everyone for standing by me with patience and encouragement while I have worked on this project. Their confidence in me has meant so much and inspired me every step of the way.

In the very first instance, I would like to express my deepest gratitude to our supervisor, Prof. Shamim Al Mamun, Professor, PhD, Institute of Information Technology (IIT), Jahangirnagar University, who provided constant assistance, valuable suggestions, and generous support to my work. His constant support, valuable suggestions and the supply of resources have played a very vital role in conducting this project in the specific time period.

I also would like to express my deep appreciation to the institute director of IIT, who guided me throughout and at some point even encouraged me to see this work through in time. I also wish to thank faculty members of IIT who have helped me directly or indirectly during the period of this project.

I sincerely appreciate any other support, without which I could not have successfully managed to complete this project.

IIT staff and friends should be thanked for the cooperation and helpfulness with which I have managed to complete this journey.

ABSTRACT

This project is completely based on the web and promotes the Retrieval-Augmented Generation (RAG) model to help users upload, manage, and search educational resources such as PDFs and previous years' question papers. By incorporating Artificial Intelligence (AI) and Natural Language Processing (NLP) technologies, the system can realize user questions and arrange clear, accurate, and context-based answers to the questions. This makes the learning process more interactive and effective, with a chatbot interface enabling users to have natural conversations while retrieving information from uploaded documents.

React and Vite are used to implement the front-end of the system. These two offer a smooth and user-friendly interface for users. These are used to sign in securely, navigate through various sections, and view files. The backend is developed in Node.js and Express. Use Node.js and Express to manage data storage, file uploads, and application logic in a well-organized and scalable way. Cloudinary is being integrated into this project because of safe file storage, while a database is being used to manage the documents and the corresponding metadata in an efficient way.

By leveraging the power of the PDF content in conjunction with a Large Language Model (LLM) using RAG, the system can create meaningful and context-aware responses to user queries. It also has essential features, like robust error handling, a way of protecting access to certain pages and an intelligent search function. Mainly, the given project provides a highly efficient and modernized system of educational resource management. This project can help users find queries from PDF. Users can upload their PDF and ask questions from it. This project delivers the answer to the users, which available in the PDF.

Here is the GitHub link: https://github.com/Arafat-shuvo/jiggasa
And the weblink: https://my-ask.onrender.com/

Keywords: Retrieval-Augmented Generation, Natural Language Processing, Chatbot, Artificial Intelligence and Large Language Model.

LIST OF ABBREVIATIONS

IIT	Institute of Information Technology
JU	Jahangirnagar University
RAG	Retrieval-Augmented Generation
AI	Artificial Intelligence
LLM	Large Language Model
PDF	Portable Document Format
NLP	Natural Language Processing
DB	Database
JS	JavaScript
HyDe	Hypothetical Document Embedding
CRAG	Corrective RAG
DB	Database
HTML	Hyper Text Markup Language
UI	User Interface

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER I

Introduction

1.1 Overview

This report is about how to design and run a Retrieval-Augmented -based Agentic Chatbot[1], where the user can upload PDF files and get response to the user's queries in an intelligent way by using the Gemini AI. Here choosing PDF files for thier wide uses.[2] The system is designed to provide smart response creation with precision, scalability, and friendly interaction with the users.

1.2 Motivation

In the age of Artificial Intelligence , there are lot of information, but user don't get it in a structured way and therefore individuals and organizations struggle to find and comprehend the relevant information within a short time period. The education sector, the healthcare sector, the legal sector and the corporate world are among many industries that use PDF documents to store crucial information, reports and research papers. Nonetheless, it may take time and be ineffective and full of errors to use these documents manually.[3]

The main reeeason behind this project was to close the gap existing between data storage and smart information retrieval. Older types of search, such as key word searches, can be irrelevant and cannot interpret context. As Artificial Intelligence (AI) develops at an alarming rate, it is possible to improve the document management system, allowing it to offer exploratory responses instead of just keyword searches.

I intend to develop a system in which users will be able to:

Upload documents easily,

Automate process and organize of content, and

The correct and AI-based response to questions is achieved by asking natural language questions.

This project will simplify the process of PDF document search and make it faster, smarter, and more reliable through the combination of Natural Language Processing (NLP) and Gemini AI. Students, researchers, and offices that use a large number of PDFs find it especially useful as it saves their time and allows them to concentrate on making decisions instead of searching for them manually.

In brief, this project was driven by the idea of turning the non-living document storage into a living system of knowledge, where one can not only store information but also retrieve it immediately and smartly understand it. With this innovation, I want to enable users to have a smooth, productive and smart experience in dealing with their information.

1.3 Problem Statement

In most industries and organizations, PDF documents are the most common types of form to store valuable information, such as research papers, reports, policies, manuals, and historical records. These documents hold valuable data, retrieval of a particular information in a quick and accurate manner is a big challenge.

Basic keyword searches and other traditional ways of searching do not always provide contextually relevant search results. Users have to waste time reading long documents to get the information they want, which are:

Long, particularly when using big document sets.

Slow because when keywords are typed in the search form, irrelevant or unfinished answers might be retrieved.

Prone to errors, which has the effect of missed or misinterpreted information.

The worst part is that the number of PDF documents keeps increasing. Organizational search and retrieval require more than just a basic search feature that only searches and pulls back what a user requests but rather searches and provides the most important information automatically.

Moreover, current systems do not have intelligent support with present AI technology to support natural language query processing and context-based responses. This brings the query pattern by humans and the information retrieval pattern by machines apart.

Core Problem

No system exists that is efficient, AI-driven and lets users:

Use a lot of PDF documents.

Extract, organize and save text information automatically to use it later.

Ask natural language questions and get specific answers, rooted in context, without manual search.

Project Goal

For issue, this project suggests a Retrieval-Augmented Generation (RAG) system that is consists of:

NLP text extraction and text keyword identification techniques,

MongoDB to store in an organized and effective manner, and

Gemini AI of smart and context-aware response generation.

This solution can save time, require less manually work, and be more accessible to users in other fields, such as education, research, and business; streamline the process of document search and make it more intelligent, fast, and reliable.

1.4 Objective

To create a chatbot is the main intention of this project. To make a study software that allows users to post learning resources, including textbooks, lecture notes, and guides, and get correct, context-sensitive answers to their queries based on Retrieval-Augmented Generation (RAG)[4] methods. This platform try to make the study process more efficient, faster, and smarter. for learners and students. This platform provide a deeper understanding and conceptual learning, not just data retrieval. The system has combined Natural Language Processing (NLP) with Gemini AI.[5]

Specific goals of this project that should be mentioned:

1. To provide a better learning experience.
2. To provide authentic and interactive feedback on learners' questions from their own PDF is the goal of creating this AI-powered chatbot. Their uploaded PDFs are a sound source of documents.
3. To reduce hallucinations and fake information by using artificial intelligence is the goal of this project.

This proposed model also answers the following questions:

1. What are the major challenges that have faced in RAG?
2. Why users use this?
3. What are the best way to retrieval the information?

1.5 Research Outline

Rest of the report is structured as follows: In **Chapter II** a literature study is presented in this chapter, and explanations of the key terms that will be used in this project. The basic ideas behind RAG and the different kinds of it are also discussed in detail. **Chapter III** introduces the system model, including the data flow and flowchart of the working procedure of the entire system model. **Chapter IV** explains the details of project development process, folder, raw codes and file structure. **Chapter V** discusses about the result, user guide,and the performance of this platform. Lastly, in **Chapter VI** Limitation and the conclusion are mentioned.

CHAPTER II

Literature Review

2.1 Related Work

2.1.1 RAG

RAG merges information retrieval and generative models to reduce hallucinations and to keep replies grounded in documents. RAG enhancements expand to query rewriting, multi-hop retrieval, and iterative RAG that learns when and what to retrieve.[6] The Retrieval- Augmented Generation (RAG) is a methods which make artificial intelligence (AI) systems more smarter and more accurate. Think about a typical AI such as a student who has read a great number of books before and is not able to read new books now. When the student want to know more specific then he/she may guess or even provide the incorrect answer. This situation mimics the principle of normal AI models, since they can only proceed in accordance with the information they were trained to proceed, and cannot adjust to new verity.

The AI to search to answer questions is enabling by RAG. It is like a student who does not just remember that he/she studied something but reviews the most recent books and then answer the questions. This will help student to get up-to-date and relevant.

RAG system is whereby when user ask a question, the AI goes through the stored documents or databases to locate useful information. It then combines that with what it already has as a way to make a transparent and fact-based response. This would minimize errors, or hallucinations. Naive Rag used in this project.

2.2 RAG Types

2.3 Retrieval-Augmented Generation Classifiers

RAG is an AI algorithm that involves information retrieval, which can generate text and allow a model to generate facts. and context-aware answers. RAG finds a great deal of application in document. question answering, knowledge management, chatbots and search.

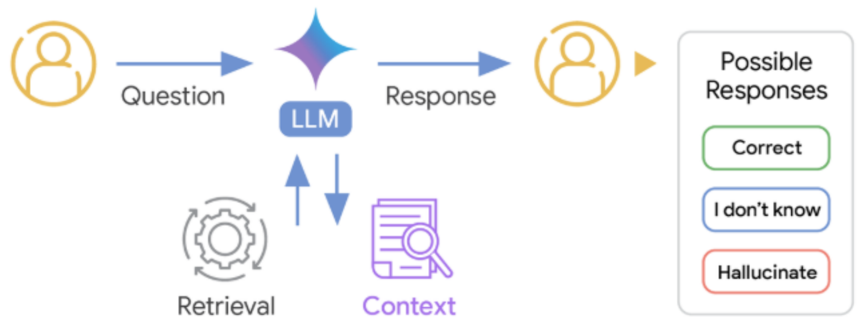


Figure 2.1: Rag Model

There are several types of RAG architectures based on how retrieval and generation are integrated:

2.3.1 Simple RAG

This version of RAG consists a language model that retrieves relevant documents from a static database, in response to a query, and provides an output based on the information that has been retrieved. It is a very simple perspective. Which is good for small database. And it does not need to process large or dynamic data. Here a user enters a question, and the model queries the fixed database to get the related information. The model produces a response based on real-world information. Simple RAG is especially used in the applications like frequently asked questions systems or customer support chatbots, where the response has to be accurate and fact related, but the available information is constrained to a known set of documents. Basically it will answer from a product manual or internal storage. [7]

2.3.2 Simple RAG with Memory

It is an expansion of the simple RAG model to include a storage module, which allow the model to remember information from the past interactions. This type of improvement increase the effiecncy of the system. It will helpful for continuous interactions. Workflow will start with a user's query or prompt. This model uses this query to access stored memory and retrieve applicable past interactions. Then it goes out to the external database and retrieves any additional relevant information and produces an answer from the retrieved documents with the stored memory. It is especially useful with chatbot. It may help with customer services where the current dialogue with the customer will use the information about the customer's choice or about previous problems. It used past information for provide a more accurate answer.

2.3.3 Branched RAG

Branched RAG is much more flexible and efficient. When user's query is presented, the system finds the right sources which saves times. Then it extracts documents of the right source. Then ut creats a response based on the same information. Particularly it applies to the processing of complex queries that require expert knowledge, like legal use or in multidisciplinary studies.[8]

2.3.4 HyDe (Hypothetical Document Embedding)

It is a unique RAG type. It creates hypothetical documents according to the query and then retrieves the information. Firstly, it will create an ideal document from the prompt. This hypothetical document helps the retriveal process. It helps to increase relevancy and quality of the results. The user then enters the query. Then the hypothetical document is generated, upon which real documents in a knowledge base are retrieved. Lastly, the model produces output based on not only the documents retrieved but also on the hypothetical representation. HyDe is particularly useful in research and development (R&D), where queries may not be transparent and searching data based upon positive answers helps refine convoluted answers. It can also be used to create creative content.[9]

2.3.5 Adaptive RAG

It is an adaptive version of RAG varies the retrieval strategy. It can switch in diffenrt action at any time. It can retrieve single source documents in simple queries and multiple source documents or use sophisticated retrieval methods in more complex queries. When the user providing a prompt, if there is adaptive retrieval, in which the model executes the best retrieval strategy. Then it will digest the information retrieved and produce a query-optimized response. Adaptive RAG is particularly powerful within an enterprise search framework.[10]

2.3.6 Corrective RAG (CRAG)

Corrective RAG Corrective RAG or CRAG extends earlier RAG models by adding a self-reflection or self-grading aspect to the responses that can be generated to achieve accuracy and relevance. In contrast to normal RAG, CRAG takes a critical look at the quality of retrieved documents before going to generation. Workflow consists of a user query, document retrieval and evaluation. The documents retrieved are sorted into smaller units of information called knowledge strips, which are also rated in terms of relevance. Unrelevant strips are filtered and when no strip passes a relevance test, the system adds more retrieval stages to complement the information including web searches. After getting a satisfactory set of knowledge strips, the model creates a response using the most capable and correct content. CRAG tends to be more useful in methods where a high level of factual accuracy is necessary (e.g. legal document generation, medical diagnostic support, or financial analysis) and even small errors may be extremely damaging. The accuracy of retrieval of CRAG and its area consistency make the function of CRAG measure reliable in delicate domains.

2.3.7 Self-RAG

To enable the model to automatically create retrieval queries during the content generation process, Self-RAG provides a self-retrieval mechanism. Contrary to old-fashioned RAG models, which only seek information based on the information entered by the user, Self-RAG can also refine its queries as it generates content. The process begins when the user gives a query in the form of a prompt, and then some preliminary searches are made of documents in response to the query. In the course of generation, the model detects the gaps in the information and makes new retrieval queries to enrich the content. It is an iterative self-retrieval process that helps to ensure that the resulting response is more extensive, and broader. Self-RAG works especially well when the research is exploratory or the content is long-form and it is better to dynamically add more information to the output to make it more accurate and complete. [11]

2.3.8 Agentic RAG

The agentic RAG adds an agentic behavior to the retrieval and generation process that allows the model to engage in complex and multi-step tasks by actively interacting with multiple data sources or APIs. The most significant characteristic of the Agentic RAG is that Document Agents attached to specific documents are defined and integrated by a Meta-Actor, which integrates interactions and selects the most sufficient retrieval strategies based on the intricacy of a query. The user enters a multi-faceted query and the workflow starts with the initiation of multiple document agents that summarize and answer a query about their assigned document. The Meta-Agent is in charge of these interactions, it synthesizes the results, and it produces a full and coherent answer. This is especially useful in applications like automated research, aggregation of multi-source data, or executive decision support, where the model has to gather, analyse and synthesise information across multiple systems independently.[12]

2.4 Artificial Intelligence

By handling digital documents like PDFs, Artificial Intelligence (AI) is becoming a revolutionary technology.

Among the many digital formats, the Portable Document Format (PDF) is one of the most universal formats, widely used in the educational, medical, legal, research and corporate documents. These PDF files often store important information such as policies, reports, contracts, user manuals, academic papers. Hence, extracting relevant information from PDFs is a manual process, which is time-consuming and can result in inefficiencies and errors.[13]

In future AI will help and automate the process and make the process simple. And make the process not only faster but also more accurate and efficient. AI uses the method called Natural Language Processing (NLP) after removing the text from PDFs. Because of explaining and clarifying the circumstance and meaining the things that it is contained. In the future, AI will be the part and parcel of our daily life. It will be used in healthcare, legal, finance, economic, corporate and educational sector. Alan Turing's AI is now a great technology now.[14] For example, in healthcare, healthcare AI algorithms can analyze patient reports to automatically identify important medical terms, diagnoses, and treatment details.[15] Likewise, in the judicial sector, artificial intelligence will be able to recognize clauses, contrasts, and specialized legal terms in contracts more effectively and help users to navigate and read legal documents better.

Moreover, AI plays an important part in structuring and categorizing PDFs. Thousands of files with many different formats find in large organizations. Which is very difficult to organize. AI systems can be used to characterize PDFs automatically based on their subject. This automation lets businesses have well-structured digital libraries. Which can quickly isolated documents and confirm that there is continuous management of their organizational data.

2.5 Natural Language Processing

Natural Language Processing (NLP) plays a fundamental role in modern document management[16], especially in the-Be Extraction of data from PDF documents. PDF documents are one of the world's most widely used digital files. In PDF format, [17]documents are considered a preferred format because of the device and platform independent ability to maintain the layout and structure of information. They are extensively used in various industries from the educational and healthcare fields to legal services, corporate business industries, and research and development. An example of this is how educational institutions use PDFs to store research papers, assignments and question banks. Healthcare organizations deploy PDFs for patients' medical records, medical diagnosis reports, prescribing and treatment history. In the legal industry, important papers like contracts, agreements and even case files are stored in PDF files so as to sustain standardization and facilitate easy sharing. Corporate businesses use PDFs for financial reports, employee policies, invoices, and documentation regarding compliance. Similarly, researchers use PDFs to communicate and publish their results, publish research, and save references. This widespread use speaks volumes on how integral PDFs is to dealing with a large amount of critical information in multiple domains.

2.6 Research Gap

While Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs) have proven to have a lot of potential in engaging in improved information retrieval and AI-driven systems, there are still quite a number of gaps, especially in the situation of educational resource management systems. Current solutions provided in this space mainly focus on providing document search and storage functionalities but do not even focus on constructing an intelligent response that is context-specific based on a complex query provided by a user.

Most current educational platforms are operating via keyword-based search mechanisms, which often return irrelevant or incomplete results: inefficiency and user frustration are the result. Although in recent years we have seen future advances in Natural Language Processing (NLP) [18] and AI-powered Chatbots that have brought conversational interfaces, they often make the following sufferings:

AI creates the wrong or fake information. Which is called hallucination.[19]

AI responses aren't directly linked to sound source documents.

Getting enhanced learning experiences from AI is difficult.

This project works to fill these gaps by:

Integrating RAG with a backend to enable accurate context and also enable question answering.

Intelligent responses, versus simple keyword matches, using AI and NLP to provide an intelligent response.

Introducing a chatbot interface for interactive learning and retrieval,

Implementing file handling and storage using access control

Ensuring performance and reliability with modern development practices.

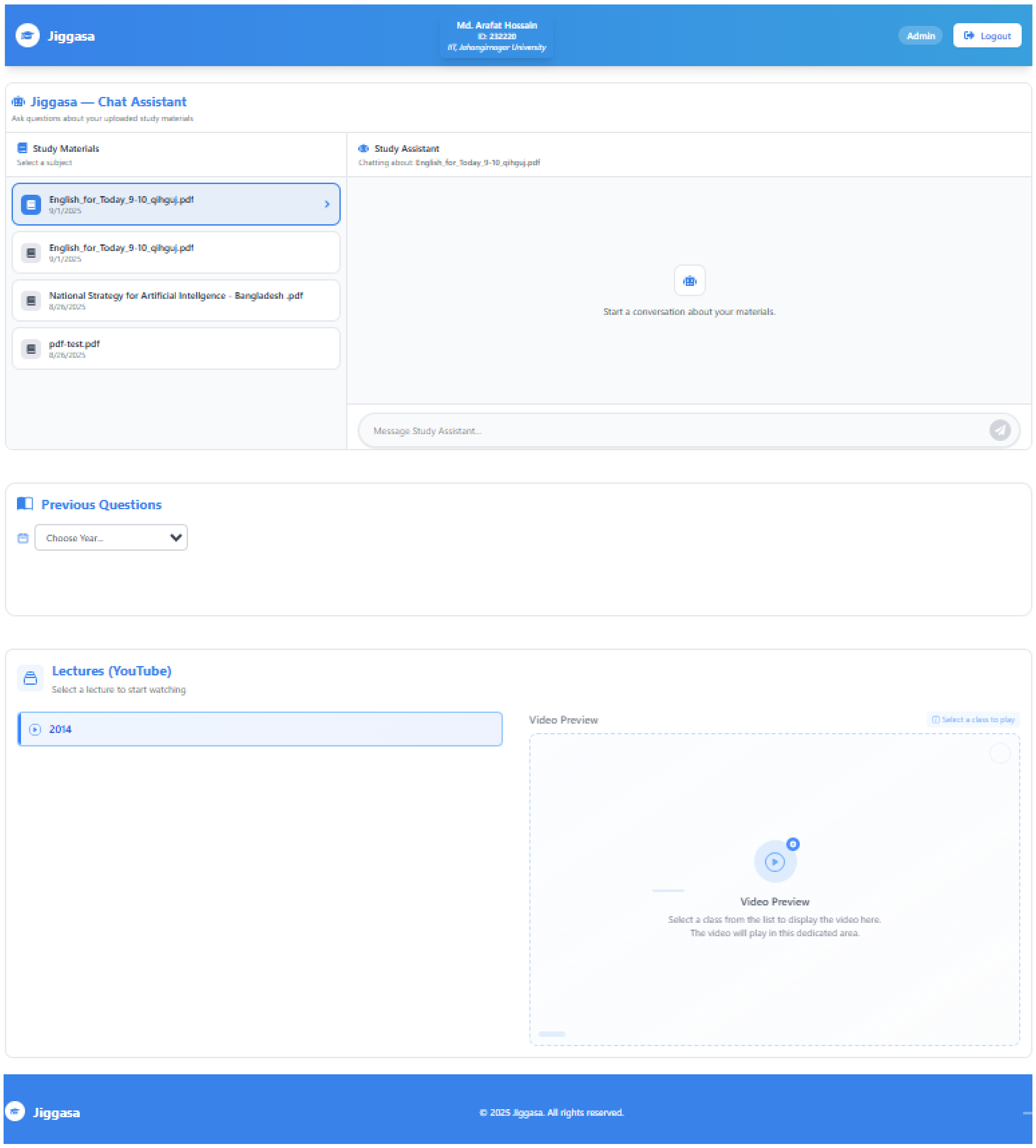

Jiggasa
Md. Arafat Hossain
ID: 232220
IIT, Jahangirnagar University
Admin
Logout
Jiggasa — Chat Assistant
Ask questions about your uploaded study materials
Study Materials
Select a subject
English_for_Today_9-10_qihguj.pdf
9/1/2025
English_for_Today_9-10_qihguj.pdf
9/1/2025
National Strategy for Artificial Intelligence - Bangladesh .pdf
8/26/2025
pdf-test.pdf
8/26/2025
Study Assistant
Chatting about: English_for_Today_9-10_qihguj.pdf
Start a conversation about your materials.
Message Study Assistant...
Previous Questions
Choose Year...
Lectures (YouTube)
Select a lecture to start watching
2014
Video Preview
Select a class to play
Video Preview
Select a class from the list to display the video here.
The video will play in this dedicated area.
Jiggasa
© 2025 Jiggasa. All rights reserved.

Figure 2.2: Interface

CHAPTER III

System Model

3.1 Discussion about System

The goal of this chatbot is to answer questions from the uploaded PDF. Normally, an individual tries to find out the answer from the PDF manually, but it is not that easy to get the correct answer from the PDF or documents. Although it is laborious and not easy to navigate. Sometimes people need the answer very fast, but in manually it is time-consuming. It is more efficient and accurate than a human. With this chatbot, users can simply upload their PDFs and ask in natural language. The usability of this chatbot is in the educational sector, research, the industrial sector and individual usage. Google gemini 2.5 Flash as an AI agent, MongoDB as a database, React for the frontend are used here.

Firstly, user upload their PDF. These PDFs are stored securely. Then the system creates chunks from the PDF. These optimized chunks are stored in MongoDB. During which time a user asks a question then the system tokenize the question. And the system analyzes the quieries. And match the question with the selected PDF. Then it returns a selected text. Finally, sent the question and the selected text send to AI. Then AI provides a better answer.

figure 3.1 shows, there is a flow of uploading a PDF and user questions.

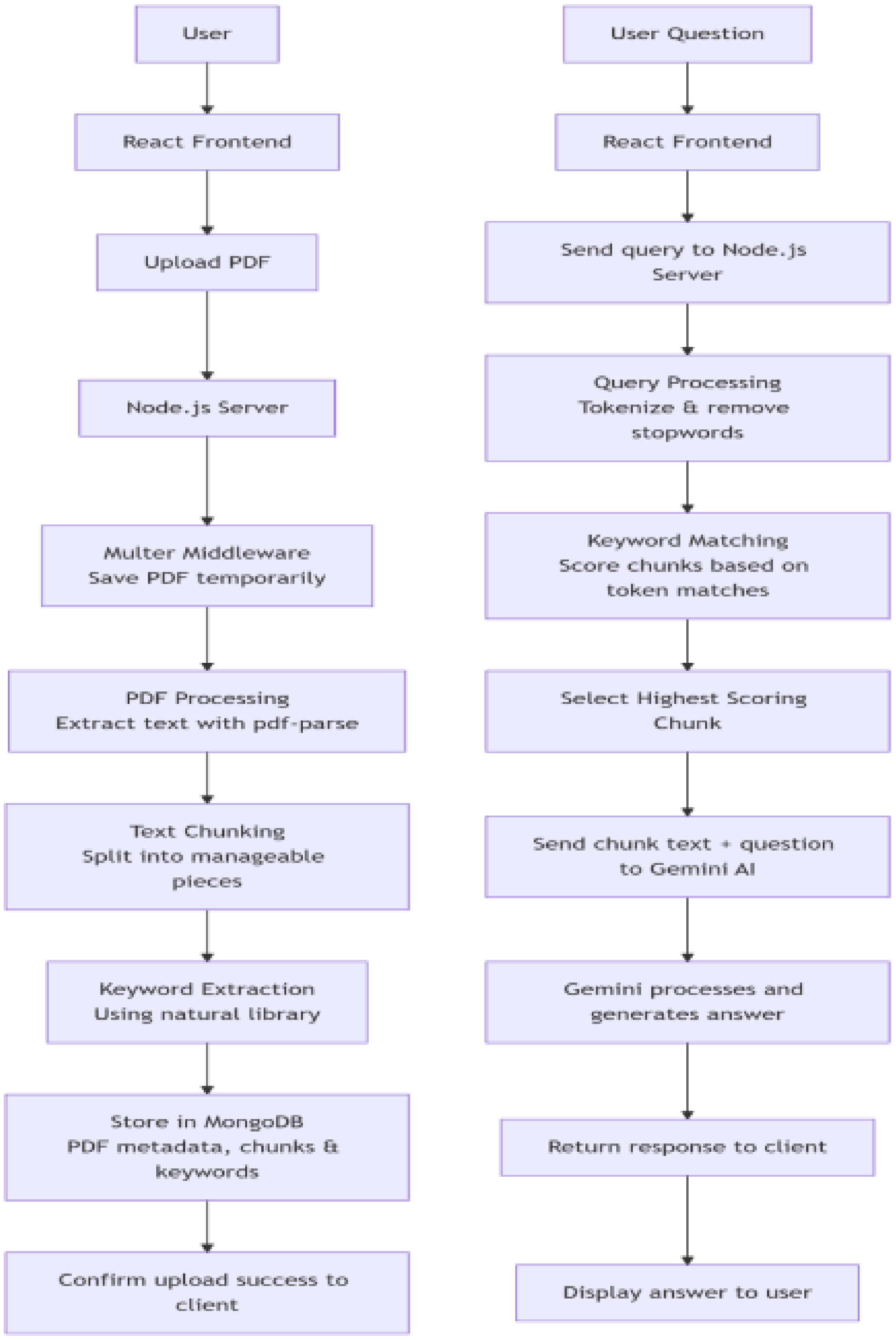

User
React Frontend
Upload PDF
Node.js Server
Multer Middleware Save PDF temporarily
PDF Processing Extract text with pdf-parse
Text Chunking Split into manageable pieces
Keyword Extraction Using natural library
Store in MongoDB PDF metadata, chunks & keywords
Confirm upload success to client
User Question
React Frontend
Send query to Node.js Server
Query Processing Tokenize & remove stopwords
Keyword Matching Score chunks based on token matches
Select Highest Scoring Chunk
Send chunk text + question to Gemini AI
Gemini processes and generates answer
Return response to client
Display answer to user

Figure 3.1: Data Flow

3.2 Model

The user upload PDF in frontend React web interface. Which is simple and user friendly. The React frontend sends files to the server via HTTP request. After uploading the PDF, user asked a question related to the PDF content. Then the frontend send the question with the PDF ID to the server. Then backend server processes the uploaded PDF with PDF Parse. Then the text will be chunked and keywords extracted. Then the query will be processed. In this processing, MongoDB search the best chunk matched with the question. After getting relevant chunks, these chunks and questions sent to Gemini AI. Then Gemini AI undedrstand the question and relavant chunks and generate a context-based answer. The answer will be generated from the chunks of PDF strictly. After processing by AI, then the answer send to the server. Then the server sent it to the React frontend for the user. And it is the final response.

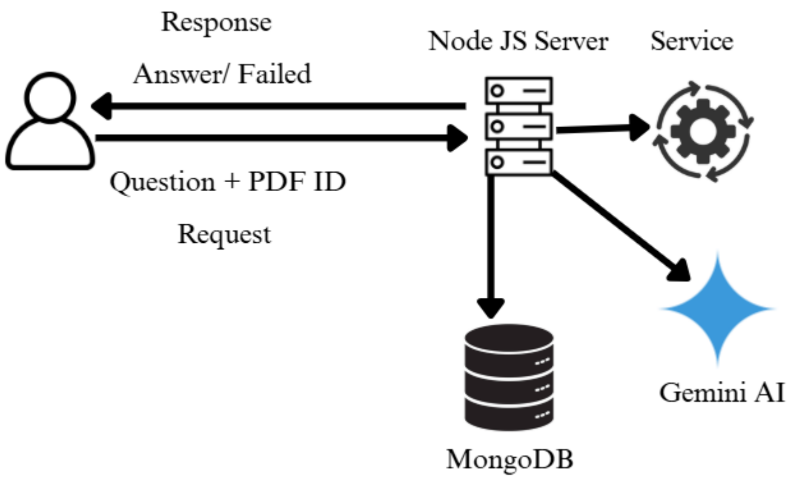


Figure 3.2: Model

3.3 Achievement of Objective

The Chunk size will be optimize for getting the objective of this project. This will lead to improved retrieval accuracy.
Increase keyword length size for a better match. This will improve the document search.
Also optimize the database query.
Gemini AI instruction improvement. Reduce hallucination by using improved instruction sets.

3.4 Research Saturation

Artificial Intelligence (AI) and Natural Language Processing (NLP) have notably transformed the way users interact with digital content in recent times. The Retrieval-Augmented Generation (RAG) is new phenomenon in this arae, Which can retrieval information from documents. And it is very influential in providng highly contextual and error-free answers.The combination of Large Language Models (LLMs), and Google Gemini Flash 2.5 can provide human-like responses.

Create a context-related answer. This chatbot understand the user's document and reply with an accurate answer based on PDFs.
Sometimes AI generates misinformation and deliver doutful reply. In this system, remove hallucination by providing PDF documents. Make the answer factual and prevent distortion.
Get an intelligent response by using Gemini Flash 2.5. Gemini reads the chunks, understand the questions and generate clear answers.

Implement MongoDB database for file handling and storage.

CHAPTER IV

Project Development Process

4.1 Server Setup

Firstly, select node.js for the server because of its JavaScript runtime. It is best for the request to be handle. It can handle 10,000 concurrent requests. Moreover, Node.js handle asynchronous requests and fullfill non-blocking I/O.
As per document:
Primarily, Node.js and npm install in local machine
then the Related package "npm"
Write "npm init" command in command line interface.
Then "package.json" file will be auto generated.

MongoDB Atlas used in this project as a database.[20] Make a collection for this project. Because:
Simple way of deployment
Easily managable database
Highly scalable

Cloudinary account created:
Here, create a account via gmail for store files.

Folder as per this project requirement:

4.1.1 Config Folder

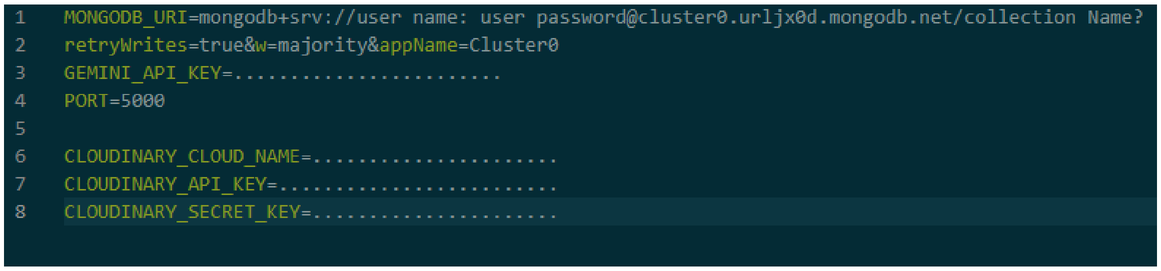
Helping APIs for this project will find in this folder. After auto generate the "package.json," then create config folder with "db.js," "cloudinary.js", files. MongoDB config file will find in "db.js," and the Mongodb connecting string configures here. And the Cloudinary connecting string configures here.

In "db.js" file, import "mongoose," and environment variables load from the "dot env" file. In the "connectDB" function use "mongoose.connect()" to create a connection with "MongoDB". All kinds of MongoDB connection events will find here. For example: connected, error, disconnected, shutdown.

For connected, "Mongoose connected to DB" will show in console log
For error, "Mongoose connection error" will apear in console log
For disconnected, "Mongoose disconnected from DB" will show in the console log

Here, use "export" for exporting "connectDB" and "mongoose" for using in other files.

In "cloudinary.js" import Cloudinary Software Development Kit version 2 and environment variables load from the "dot env" file. Got cloudinary configuration setup code from cloudinary documentation. In the Cloudinary configuration setup, provide cloud name, API key, API secret key. And make cloudinary as a object to use in other files.



```
MONGODB_URI=mongodb+srv://user name: user password@cluster0.urljx0d.mongodb.net/collection Name?
retryWrites=true&w=majority&appName=Cluster0
GEMINI_API_KEY=.......................
PORT=5000

CLOUDINARY_CLOUD_NAME=.....................
CLOUDINARY_API_KEY=........................
CLOUDINARY_SECRET_KEY=.....................
```

Figure 4.1: .env File

4.1.2 Module

Create a modules folder after config folder had created. All data blue print will be found in this folder. In this folder, there are two files "pdf.js' and "previous.js." Create PDF schema here.

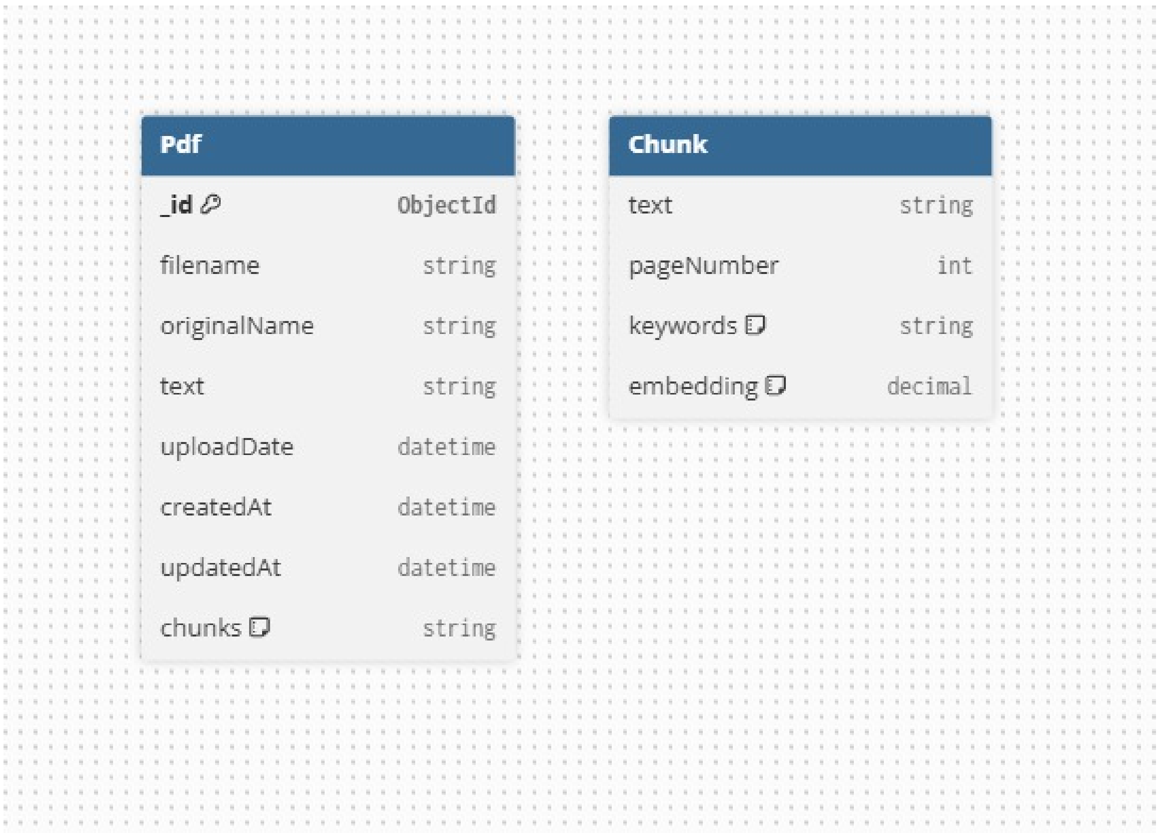
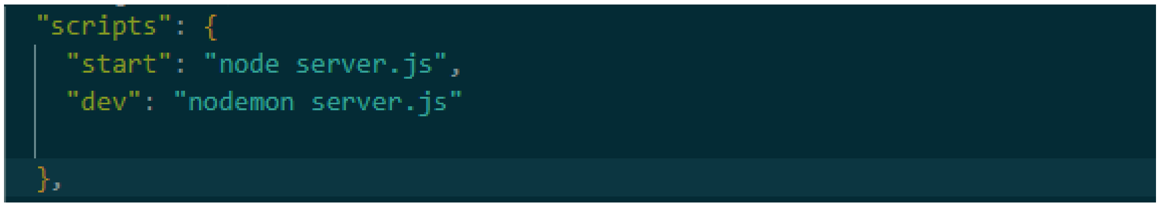


Figure 4.2: PDF Schema

4.1.3 App.js and server.js

After modules folder, create app.js and server.js file. Need to check the database connected or not. For this reason, project run command set up in the package.json script. Then install all project related dependencies. For this install all dependencies. Write: "npm i". Then it will auto-generate "package-lock.json" and "node modules" then setup gitignore. Here use command line code:
npm start / npm run dev then get the knowledge of database connected or not.

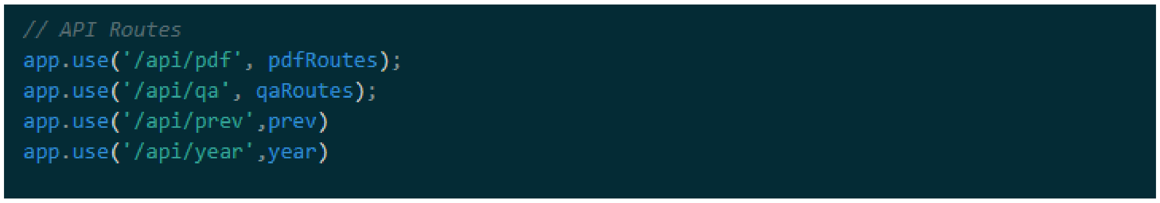


```
"scripts": {
  "start": "node server.js",
  "dev": "nodemon server.js"

},
```

Figure 4.3: Script

4.1.4 Config other API set

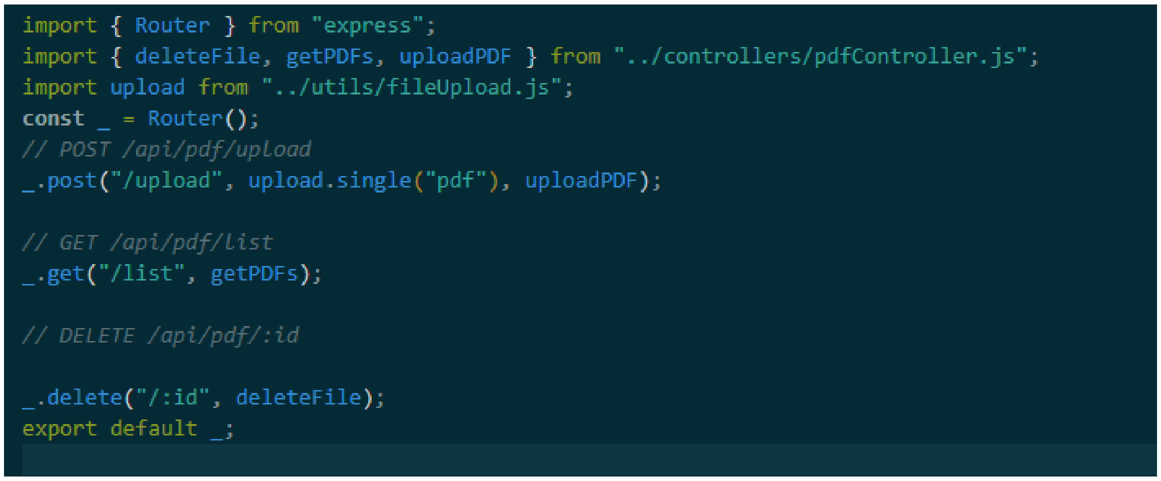
API request received in apps.js. And set end points in API Routes. Setup the routes for redirecting from routes to the controller. Create a controller folder to set control flow. This project have four controllers:
PDF Controller
Previous Controller
Question Answer Controller
Year Controller



```
// API Routes
app.use('/api/pdf', pdfRoutes);
app.use('/api/qa', qaRoutes);
app.use('/api/prev',prev)
app.use('/api/year',year)
```

Figure 4.4: API Routes

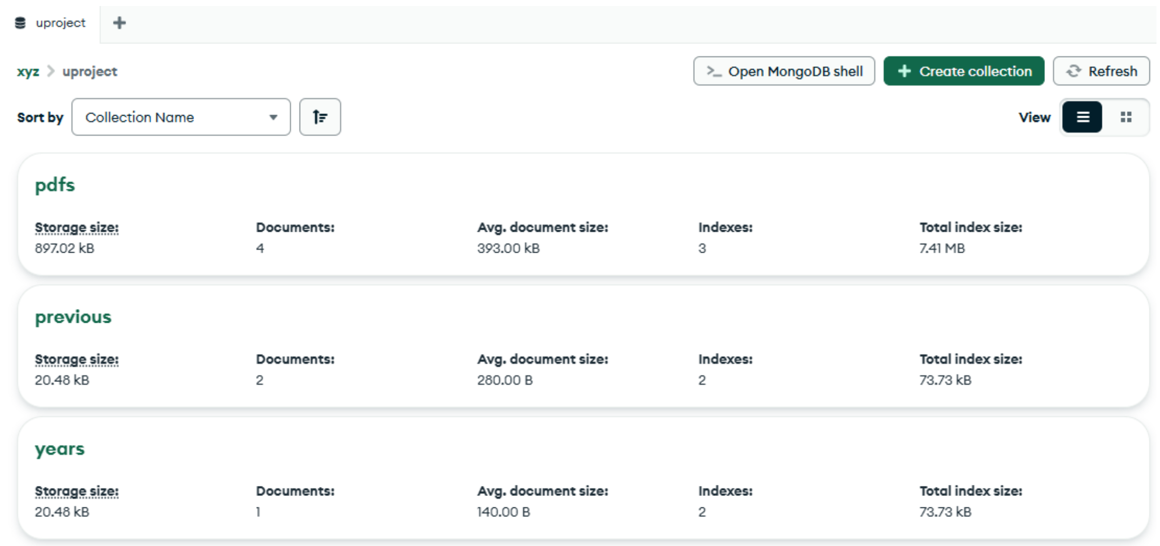
Then setup the error controller for error handling. Which located in the middleware folder.
In the PDF controller, sentences from the PDF are extracted into chunks. Each chunk has text, a page number and keywords. Here, three function used: upload PDF, get PDF, and delete PDF. It also import and extract keywords and chunks text from the utils folder. And import PDF from "Pdf.js." This "Pdf.js" located in modules. It also import extract text from PDF among "pdfService.js." In "uploadPDF" function it extract keyword chunks where chunk size is 5. Select this chunk size for time management challenges.



```
import { Router } from "express";
import { deleteFile, getPDFs, uploadPDF } from "../controllers/pdfController.js";
import upload from "../utils/fileUpload.js";
const _ = Router();
// POST /api/pdf/upload
_.post("/upload", upload.single("pdf"), uploadPDF);

// GET /api/pdf/list
_.get("/list", getPDFs);

// DELETE /api/pdf/:id

_.delete("/:id", deleteFile);
export default _;
```

Figure 4.5: PDF Controller

In this section, used the MongoDB PDF model to store PDFs. In the "getPDFs" function all PDFs are gotten from the MongoDB. The delete function is used to delete PDFs from the Database.



pdfs

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
897.02 kB	4	393.00 kB	3	7.41 MB

previous

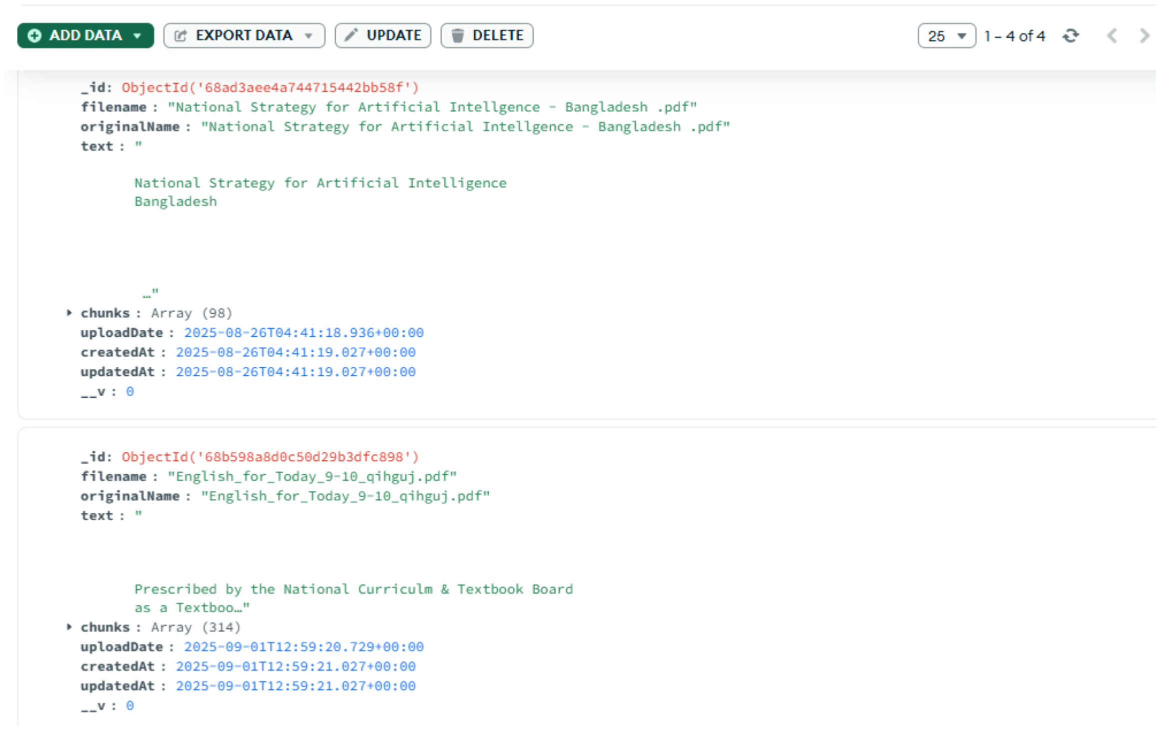
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	2	280.00 B	2	73.73 kB

years

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	1	140.00 B	2	73.73 kB

Figure 4.6: MongoDB Database

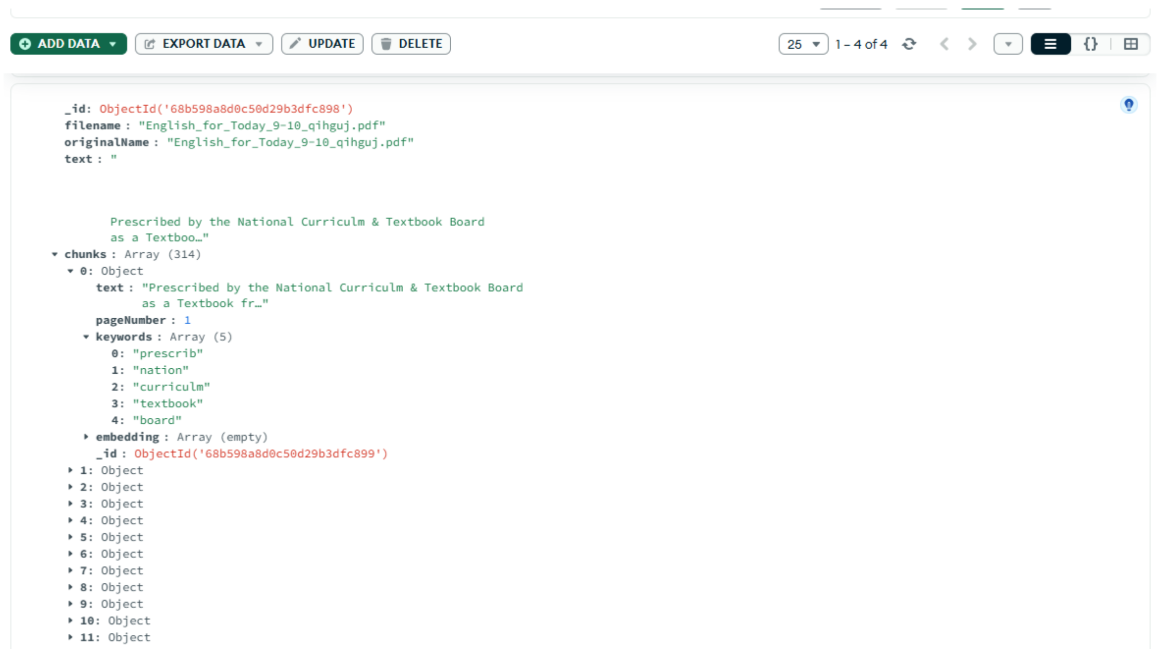
In this figure, got the view of the MongoDB database.



```
_id: ObjectId('68ad3aee4a744715442bb58f')
filename : "National Strategy for Artificial Intellgence - Bangladesh .pdf"
originalName : "National Strategy for Artificial Intellgence - Bangladesh .pdf"
text : "

       National Strategy for Artificial Intelligence
       Bangladesh



        …"
▸ chunks : Array (98)
uploadDate : 2025-08-26T04:41:18.936+00:00
createdAt : 2025-08-26T04:41:19.027+00:00
updatedAt : 2025-08-26T04:41:19.027+00:00
__v : 0

_id: ObjectId('68b598a8d0c50d29b3dfc898')
filename : "English_for_Today_9-10_qihguj.pdf"
originalName : "English_for_Today_9-10_qihguj.pdf"
text : "


       Prescribed by the National Curriculm & Textbook Board
       as a Textboo…"
▸ chunks : Array (314)
uploadDate : 2025-09-01T12:59:20.729+00:00
createdAt : 2025-09-01T12:59:21.027+00:00
updatedAt : 2025-09-01T12:59:21.027+00:00
__v : 0
```

Figure 4.7: Files in MongoDB

In this figure, got the view of the file in the MongoDB database. It also shows the PDF ID, number of chunks, and upload date.



```
_id: ObjectId('68b598a8d0c50d29b3dfc898')
filename : "English_for_Today_9-10_qihguj.pdf"
originalName : "English_for_Today_9-10_qihguj.pdf"
text : "


      Prescribed by the National Curriculm & Textbook Board
      as a Textboo…"
▾ chunks : Array (314)
  ▾ 0: Object
      text : "Prescribed by the National Curriculm & Textbook Board
            as a Textbook fr…"
      pageNumber : 1
    ▾ keywords : Array (5)
        0: "prescrib"
        1: "nation"
        2: "curriculm"
        3: "textbook"
        4: "board"
    ▸ embedding : Array (empty)
      _id : ObjectId('68b598a8d0c50d29b3dfc899')
  ▸ 1: Object
  ▸ 2: Object
  ▸ 3: Object
  ▸ 4: Object
  ▸ 5: Object
  ▸ 6: Object
  ▸ 7: Object
  ▸ 8: Object
  ▸ 9: Object
  ▸ 10: Object
  ▸ 11: Object
```

Figure 4.8: Keywords in Chunks

This figure shows keywords in every chunks. Each chunks contains 5 keywords. We can change the number of keywords.

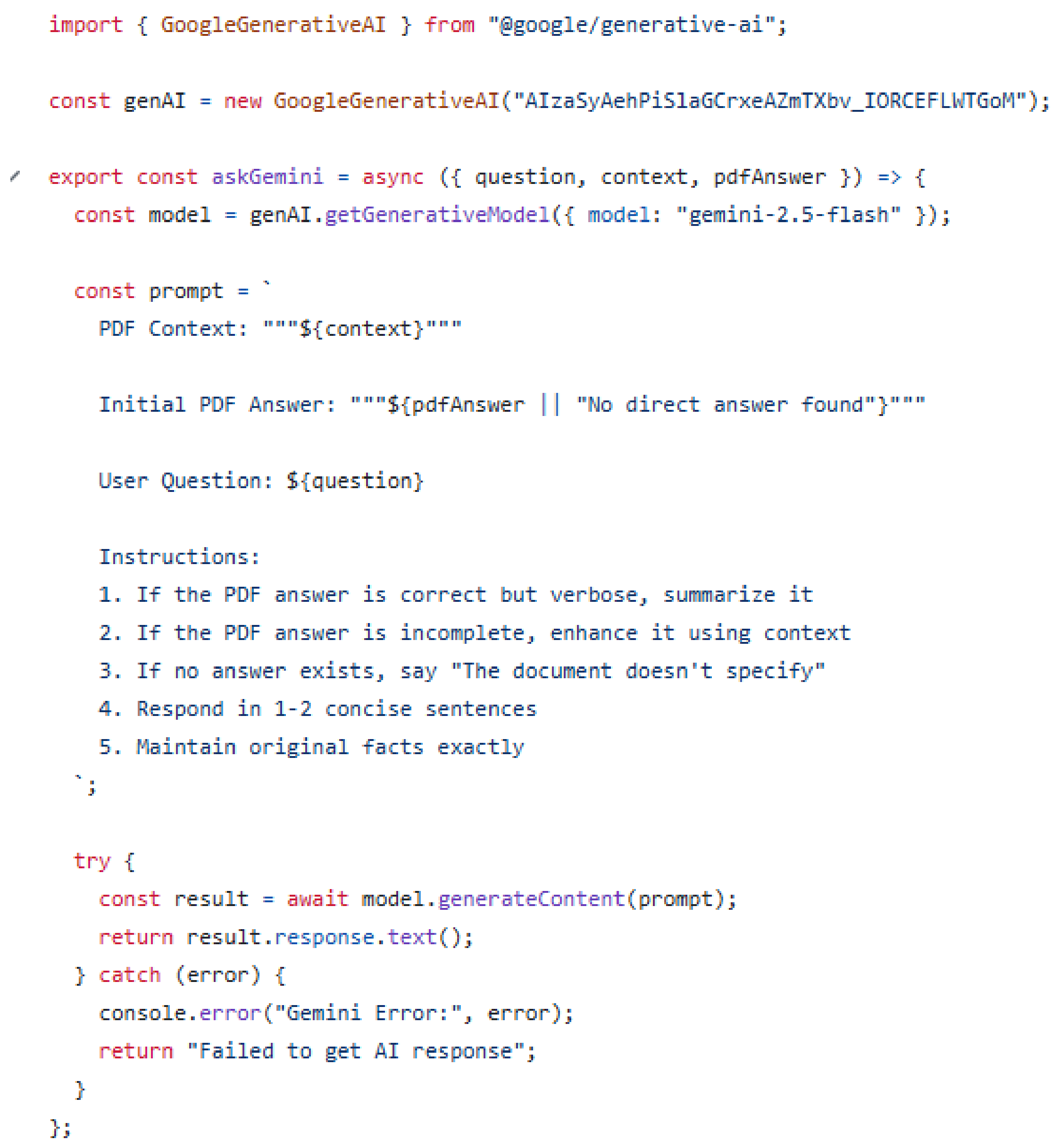
After working with PDF controller, Start working with the question answer controller. In this project, the name of that file is "qaController.js." Which is available in the controller folder. In this file, there is a function named "answerFromPDF" and two helper functions named "extractKeywords" and "calculateRelevanceScore." In the PDF controller, the question and PDF ID had retrieved. It will search the PDF in DB with its PDF ID. It also tokenize the question. For tokenization, it omits some words. For example: "the," "and," "of," "to," "a," "in," "is," "it," "that," "this," "for," "with," "as," "on," "was," "at," "by," "an," "be," "are," "from," "or," "not," "but," "all," "if," "we," "you," "they," "he," "she," "my," "your," "his," "her," "their," "its," "there," "what."

Then the tokens from the tokenize question match with PDF's keywords. And return the best match score. Which calculate in "calculateRelevanceScore" function. Base score is 0.3 for per keyword match. Now, have the text but don't get the answer. Then return the text to "qaRoutes" of the Routes folder to get the accurate response from the Gemini AI.

4.1.5 Integration of Gemini AI

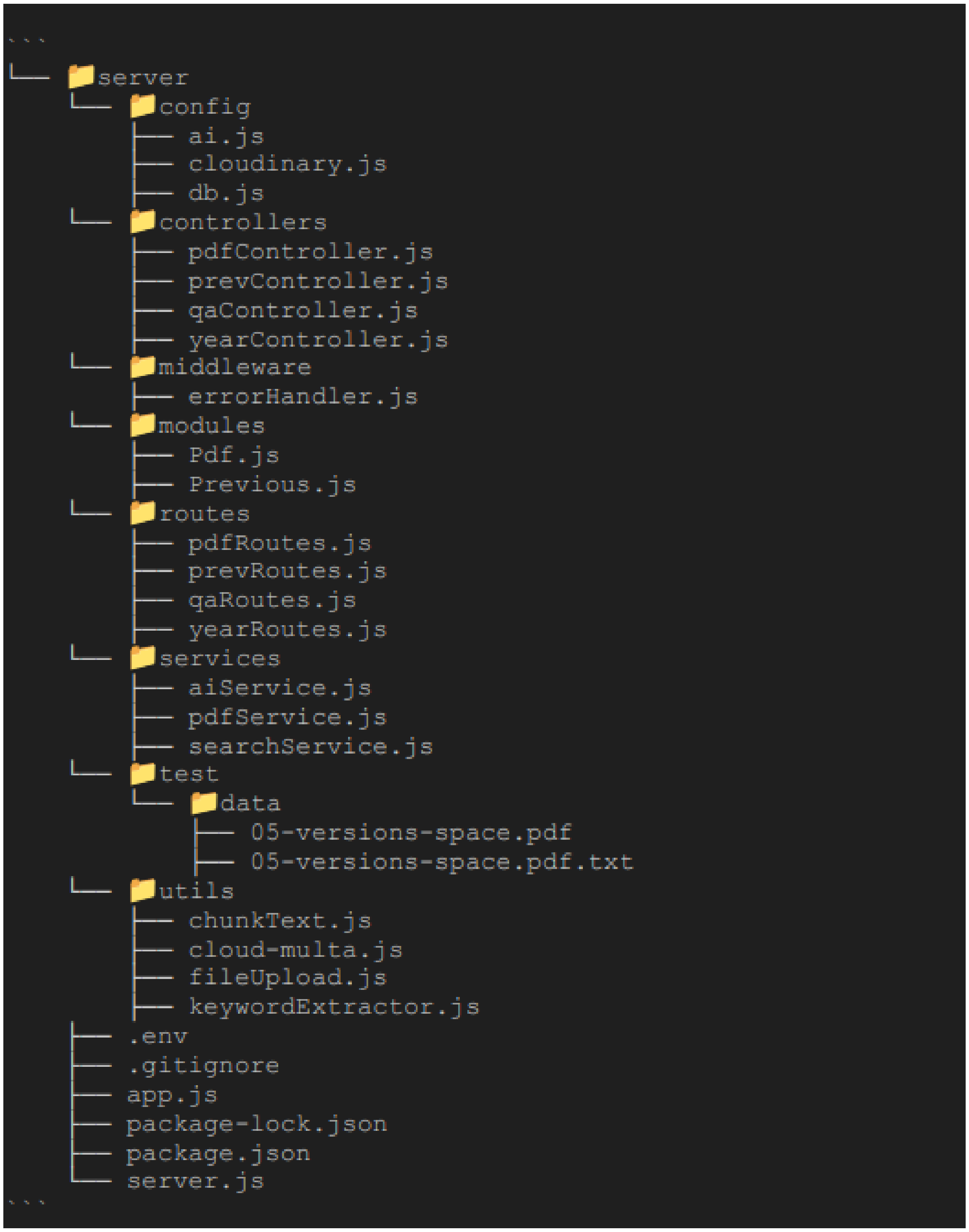
Gemini 2.5 Flash[21] used in this project. Used this version because of:
Low latency
Lower cost
For agentic use cases
High performance

As per Gemini AI documentation, import GoogleGenerativeAI and export "pdfAnswer." It will handle the question, context and PDF answer. Here "context" refer to PDF, PDF answer is our matching text. Here, use of generative AI and provide some instructions. Also using a try-catch block for handling unexpected errors during execution of this project. Then, it respond to the client.



```javascript
import { GoogleGenerativeAI } from "@google/generative-ai";

const genAI = new GoogleGenerativeAI("AIzaSyAehPiSlaGCrxeAZmTXbv_IORCEFLWTGoM");

export const askGemini = async ({ question, context, pdfAnswer }) => {
  const model = genAI.getGenerativeModel({ model: "gemini-2.5-flash" });

  const prompt = `
    PDF Context: """${context}"""

    Initial PDF Answer: """${pdfAnswer || "No direct answer found"}"""

    User Question: ${question}

    Instructions:
    1. If the PDF answer is correct but verbose, summarize it
    2. If the PDF answer is incomplete, enhance it using context
    3. If no answer exists, say "The document doesn't specify"
    4. Respond in 1-2 concise sentences
    5. Maintain original facts exactly
  `;

  try {
    const result = await model.generateContent(prompt);
    return result.response.text();
  } catch (error) {
    console.error("Gemini Error:", error);
    return "Failed to get AI response";
  }
};
```

Figure 4.9: AI integration File

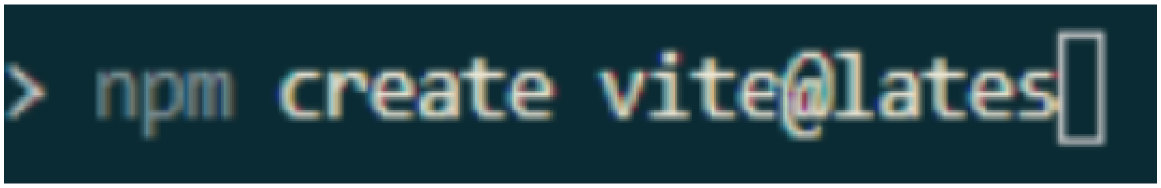


```
```
└── 📁server
    └── 📁config
        ├── ai.js
        ├── cloudinary.js
        ├── db.js
    └── 📁controllers
        ├── pdfController.js
        ├── prevController.js
        ├── qaController.js
        ├── yearController.js
    └── 📁middleware
        ├── errorHandler.js
    └── 📁modules
        ├── Pdf.js
        ├── Previous.js
    └── 📁routes
        ├── pdfRoutes.js
        ├── prevRoutes.js
        ├── qaRoutes.js
        ├── yearRoutes.js
    └── 📁services
        ├── aiService.js
        ├── pdfService.js
        ├── searchService.js
    └── 📁test
        └── 📁data
            ├── 05-versions-space.pdf
            ├── 05-versions-space.pdf.txt
    └── 📁utils
        ├── chunkText.js
        ├── cloud-multa.js
        ├── fileUpload.js
        ├── keywordExtractor.js
    ├── .env
    ├── .gitignore
    ├── app.js
    ├── package-lock.json
    ├── package.json
    └── server.js
```
```

Figure 4.10: Backend File Structure

4.2 Frontend

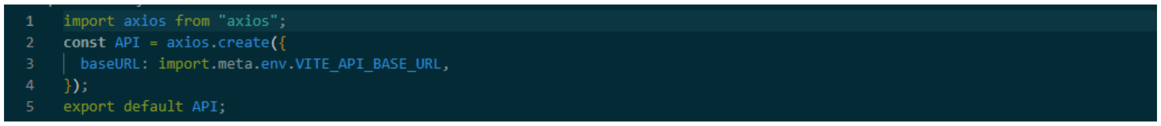
In frontend, the React libraby[22] is used. The Vite framework is used to make folder structures and for the project.



```
> npm create vite@lates
```

Figure 4.11: Vite

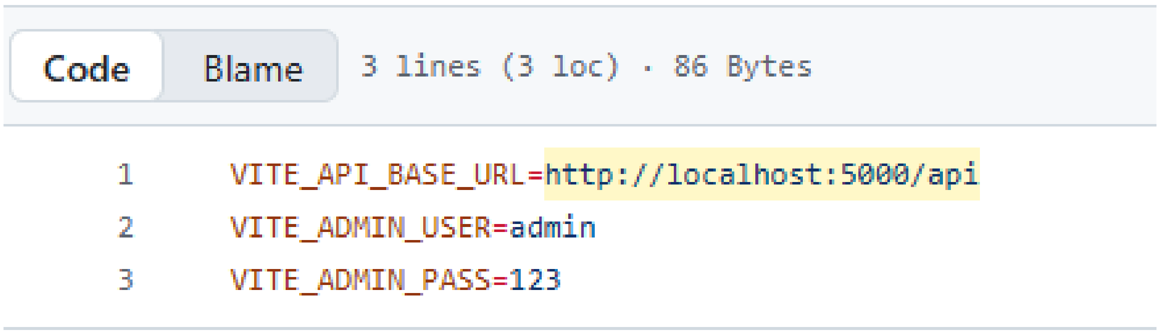
This command created the file structure.
This code, "import axios from "axios";" Import the Axios HTTP client, which used to send HTTP requests. eg; GET, POST, PUT, DELETE, etc. Then create a axios instance.
"import.meta.env.VITE_API_BASE_URL,"



```
import axios from "axios";
const API = axios.create({
  baseURL: import.meta.env.VITE_API_BASE_URL,
});
export default API;
```

Figure 4.12: Axios

This is the way to access the variables of the "env" file's. Also export default API, the goal is use this in other files of the project. The advantage of this file is, need only one-time setup.
At this moment, setup the env file, where:
"http://localhost:5000/api"
is the server location. It means backend running locally on 5000 port.



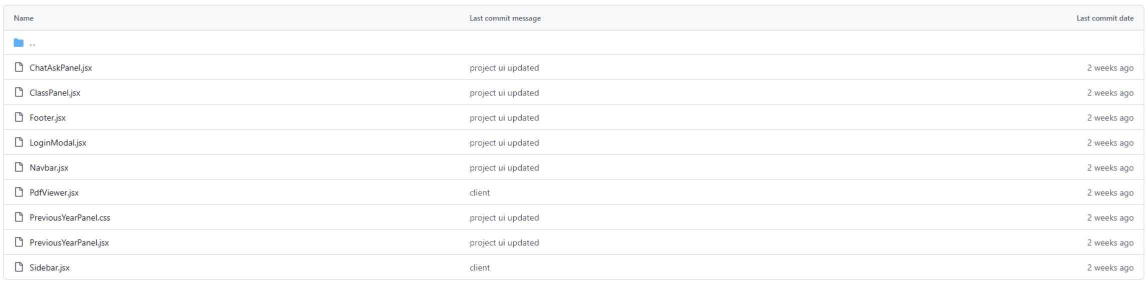
```
VITE_API_BASE_URL=http://localhost:5000/api
VITE_ADMIN_USER=admin
VITE_ADMIN_PASS=123
```

Figure 4.13: .env File

In the "AuthContext" file, login and authetication is checked. Which located in the context folder.
In the "index.css" file, import Tailwind CSS. Used it for writing inline CSS code. It is best for flexibility and customization.

4.2.1 Component

In the component section, React.js and Tailwind CSS are used. In the component folder, created some panels, for instance: Chat ask panel, side bar, navigation bar, login module, previous year panel, pdf viewer, footer etc.

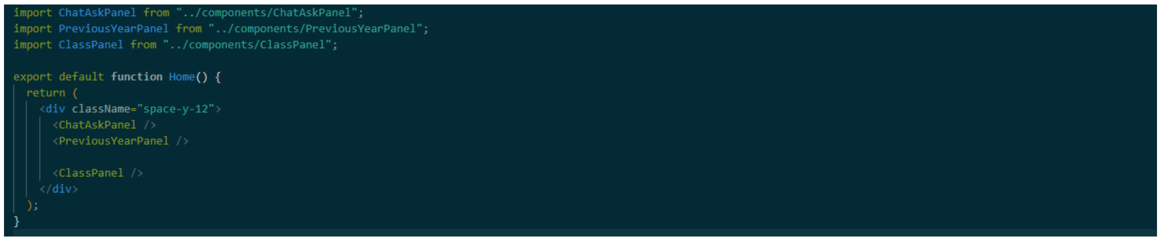


Name	Last commit message	Last commit date
..		
ChatAskPanel.jsx	project ui updated	2 weeks ago
ClassPanel.jsx	project ui updated	2 weeks ago
Footer.jsx	project ui updated	2 weeks ago
LoginModal.jsx	project ui updated	2 weeks ago
Navbar.jsx	project ui updated	2 weeks ago
PdfViewer.jsx	client	2 weeks ago
PreviousYearPanel.css	project ui updated	2 weeks ago
PreviousYearPanel.jsx	project ui updated	2 weeks ago
Sidebar.jsx	client	2 weeks ago

Figure 4.14: Component Folder

4.2.2 Page

In this project, there are two pages: the Home and Admin pages. On the page related components are rearranged. On the home page, there are three panels, each panel has separate sections. For example, in the ask panel, there are two sections.



```
import ChatAskPanel from "../components/ChatAskPanel";
import PreviousYearPanel from "../components/PreviousYearPanel";
import ClassPanel from "../components/ClassPanel";

export default function Home() {
  return (
    <div className="space-y-12">
      <ChatAskPanel />
      <PreviousYearPanel />

      <ClassPanel />
    </div>
  );
}
```

Figure 4.15: Home Page

On the home page, there are three panels; each panel has separate sections. For example, in the ask panel, there are two sections. Each section has a different type of task. In section 1, listing all the PDFs from the database using Axios. In section 2, students can query questions and get answers.

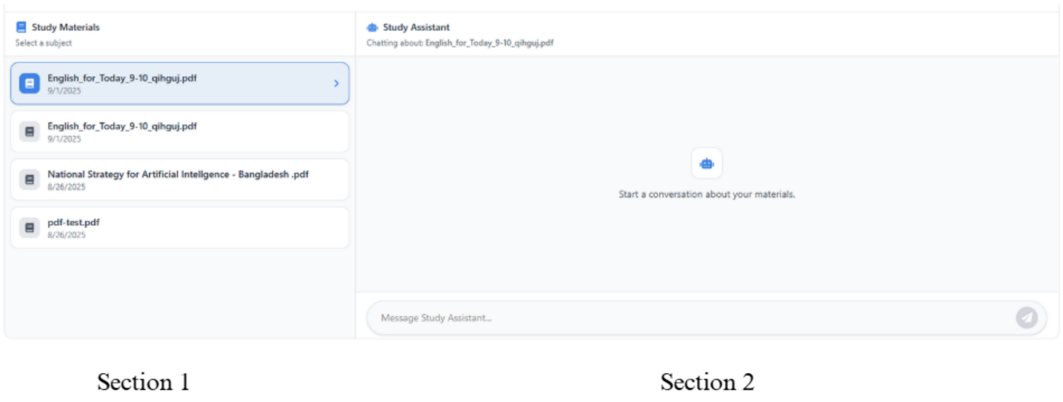

Study Materials
Select a subject
Study Assistant
Chatting about: English_for_Today_9-10_qihguj.pdf
English_for_Today_9-10_qihguj.pdf
9/1/2025
English_for_Today_9-10_qihguj.pdf
9/1/2025
National Strategy for Artificial Intellgence - Bangladesh .pdf
8/26/2025
pdf-test.pdf
8/26/2025
Start a conversation about your materials.
Message Study Assistant...
Section 1
Section 2

Figure 4.16: Ask Panel

To access the admin panel, the user needs to verify their own identity.

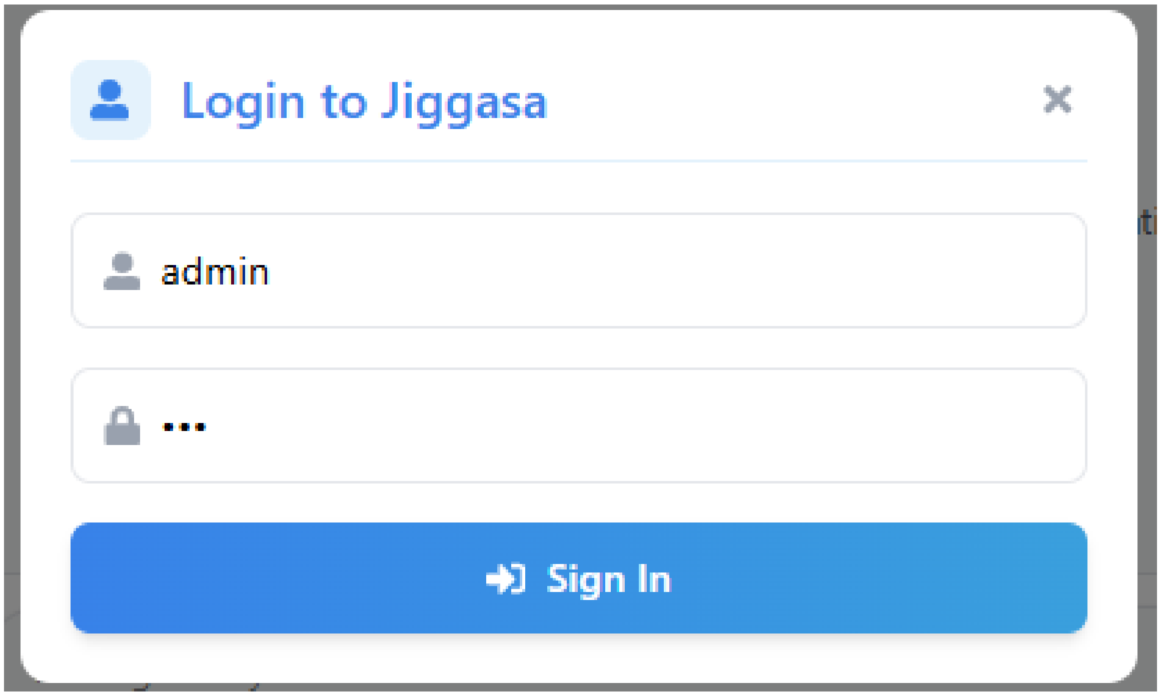
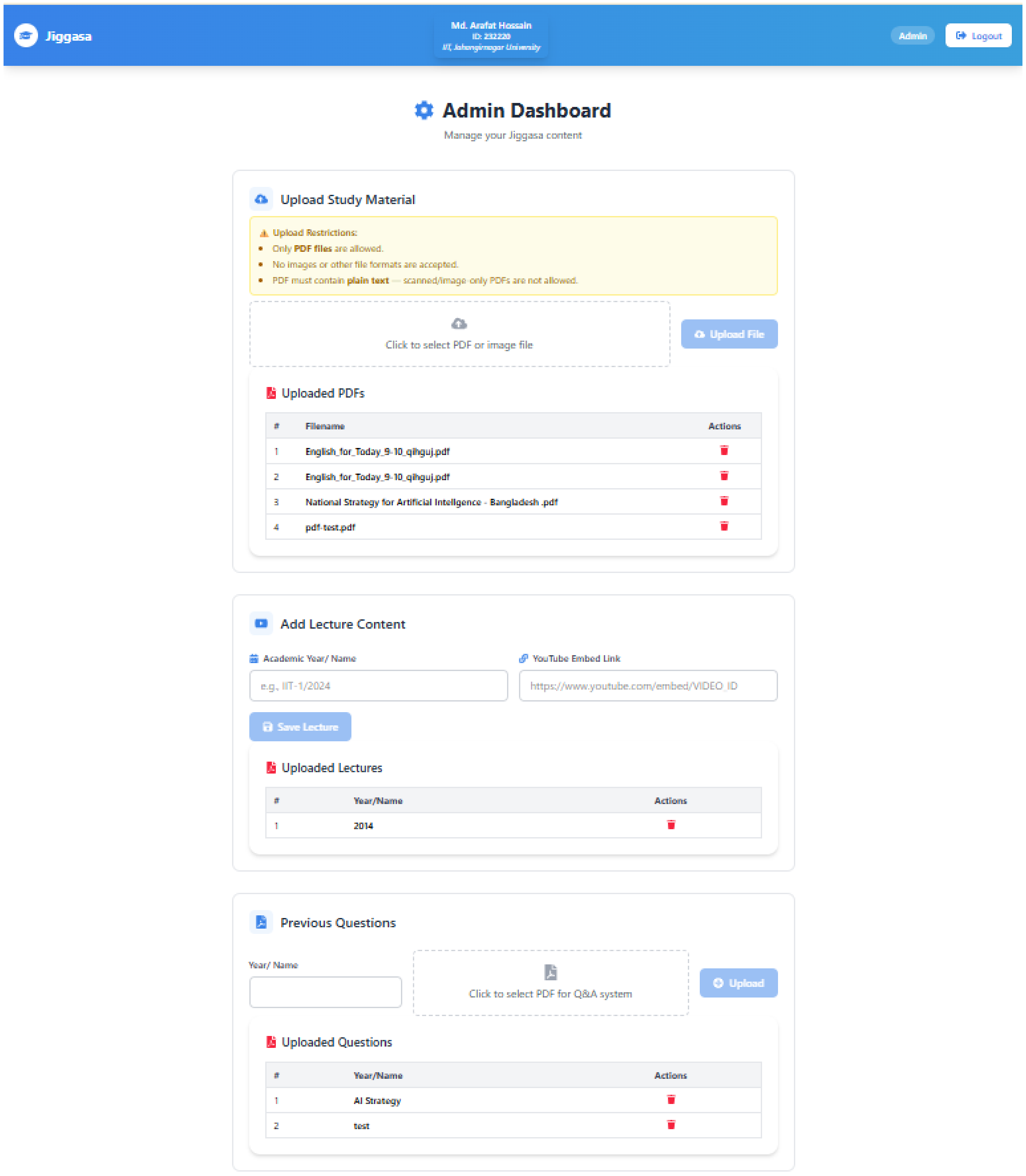


Figure 4.17: Admin Access

In the admin panel, there are three components. Each component has 2 sections. Named of these sections: the Upload section and listing section.



Jiggasa

Md. Arafat Hossain
ID: 232220
IIT, Jahangirnagar University

Admin
Logout

Admin Dashboard

Manage your Jiggasa content

Upload Study Material

Upload Restrictions:

- Only **PDF files** are allowed.
- No images or other file formats are accepted.
- PDF must contain **plain text** — scanned/image-only PDFs are not allowed.

Click to select PDF or image file

Upload File

Uploaded PDFs

#	Filename	Actions
1	English_for_Today_9-10_qihguj.pdf	🗑
2	English_for_Today_9-10_qihguj.pdf	🗑
3	National Strategy for Artificial Intelligence - Bangladesh .pdf	🗑
4	pdf-test.pdf	🗑

Add Lecture Content

Academic Year/ Name: e.g., IIT-1/2024

YouTube Embed Link: https://www.youtube.com/embed/VIDEO_ID

Save Lecture

Uploaded Lectures

#	Year/Name	Actions
1	2014	🗑

Previous Questions

Year/ Name:

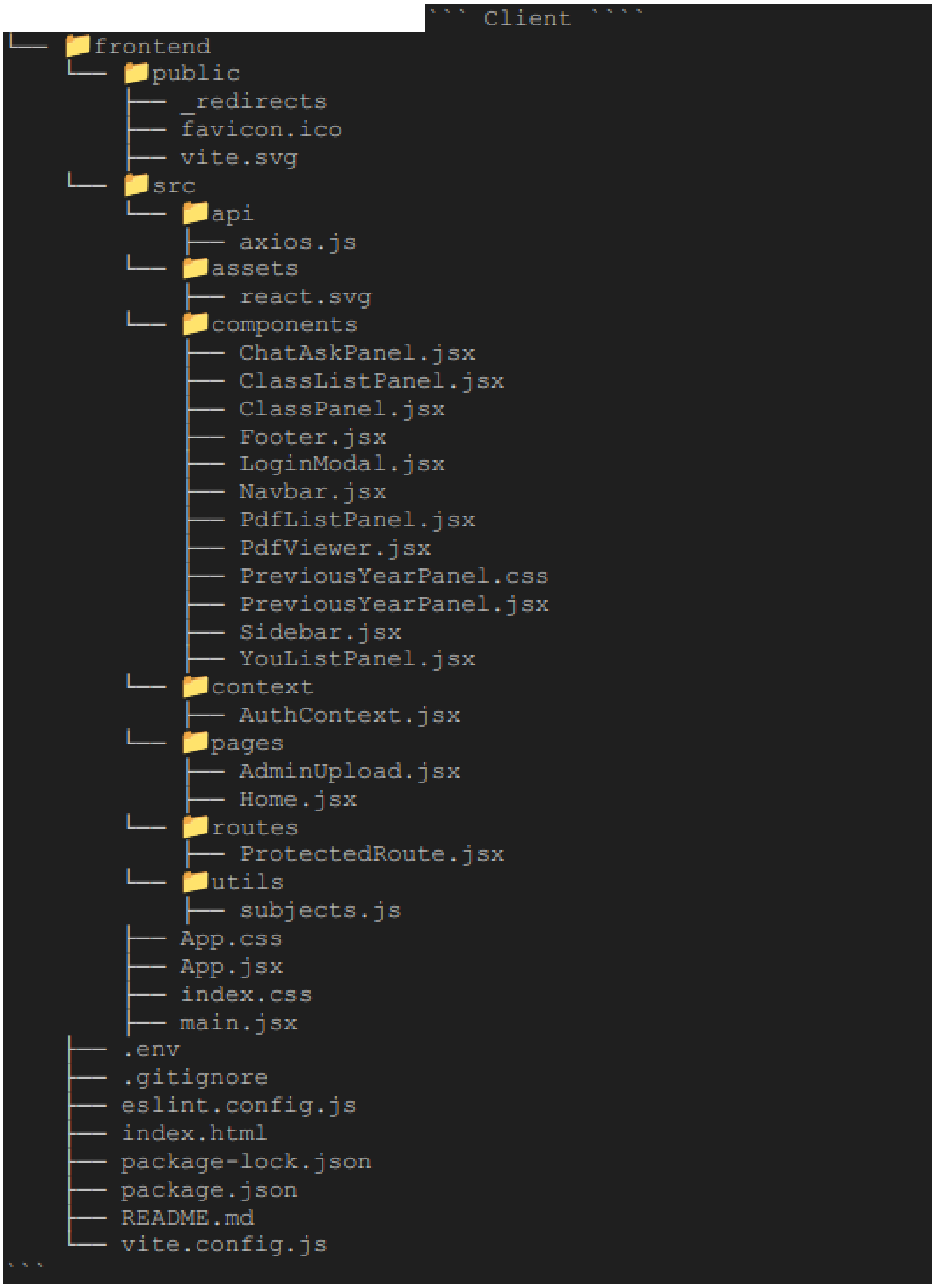
Click to select PDF for Q&A system

Upload

Uploaded Questions

#	Year/Name	Actions
1	AI Strategy	🗑
2	test	🗑

Figure 4.18: Admin Panel



~~~
``` Client ````
└── 📁frontend
    └── 📁public
        ├── _redirects
        ├── favicon.ico
        ├── vite.svg
    └── 📁src
        └── 📁api
            ├── axios.js
        └── 📁assets
            ├── react.svg
        └── 📁components
            ├── ChatAskPanel.jsx
            ├── ClassListPanel.jsx
            ├── ClassPanel.jsx
            ├── Footer.jsx
            ├── LoginModal.jsx
            ├── Navbar.jsx
            ├── PdfListPanel.jsx
            ├── PdfViewer.jsx
            ├── PreviousYearPanel.css
            ├── PreviousYearPanel.jsx
            ├── Sidebar.jsx
            ├── YouListPanel.jsx
        └── 📁context
            ├── AuthContext.jsx
        └── 📁pages
            ├── AdminUpload.jsx
            ├── Home.jsx
        └── 📁routes
            ├── ProtectedRoute.jsx
        └── 📁utils
            ├── subjects.js
        ├── App.css
        ├── App.jsx
        ├── index.css
        ├── main.jsx
    ├── .env
    ├── .gitignore
    ├── eslint.config.js
    ├── index.html
    ├── package-lock.json
    ├── package.json
    ├── README.md
    └── vite.config.js
~~~
```

Figure 4.19: Frontend File Structure
```

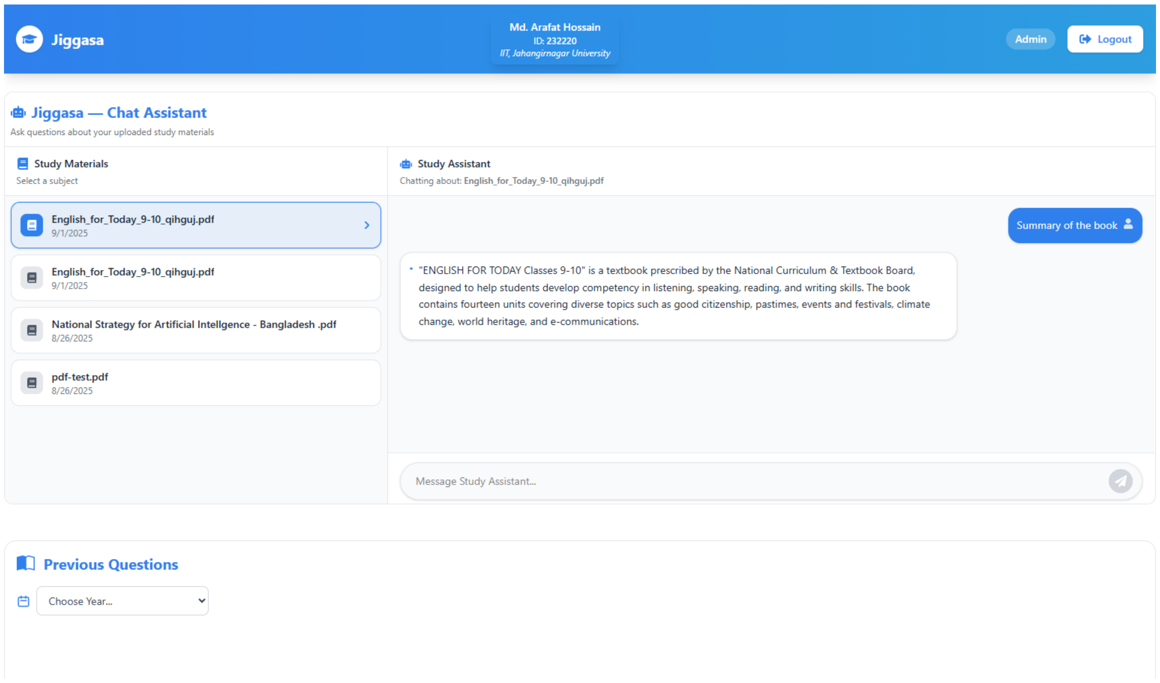
# CHAPTER V

# Result

A set of 250 questions used to test the ability of the Jiggasa Chatbot to answer them. The types of questions will be as follows:

"Descriptive questions" - in which detailed explanations are required. "Wh-questions" -like what, how, which and when. Fill-in-the-blank questions to test knowledge and recall. Multiple Choice Questions (MCQs) with a number of choices to choose from.

Most questions will be based on the provided PDF, but also include some questions from outside the PDF as well. For those outside questions, the chatbot's answer should be "Insufficient information" as the information does not exist in the PDF. And this chatbot answers all the questions.


Jiggasa
Md. Arafat Hossain
ID: 232220
IIT, Jahangirnagar University
Admin
Logout
Jiggasa — Chat Assistant
Ask questions about your uploaded study materials
Study Materials
Select a subject
English_for_Today_9-10_qihguj.pdf
9/1/2025
English_for_Today_9-10_qihguj.pdf
9/1/2025
National Strategy for Artificial Intellgence - Bangladesh .pdf
8/26/2025
pdf-test.pdf
8/26/2025
Study Assistant
Chatting about: English_for_Today_9-10_qihguj.pdf
Message Study Assistant...
Previous Questions
Choose Year...

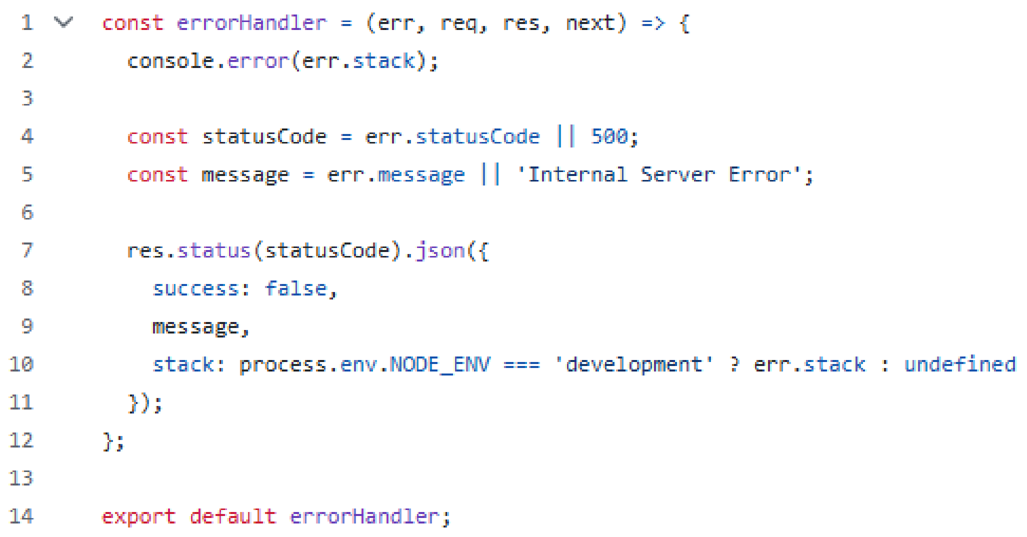
Summary of the book

- "ENGLISH FOR TODAY Classes 9-10" is a textbook prescribed by the National Curriculum & Textbook Board, designed to help students develop competency in listening, speaking, reading, and writing skills. The book contains fourteen units covering diverse topics such as good citizenship, pastimes, events and festivals, climate change, world heritage, and e-communications.

Figure 5.1: Question and Reply

## 5.1 Robust System

In this project, using an error handler file make capable this project to handle the error smothly. Error handler file located in the "middleware" folder.



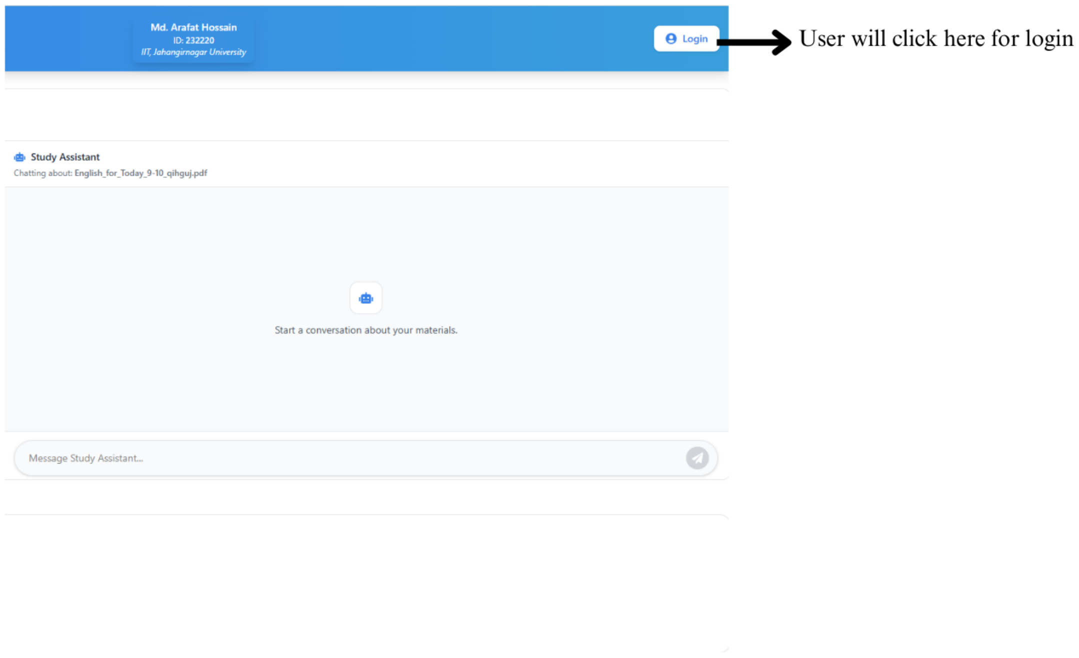
```
const errorHandler = (err, req, res, next) => {
  console.error(err.stack);

  const statusCode = err.statusCode || 500;
  const message = err.message || 'Internal Server Error';

  res.status(statusCode).json({
    success: false,
    message,
    stack: process.env.NODE_ENV === 'development' ? err.stack : undefined
  });
};

export default errorHandler;
```

Figure 5.2: Error Handler File

It is a robust system because of scalability, dependability. It can handle errors and unexpected inputs, which is manually checked. At the time of checking, provides unexpected inputs. But the system didn't fail at that moment. It can handle continuos load, Which is checked manually by contuniously askeing more than 100 questions. And it didn't crush at that time.

5.2 User Guide

Firstly, user should enter this url: https://my-ask.onrender.com/
Then login to this app. For login click the login button which in the top right of the page. Input username and password and click the sign-in button. Input username and password and enter the sign-in button. Which shown in the Fig. 5.2
At this time the user will find the Admin Dashboard. Where the user can select PDF and upload the PDF by clickling the upload button. The Uploaded PDFs section shows the uploaded PDF list. User can find all the PDFs uploaded by him. Which shown in the Fig. 5.3
Then Click the Jiggasa apps logo. Then select PDF from the study materials section. User will select that PDF from which he/she want to ask questions. Then submit their question in "Message Study Assistant" section and press enter. Then it will provide the answer. Which shown in the Fig. 5.4


Md. Arafat Hossain
ID: 232220
IIT, Jahangirnagar University
Login
User will click here for login
Study Assistant
Chatting about: English_for_Today_9-10_qihguj.pdf
Start a conversation about your materials.
Message Study Assistant...

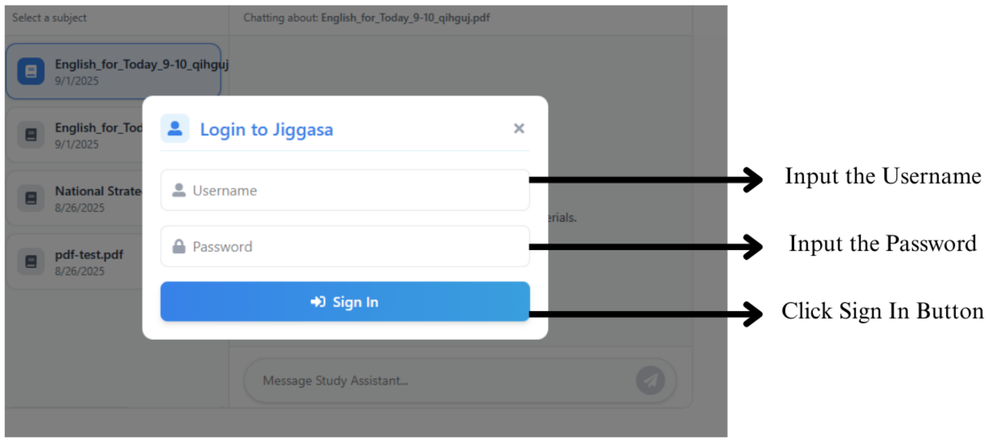
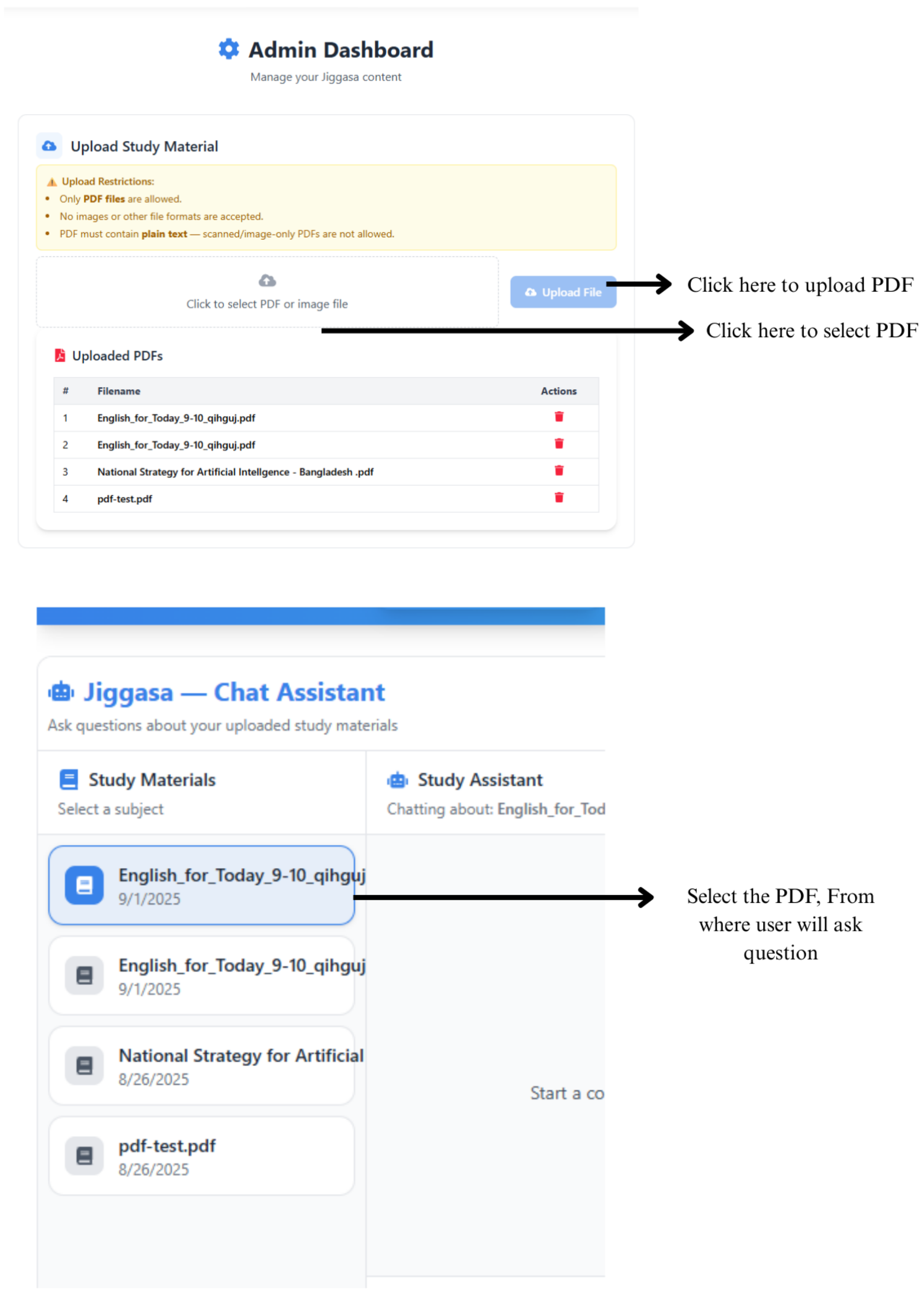


Figure 5.3: Login Section



Admin Dashboard

Manage your Jiggasa content

Upload Study Material

Upload Restrictions:
- Only **PDF files** are allowed.
- No images or other file formats are accepted.
- PDF must contain **plain text** — scanned/image-only PDFs are not allowed.

Click to select PDF or image file

Uploaded PDFs

#	Filename	Actions
1	English_for_Today_9-10_qihguj.pdf	🗑
2	English_for_Today_9-10_qihguj.pdf	🗑
3	National Strategy for Artificial Intellgence - Bangladesh .pdf	🗑
4	pdf-test.pdf	🗑

Jiggasa — Chat Assistant

Ask questions about your uploaded study materials

Study Materials

Select a subject

English_for_Today_9-10_qihguj
9/1/2025

English_for_Today_9-10_qihguj
9/1/2025

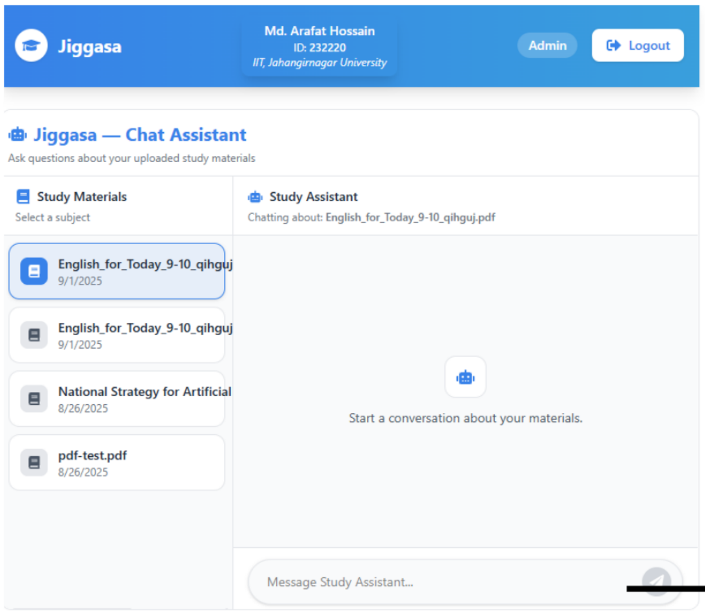
National Strategy for Artificial
8/26/2025

pdf-test.pdf
8/26/2025

Study Assistant

Chatting about: English_for_Tod

Figure 5.4: Upload PDF Section


Jiggasa

Md. Arafat Hossain
ID: 232220
IIT, Jahangirnagar University

Admin Logout

Jiggasa — Chat Assistant

Ask questions about your uploaded study materials

Study Materials
Select a subject

English_for_Today_9-10_qihguj
9/1/2025

English_for_Today_9-10_qihguj
9/1/2025

National Strategy for Artificial
8/26/2025

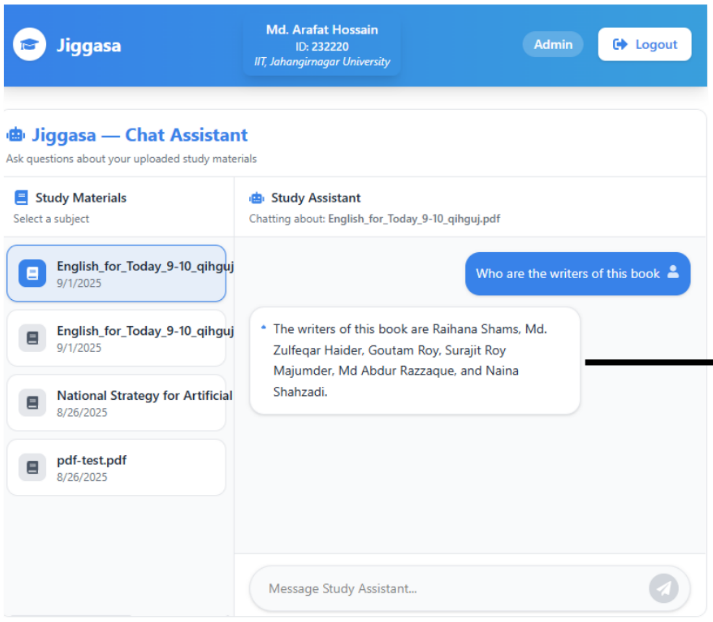
pdf-test.pdf
8/26/2025

Study Assistant
Chatting about: English_for_Today_9-10_qihguj.pdf

Start a conversation about your materials.

Message Study Assistant...

$\longrightarrow$ Submit the question here, And hit enter button


Jiggasa

Md. Arafat Hossain
ID: 232220
IIT, Jahangirnagar University

Admin Logout

Jiggasa — Chat Assistant

Ask questions about your uploaded study materials

Study Materials
Select a subject

English_for_Today_9-10_qihguj
9/1/2025

English_for_Today_9-10_qihguj
9/1/2025

National Strategy for Artificial
8/26/2025

pdf-test.pdf
8/26/2025

Study Assistant
Chatting about: English_for_Today_9-10_qihguj.pdf

Who are the writers of this book

- The writers of this book are Raihana Shams, Md. Zulfeqar Haider, Goutam Roy, Surajit Roy Majumder, Md Abdur Razzaque, and Naina Shahzadi.

Message Study Assistant...

$\longrightarrow$ Get the response here

5

Figure 5.5: Ask a Question and get Respond

CHAPTER VI

Conclusion

6.1 Conclusion

Within this project, we managed to create a server-based application that was tied to a database and optimized with AI-powered functionalities. The application also provides certainty of connection to the database, which the server starts, and all activities take place safely and efficiently. The addition of AI or automation capabilities proves the capability of the system to process user inputs intelligently and provide meaningful and context-aware responses. The application is robust and production-ready with good error handling and environment-based configurations.

Overall, this project provides an example that illustrates the existing best practices of both the backend development and the successful database, integration and scalability in implementing the server that can be enhanced or expanded further.

6.2 Limitation

Significant drawbacks of RAG (Retrieval-Augmented Generation) systems are due to a number of technical and practical issues. Relevancy and accuracy of the documents that are retrieved largely determine the quality of the output generated; when the retrieval component retrieves irrelevant or outdated information, then the response may be wrong or misleading. A second shortcoming of RAG systems is that retrieval introduces latency and execution overhead in that it is required to be run prior to generation, and is therefore potentially slow with real-time applications, particularly with large or unoptimized databases. Another problem is knowledge update, since the system can only answer with the information in the retrieval database; outdated or incomplete information can give wrong answers even when the LLM itself is up to date. The complexity of the implementation is due to the necessity to ensure a close integration and tuning of the retrieval and generation parts, such as indexing, and similarity search. Moreover, the costs of maintaining and querying large databases can be prohibitive in terms of compute and storage, and inference costs increase when an LLM is coupled with retrieval. Lastly, RAG systems generally process text and the inclusion of multimodal data like images or audio only increases complexity.

References

[1] J. Weizenbaum, "Eliza — a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966. First chatbot and conversational AI paper introducing ELIZA.

[2] D. Johnson, "Pdf's popularity online," 2021. Accessed: 2025-09-09.

[3] F. in Artificial Intelligence, "A review on knowledge and information extraction from pdf documents," *Frontiers in Artificial Intelligence*, 2025.

[4] Anonymous, "Agentic retrieval-augmented generation: Advancing ai-driven information retrieval and processing," *International Journal of Computer Technology and Trends (IJCTT)*, vol. 73, no. 1, pp. 111–125, 2025.

[5] G. DeepMind, "Gemini: A family of highly capable multimodal models," 2025. Accessed: 2025-09-09.

[6] H. Qian, P. Zhang, Z. Liu, K. Mao, and Z. Dou, "Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery," 2024.

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021.

[8] Humanloop, "8 retrieval augmented generation (rag) architectures you should know in 2025," 2025.

[9] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1762–1777, Association for Computational Linguistics, 2023.

[10] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. C. Park, "Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity," in *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024. Code available at `https://github.com/starsuzi/Adaptive-RAG`.

[11] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[12] M. Panda, "Agentic rag: Redefining retrieval-augmented generation for adaptive intelligence," *ResearchGate*, 2025.

[13] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence," in *Dartmouth Summer Research Project on Artificial Intelligence*, 1956. The seminal paper introducing the term Artificial Intelligence.

[14] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. LIX, no. 236, pp. 433–460, 1950. Introduced the concept of the Turing Test and explored whether machines can think.

[15] E. H. Shortliffe, B. G. Buchanan, and S. N. Cohen, "Mycin: A rule-based computer program for advising physicians regarding antimicrobial therapy selection," *Computers and Biomedical Research*, vol. 8, no. 4, pp. 303–320, 1975. One of the earliest applications of AI in healthcare.

[16] L. Dostert and Y. Bar-Hillel, "The georgetown-ibm experiment," in *Proceedings of the Georgetown-IBM Machine Translation Demonstration*, (Washington, D.C.), 1954. First demonstration of machine translation, translating 60 Russian sentences into English.

[17] J. Warnock and C. Geschke, "The portable document format (pdf): Adobe's approach to electronic documents," in *Proceedings of the ACM Conference on Document Engineering*, (New York, NY, USA), ACM, 1993. Introduced PDF as a standardized format for digital documents.

[18] B. Patra and M. Kumar, "Natural language processing in chatbots: A review," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 11, no. 3, pp. 2890–2894, 2020.

[19] S. Joshi, "Comprehensive review of ai hallucinations: Impacts and mitigation strategies for financial and business applications," *International Journal of Computer Applications Technology and Research*, vol. 14, no. 6, pp. 38–50, 2025.

[20] F. U. Zaman, "Comparative case study: Difference between azure cloud sql and atlas mongodb nosql database," *International Journal of Emerging Trends in Engineering Research*, vol. 9, no. 7, pp. 999–1002, 2021.

[21] G. Team, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," tech. rep., Google, 2025. Accessed: 2025-09-10.

[22] I. Journals, "React js – an emerging frontend javascript library," *IRE Journals*, 2021. Accessed: 2025-09-10.