

PDF QUERY ASSISTANT

A project Report

Submitted in Partial Fulfilment of the
Requirements for the Award of Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Mayur Kishor Kamble TYIT20

Under the esteemed guidance of

Ms. Harshala Chaudhari

Head of Department



DEPARTMENT OF INFORMATION TECHNOLOGY

BHAVAN'S COLLEGE

(Affiliated to University of Mumbai)

MUMBAI, 400058

MAHARASHTRA

YEAR 2023-2024

BHAVAN'S COLLEGE
(Affiliated to University of Mumbai)
MUMBAI- MAHARASHTRA-400058

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project entitled, " **PDF Query Assistant** " is bonafied work of **Mayur Kishor Kamble** bearing Seat No: **(TYIT20)** submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide

Coordinator

External Examiner

Date:

College Seal

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR NO: 2021016401939571

Roll No: TYIT20

1. Name of the Students

Mayur Kishor Kamble

2. Title of the Project

Auction System

3. Name of the Guide

Ms. Harshala Chaudhari

4. Teaching experience of the Guide

5. Is this your first Submission?

Yes ☐

No ☐

Signature of the Student

Date: _____

Signature of the Guide

Date: _____

Signature of the Coordinator

Date: _____

ABSTRACT

Our project, the “PDF Query Assistant using Generative AI,” is a complex integration of state-of-the-art technologies, including Langchain, OpenAI, Pinecone, and Streamlit , to optimize the process of extracting meaningful information from PDF documents. This system employs the Langchain text splitter and embeddings for text processing and HuggingFace model for natural language understanding and response generation. Pinecone is utilized for efficient vector storage and similarity searches, and Streamlit provides the user interface. This technical solution automates document analysis, question answering, and conversation history management, promising to revolutionize the domain of document management and information retrieval through AI-driven techniques.

Keywords: User ,PDF ,Query

ACKNOWLEDGEMENT

I would like to place my sincere gratitude to whom so ever has that they have contributed to the successful completion of this project directly or indirectly.

I take this opportunity to express my profound gratitude and deep regards to our guide Ms. Harshala Chaudhari without whose guidance & critical appreciation, this project would have been incomplete. Right from its inception, this project has been shaped by her expert opinions and he has helped me improve this project in all manners and achieve the level that it has acquired

Last but not the least I place a deep sense of gratitude to our family members and our friends who have been constant source of inspiration during the preparation of this project work.

DECLARATION

I hereby declare that this project report entitled “PDF Query Assistant”, submitted by me, under the guidance of (Harshala Chaudhari), (HOD) of (IT Dept.) is my own and has not been submitted to any other University or Institute or published earlier.

The project is done in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY) to be submitted as final semester project as part of our curriculum.

Name of the Student

Signature of the Student

TABLE OF CONTENT

CHAPTER-1

INTRODUCTION.....1

1.1 BACKGROUND.....1

1.2 OBJECTIVES.....1

1.3 PURPOSE, SCOPE, APPLICABILITY.....2

1.3.1 PURPOSE.....2

1.3.2 SCOPE.....2

1.3.3 APPLICABILITY.....3

1.4 ACHIVEMENTS.....4

1.5 ORGANIZATION OF REPORT4

CHAPTER-2 SURVEY OF TECHNOLOGIES.....5

2.1 Technology Overview5

CHAPTER-3 REQUIREMENTS AND ANALYSIS.....7

3.1 PROBLEM DEFINATION.....7

3.2 REQUIREMENTS SPECIFICATIONS.....7

3.3 PLANNING AND SCHEDULING.....8

3.4 HARDWARE AND SOFTWARE REQUIREMENTS.....11

3.5 PRELIMINARY PRODUCT DESCRIPTION.....12

3.6 CONCEPTUAL MODEL.....13

3.6.1 USE CASE DIAGRAM.....13

3.6.2 CLASS DIAGRAM.....15

3.6.3 ACTIVITY DIAGRAM.....17

3.6.4 FLOWCHART DIAGRAM.....19

3.6.5 ENTITY RELATIONSHIP DIAGRAM.....21

3.6.6 LEVEL 0 DATA FLOW DIAGRAM.....22

3.6.7 LEVEL 1 DATA FLOW DIAGRAM.....23

3.6.8 LEVEL 2 DATA FLOW DIAGRAM.....	24
3.6.9 SEQUENCE DIAGRAM.....	25
CHAPTER-4 SYSTEM DESIGN.....	26
4.1 BASIC MODULES.....	26
4.2 DATA DESIGN.....	28
4.2.1 SCHEMA DESIGN.....	28
4.2.2 DATA INTEGRITY AND CONSTRAINTS.....	29
4.3 PROCEDURAL DESIGN.....	31
4.3.1 LOGIC DIAGRAMS.....	37
4.3.2 ALGORITHM DESIGN FOR USER.....	32
4.4 USER INTERFACE DESIGN.....	33
4.5 SECURITY ISSUES.....	35
4.6 TEST CASE DESIGN.....	35
CHAPTER-5 IMPLEMENTATION AND TESTING	36
5.1 IMPLEMENTATION APPROACH.....	36
5.2 CODING DETAILS AND CODING EFFIEIENCY.....	37
5.3 TESTING APPROACH.....	105
5.4 MODIFICATION AND IMPLEMENTATION	108
5.5 TEST CASES.....	108
CHAPTER 6 RESULTS AND DISCUSSIONS.....	111
6.1 OUTPUT	111
6.2 USER DOCUMENTATION.....	117
CHAPTER 7 CONCLUSION	118
7.1 CONCLUSION.....	118
7.2 LIMITATION OF THE SYSTEM.....	118
7.3 FUTURE SCOPE OF THE PROJECT.....	118
7.4 REFERENCE.....	119

LIST OF FIGURES

Fig. No.	Name	Page no.
3.3.1	Gantt chart	12
3.3.2	Pert Chart	13
3.6.1	Use case diagram	17
3.6.2	Class diagram	19
3.6.3	Activity diagram	21
3.6.4	Flow chart diagram	23
3.6.5	Entity relationship diagram	25
3.6.6	Level-0 DFD	26
3.6.7	DFD Level 1	27
3.6.8	Sequence Diagram	29
4.2.1	Schema Design	30
4.3.1.1	Logic Diagram	31

LIST OF TABLES

Table No.	Name	Page no.
3.6.1	Use case table	13
3.6.2	Class table	15
3.6.3	Activity table	17
3.6.4	Flow chart table	19
3.6.6	Level-0 DFD table	22
4.2.2.1	User database table	29
4.2.2.2	Ads database table	29
4.2.2.3	Chat database table	30
4.5	Security issues table	35
4.6	Test case table	35
5.3.1	System Integration Test 1	105
5.3.2	System Integration Test 1.1	105
5.3.3	System Integration Test 2.0	106
5.3.4	System Integration Test 2.1	106
5.3.5	System Integration Test 3.0	107
5.5.1	Search bar table	108
5.5.2	Ads and Favorite Table	109
5.5.3	Add item Table	109
7.4	Reference Table	119

CHAPTER 1 INTRODUCTION

1.1 BACKGROUND:

PDF Query Assistance is an innovative solution designed to empower users with the ability to upload documents and effortlessly extract valuable insights through the simple act of asking questions. Whether you are a student researching a complex topic, a professional delving into extensive reports, or an individual seeking to understand a legal document, our platform is here to provide you with the answers you seek.

This project leverages the power of cutting-edge natural language processing technology to comprehend the content of uploaded documents and provide precise responses to user-generated queries.

Imagine uploading a lengthy research paper or a dense legal contract and being able to pose questions about specific sections or details, and receiving instant, accurate answers. It's akin to having a personal document assistant at your disposal, ready to illuminate the text with insightful responses.

In this document, we will delve into the core components of the PDF Query Assistant project, its underlying technologies, and the steps involved in utilizing this platform effectively. Whether you are a developer eager to integrate this solution into your applications, an end user seeking to harness its capabilities, or a business looking to streamline information extraction processes, this document is your gateway to understanding the power and potential of PDF Query Assistance. Together, we will embark on a journey that marks a paradigm shift in how we interact with the written word, transforming static documents into dynamic sources of knowledge.

Welcome to a new era of document query assistance – a seamless bridge between human curiosity and the vast reservoir of information stored in digital documents.

1.2 OBJECTIVES:

The primary objectives of the "PDF Query Assistant" project are as follows:

Streamline PDF document management, enable natural language interaction, semantic search, and efficient document indexing

Efficient Information Retrieval: Enable users to upload documents of various formats (especially PDFs) and efficiently retrieve specific information from within these documents by asking natural language questions.

Natural Language Understanding: Employ advanced natural language processing (NLP) techniques to understand and process both user queries and the content of uploaded documents, ensuring accurate and contextually relevant responses.

Document Content Analysis: Develop robust algorithms for parsing, summarizing, and indexing the content of uploaded documents, making it searchable and accessible for answering user queries.

User-Friendly Interface: Create an intuitive and user-friendly web-based interface that allows individuals, regardless of their technical expertise, to upload documents, ask questions, and receive prompt responses.

High Accuracy and Precision: Strive for high accuracy and precision in responding to user queries to ensure that the generated answers are both informative and reliable.

Scalability: Design the system to handle a wide range of document types, sizes, and complexities, ensuring scalability to accommodate a growing user base and increasing data volume.

our platform by facilitating communication between buyers and sellers. Building trust and fostering relationships among our users is a core objective, as it contributes to a positive and sustainable trading environment

1.3 PURPOSE, SCOPE, AND APPLICABILITY

1.3.1 PURPOSE

The purpose of the "PDF Query Assistance" project is driven by the need to streamline and enhance the process of accessing, understanding, and extracting valuable information from documents, particularly PDF files. This innovative platform aims to empower individuals and organizations by providing a user-friendly and efficient means of navigating the vast sea of information contained within documents of diverse formats. By enabling users to ask natural language questions related to document content, this project revolutionizes the way we interact with written materials.

One of the primary objectives of this project is to democratize access to knowledge. It empowers a broad spectrum of users, from students and researchers to professionals and individuals, by simplifying the arduous task of deciphering and comprehending complex documents. Instead of laboriously scanning pages or sifting through extensive reports, users can effortlessly seek and receive specific information. This enhances their ability to make informed decisions, conduct thorough research, or simply stay well-informed on various subjects.

Another crucial purpose of this project is to save time and resources. In a world inundated with information, efficiency in information retrieval is essential. PDF Query Assistance reduces the time and effort required to find relevant data within documents, leading to significant time and resource savings. This, in turn, fosters increased productivity and the potential for more efficient work processes.

1.3.2 SCOPE

The scope of the "PDF Query Assistance" project is comprehensive, encompassing several critical components that define its functionality and reach. At its core, the project will focus on PDF documents, recognizing their prevalence in academia, business, and research. These documents, often complex and lengthy, will serve as the primary content source. The project's scope also extends to the implementation of advanced natural language processing (NLP) techniques, enabling the system to comprehend both user queries and the content within the uploaded documents. It involves the development of a user-friendly web-based interface, designed for accessibility and ease of use for individuals with varying levels of technical expertise. Furthermore, the project encompasses the creation of algorithms for document parsing, summarization, and indexing, making the content of uploaded documents searchable and accessible for efficient question-answering. Ultimately, the project's scope spans the entire process, from document analysis to accurate answer generation, with the aim of revolutionizing the way user's access and extract knowledge from documents.

1.3.3 APPLICABILITY (FEASIBILITY STUDY)

The applicability of the "PDF Query Assistance" project is wide-ranging and holds substantial potential for diverse sectors and industries. In academia and research, it equips students and scholars with a powerful tool for efficiently accessing and extracting knowledge from academic papers and research articles, significantly expediting the literature review and data extraction processes. The project is equally valuable in the legal and compliance domain, empowering legal professionals to streamline contract analysis, legal research, and due diligence efforts. It aids in pinpointing crucial clauses and legal insights within complex documents, enhancing the accuracy and speed of legal operations. Moreover, in the business and market research arena, the project serves as an invaluable resource for business analysts and professionals seeking to gather market trends, competitor insights, and industry-specific data from extensive reports and documents. This accelerates the information-gathering process, facilitating more informed and strategic decision-making. The applicability of the "PDF Query Assistance" project extends to any field that relies on efficient information retrieval and knowledge extraction from documents, making it a versatile and indispensable asset for various sectors and industries.

1.3.4 ACHIEVEMENTS

Our PDF Query Assistant represents a significant advancement in document analysis and management. With a sleek and intuitive user interface built using Python Streamlit, users can effortlessly extract valuable insights from PDF documents. Leveraging the power of HuggingFace's state-of-the-art NLP models, our assistant enables precise querying and analysis, making it an indispensable tool for researchers, businesses, and educators alike. PyPDF facilitates seamless extraction of text and metadata, ensuring accurate and comprehensive document processing. As we continue to refine and enhance our assistant, incorporating user feedback and embracing emerging technologies, we're committed to empowering users with efficient and effective document analysis capabilities. Join us on this journey as we redefine document management in the digital era.

1.3.5 ORGANIZATION OF REPORT

This report aims to provide a comprehensive understanding of our PDF query assistant project. It begins with an exploration of the project's background, shedding light on the motivations behind its development and its relevance within the realm of document analysis and management. We delve into the evolution of document processing needs and the challenges users face, setting the stage for the inception of our innovative solution.

Following the background, we outline the project's objectives, emphasizing our core principles of efficiency, accuracy, and user-centric design. With a focus on security, usability, and community engagement, we aim to redefine the landscape of document analysis tools, catering to the diverse needs of individuals, businesses, and researchers alike.

Moving forward, the report discusses the purpose, scope, and applicability of our PDF query assistant. Here, we highlight the versatility and broad utility of our platform, showcasing its ability to handle various document types, languages, and analytical tasks. Whether it's extracting insights from research papers, analyzing legal documents, or conducting sentiment analysis on customer feedback, our platform offers a comprehensive solution for document-related challenges.

CHAPTER 2 SURVEY OF TECHNOLOGY

2.1 TECHNOLOGY OVERVIEW

In the process of PDF Query Assistant different type of technologies are being used.

Following are the few process and technologies:

2.1.1 NLP

Natural Language Processing (NLP) is a branch of artificial intelligence focused on enabling computers to understand, interpret, and generate human language. It encompasses a range of tasks, including language translation, sentiment analysis, text summarization, and speech recognition. NLP algorithms analyze and process large volumes of natural language data, extracting meaning, patterns, and insights to facilitate human-computer interaction. By applying techniques from linguistics, machine learning, and computational linguistics, NLP systems can comprehend and generate human-like text, enabling applications such as chatbots, virtual assistants, and language translation services.

.

2.1.2 LLM

Large Language Models (LLMs) are advanced artificial intelligence models designed to understand and generate human-like text. They are trained on vast amounts of text data using deep learning techniques, enabling them to comprehend and produce natural language with remarkable accuracy and coherence. LLMs have revolutionized various natural language processing tasks, including language translation, text generation, and sentiment analysis. Examples of LLMs include GPT (Generative Pre-trained Transformer) models developed by OpenAI and BERT (Bidirectional Encoder Representations from Transformers) developed by Google. These models have demonstrated impressive capabilities in understanding and generating text across different languages and domains.

2.1.3 Fast Text Extraction and Interaction :

PyPDF2 is a python library that provides methods for fast text Extraction

It seamlessly extract text without any errors , mistakes

CHAPTER 3 REQUIREMENTS AND ANALYSIS

3.1 PROBLEM DEFINITION

The current landscape of PDF document management systems faces several significant challenges that hinder their effectiveness and usability. These limitations include issues related to information retrieval, human-computer interaction, context comprehension, complexity, and manual labor .

Firstly, many existing systems struggle with efficiently finding and retrieving information within PDF documents. Users often encounter difficulties when searching for specific content, keywords, or relevant sections within a document. This inefficiency can be frustrating and time-consuming, especially when dealing with extensive PDF libraries.

Secondly, these systems typically lack natural language interaction capabilities. Users are often forced to rely on rigid keyword-based queries, which may not fully align with how people naturally communicate. This can create a disconnect between the user and the system, making interactions less intuitive and efficient.

Additionally, the context in which user queries are made is often not well-understood by current systems. They may struggle to interpret the user's intent accurately, leading to less precise search results. The inability to consider context can be a significant drawback, especially when dealing with complex or multifaceted queries.

Another challenge is the complexity of many PDF document management systems. They may have intricate user interfaces, require technical expertise, or demand a steep learning curve, making them less accessible to non-technical users. This complexity can deter potential users and limit the adoption of these systems.

3.2 REQUIREMENT SPECIFICATION

The application for PDF Query Assistant will require a software tool which will help to make an website

For building Online Auction System application there will be need of laptop or a computer with minimum 4GB of ram i3 processor and minimum space in hard disk because android studio is very heavy

- It should be user friendly
- It should perform the basic requirements.
- The GUI should be interactive.
- The system should not clash.
- It should be able to generate answers after all the data extraction process.
- The Report generated needs to be in standard format.
- The system should be easy to understand and work with it.

3.3 PLANNING AND SCHEDULING

To successfully complete my project, it was necessary for us to learn various technologies. A Gantt chart may be a sort of chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars within the graph shows the duration of every activity.

Gantt charts illustrate the beginning and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements constitute the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts will be accustomed show current schedule status using percent-complete shadings and a vertical "TODAY" line. Gantt charts are sometimes equated with bar charts.

Gantt charts are usually created initially using an early start time approach, where each task is scheduled to start out immediately when its prerequisites are complete. This method maximizes the float time available for all tasks. Gantt charts are often used for scheduling generic resources, so yet as they use in project management, they'll even be utilized in scheduling production processes and employee rostering. within the latter context, they will even be referred to as time bar schedules.

GANTT CHART



Fig No: 3.3.1 Gantt chart

PERT CHART

PERT was developed to take account of the uncertainty surrounding estimates of task durations. It was developed in an environment of expensive, high-risk and state-of-the-art projects — not that dissimilar to many of today's large software projects.

Most likely time: the time we would expect the task to take under normal circumstances.
We shall identify this by the letter m.

Optimistic time: the shortest time in which we could expect to complete the activity, barring outright miracles. We shall use the letter a for this.

Pessimistic time: the worst possible time, allowing for all reasonable eventualities but excluding 'acts of God and warfare' (as they say in most insurance exclusion clauses).
We shall call this b.

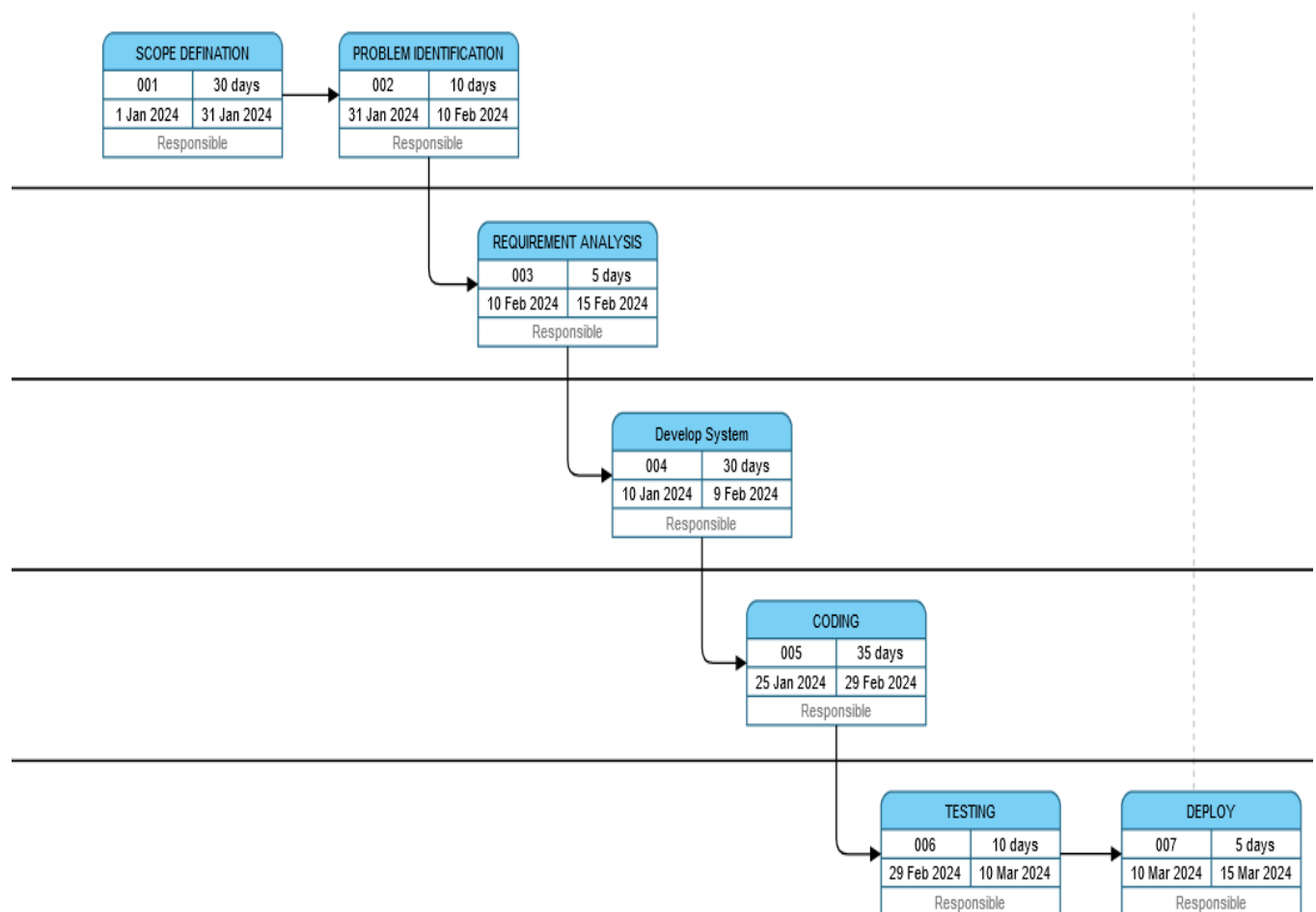


Fig No 3.5.2 Pert chart

3.4 SOFTWARE AND HARDWARE REQUIREMENTS

The Project application is loaded in VS Code. We used VS code for Design and coding of project. Created and maintained all databases into Firebase and CPU , firebase is a real time database used for fast authentication.

Hardware Requirement:

1. Laptop or PC

- i5 Processor Based Computer or higher
- 8GB RAM
- 500 GB Hard Disk

❖ Software Requirement:

1. Laptop or PC

- Windows 7 or higher.
- Python
- Vs Code

3.5 models used in project (hkunlp/instructor-xl)

LLM stands for Large Language Model, and it represents a class of artificial intelligence models designed for understanding and generating human language. These models are characterized by their vast size, typically involving billions of parameters, and they are built using deep learning techniques, particularly transformer architectures.

LLMs have demonstrated remarkable capabilities in various natural language processing tasks, such as machine translation, text generation, question answering, and sentiment analysis. They excel at capturing complex patterns and nuances in language, making them highly versatile tools for understanding and generating text.

One of the most renowned LLMs is hkunlp/instructor-xl (Generative Pre-trained Transformer 3), which has been applied in chatbots, virtual assistants, content generation, and more. LLMs learn from massive amounts of text data, allowing them to respond to user inputs or generate coherent text in a contextually relevant manner.

However, LLMs are not without challenges, including ethical concerns, biases in their training data, and the resources required to train and deploy them. These models continue to evolve, promising a broad range of applications in text-based tasks and human-computer

Interactions.

The following hardware and software are required for the development and deployment of the system.

3.6 CONCEPTUAL MODEL

3.6.1 USE CASE DIAGRAM

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures. Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed.

COMPONENTS:

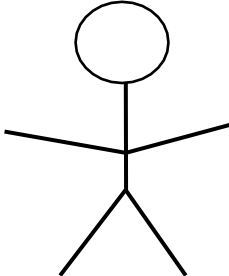


COMPONENT NAME	SYMBOL	DESCRIPTION
Actor/user		An actor specifies a roles played by a user or any other system that interacts with that subject.
Use cases		Represent the different uses that a user might have.
Associations		A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.

Table No 3.6.1

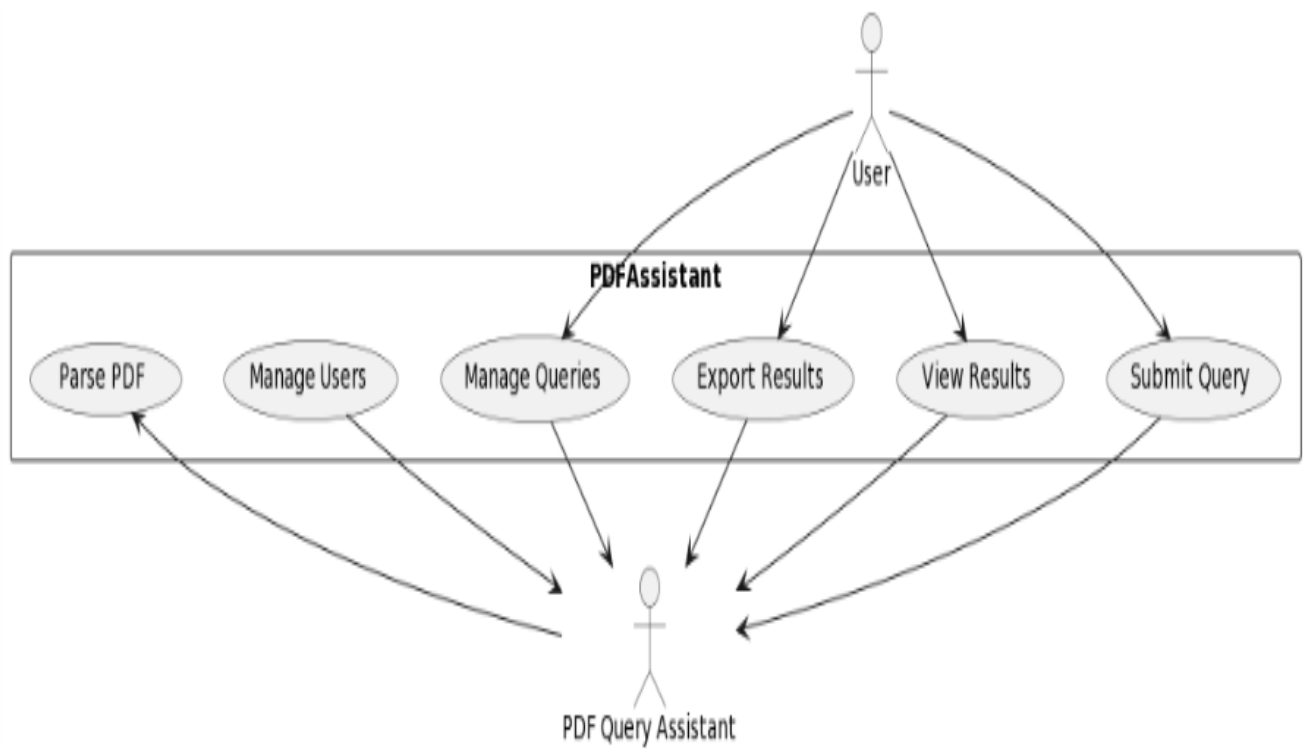


Fig 3.6.1 Use case diagram

3.6.2 CLASS DIAGRAM:

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object- oriented languages.





COMPONENT NAME	SYMBOL	DESCRIPTION
Upper section		Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
Middle section		Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
Bottom section		Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.
Association		A line between classes. In complex diagrams, it is important to know which actors are associated with which use cases.

Table -3.6.2 Class Table

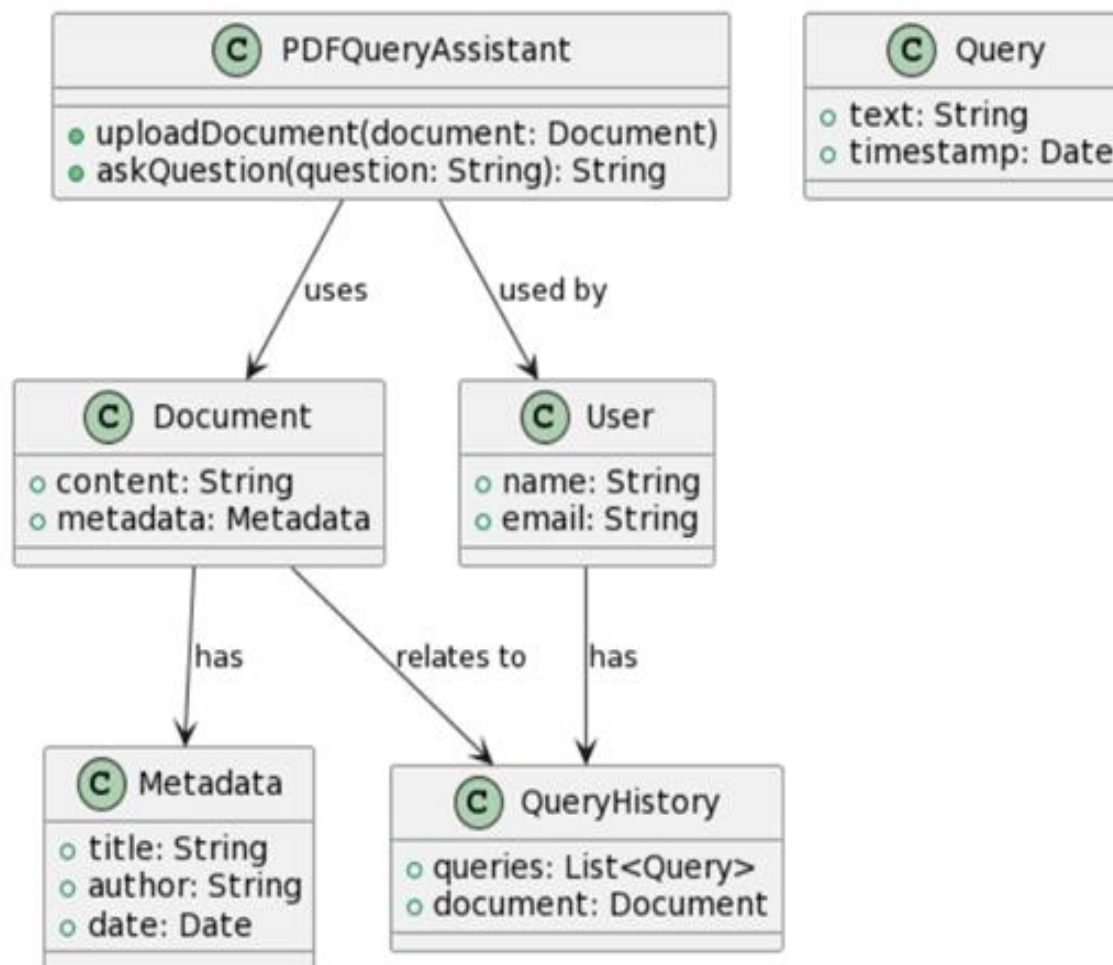


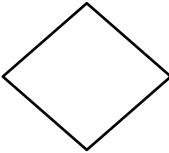



Fig: 3.6.2 Class Diagram

3.6.3 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the United Modelling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities.

COMPONENTS:

COMPONENT NAME	SYMBOL	DESCRIPTION
Start		Represents the beginning of a process or workflow in an activity diagram.
Activity		Indicates the activities that make up a modelled process.
Decision		Represented a decision and always has at least two paths branching out with condition text to allow users to view options.
Control Flow		Shows the directional flow, or control flow, of the activity. An incoming arrow starts a step of an activity; once the step is completed, the flow continues with the outgoing arrow.


End		Marks the end state of an activity and represents the completion of all flows of a process.
-----	---	---

Table -3.6.3 Activity Table

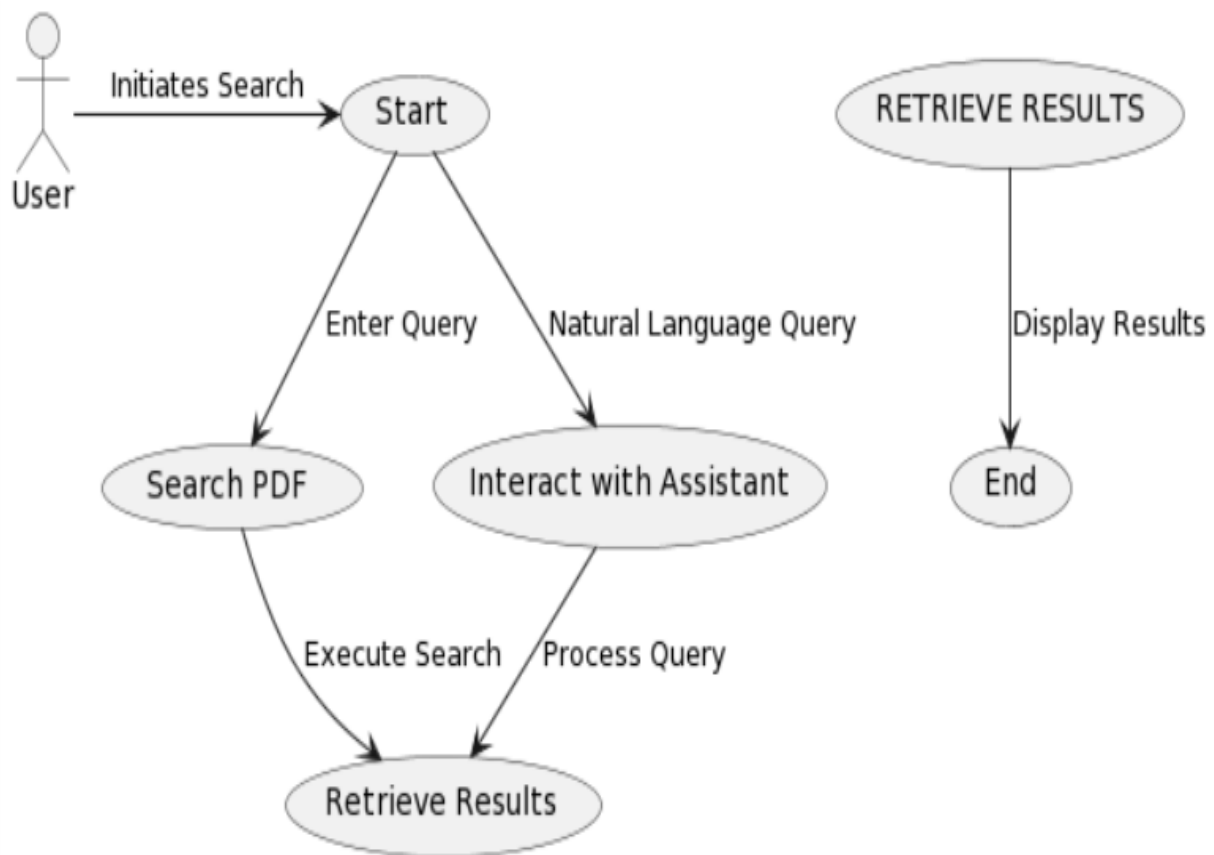


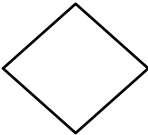
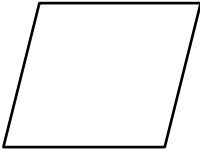



Fig 3.6.3 Activity Diagram

3.6.4 FLOWCHART DIAGRAM:

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams.

COMPONENTS:

COMPONENT NAME	SYMBOL	DESCRIPTION
Start/End		Also known as the “Terminator Symbol”, this symbol represents the start points, end points, and potential outcomes of a path. Often contains “Start” or “End” within the shape.
Process		Also known as an “Action Symbol”, this shape represents a process, action, or function. It’s the most widely-used symbol in flowcharting.
Decision		Indicated a question to be answered – usually yes/no or true/false.
Data		It represents data that is available for input or output as well as representing resources used or generated.
Flow		Used to indicate the direction of flow of control. In drawing a proper flow chart, all necessary requirements should be listed out in logical order.

3.6.4 Flow chart Table

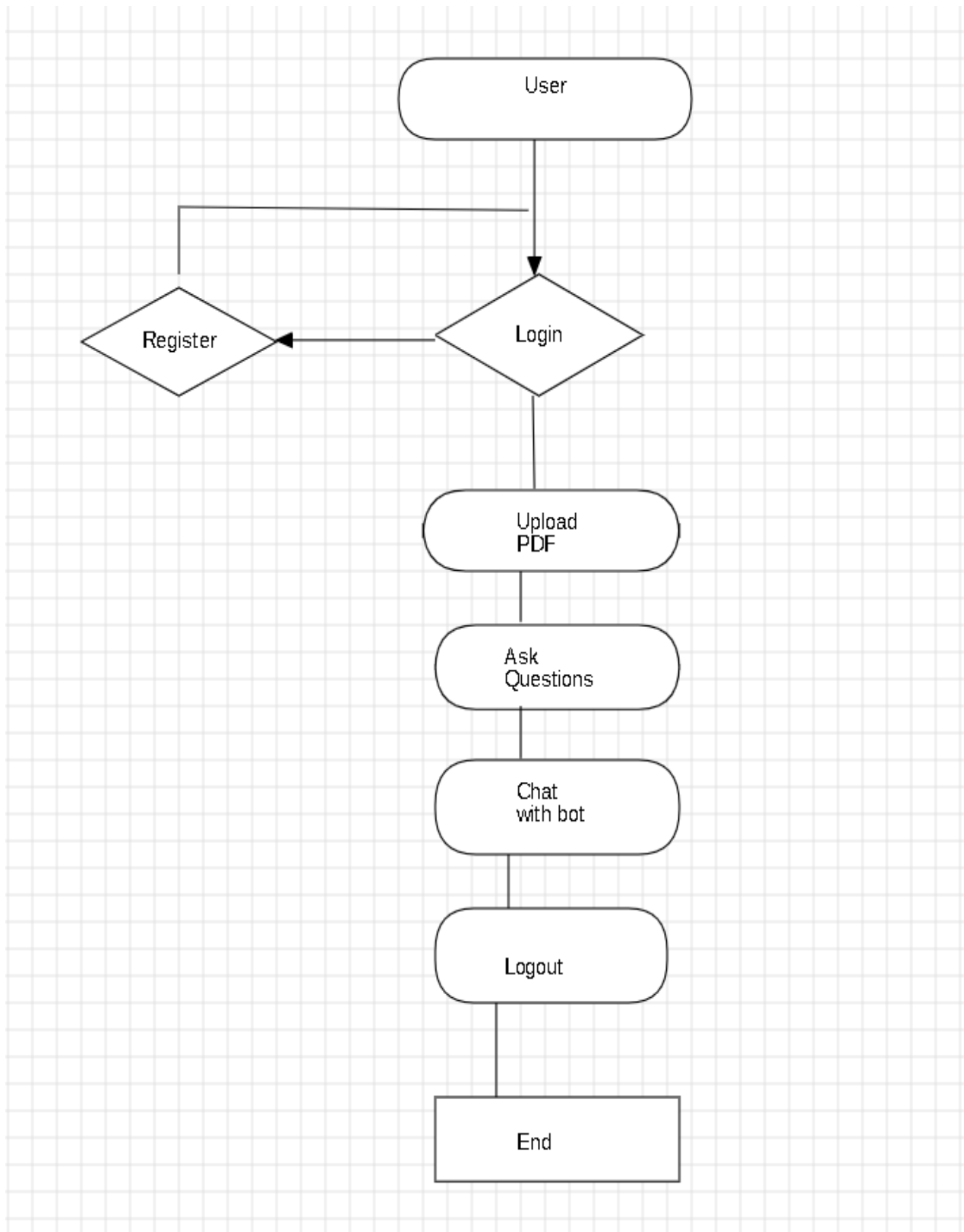


Fig No 3.6.4 Flow chart diagram

3.6.5 ENTITY RELATIONSHIP DIAGRAM (ERD)

ERD stands for entity relationship diagram. People also call these types of diagrams ER diagrams and Entity Relationship Models. An ERD visualizes the relationships between entities like people, things, or concepts in a database. An ERD will also often visualize the attributes of these entities.

By defining the entities, their attributes, and showing the relationships between them, an ER diagram can illustrate the logical structure of databases. This is useful for engineers hoping to either document a database as it exists or sketch out a design of a new database.

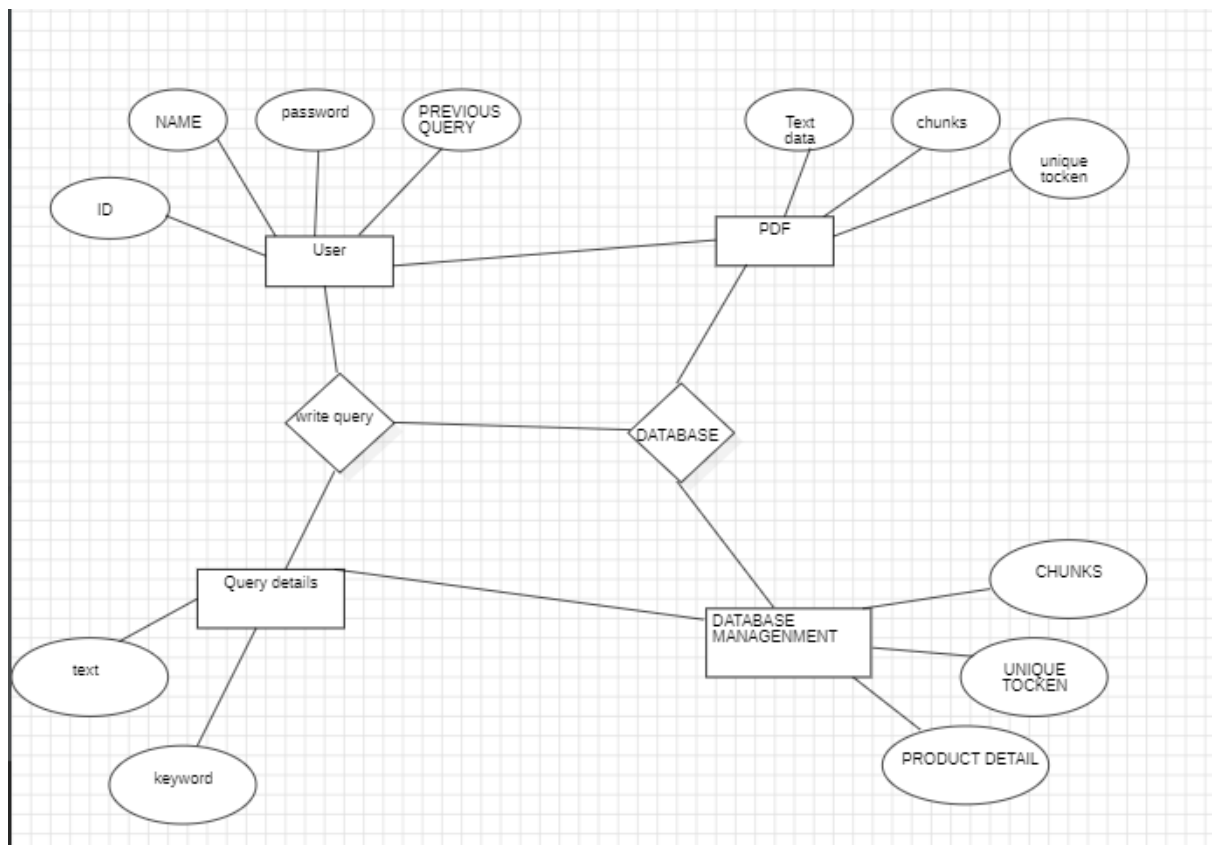


Fig No3.6.5 ER Diagram

3.6.6 LEVEL 0 DATA FLOW DIAGRAM

COMPONENT


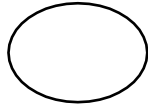

Symbol	Name	Description
	Data flow	Data flow are pipelines through the packets of information flow.
	Process	A process or a task performed by system.
	Entity	Entity are object of the system. A source or destination data of a system.

Table- 3.6.6 Level-0 DFD Table

LEVEL 0 CONTEXT LEVEL

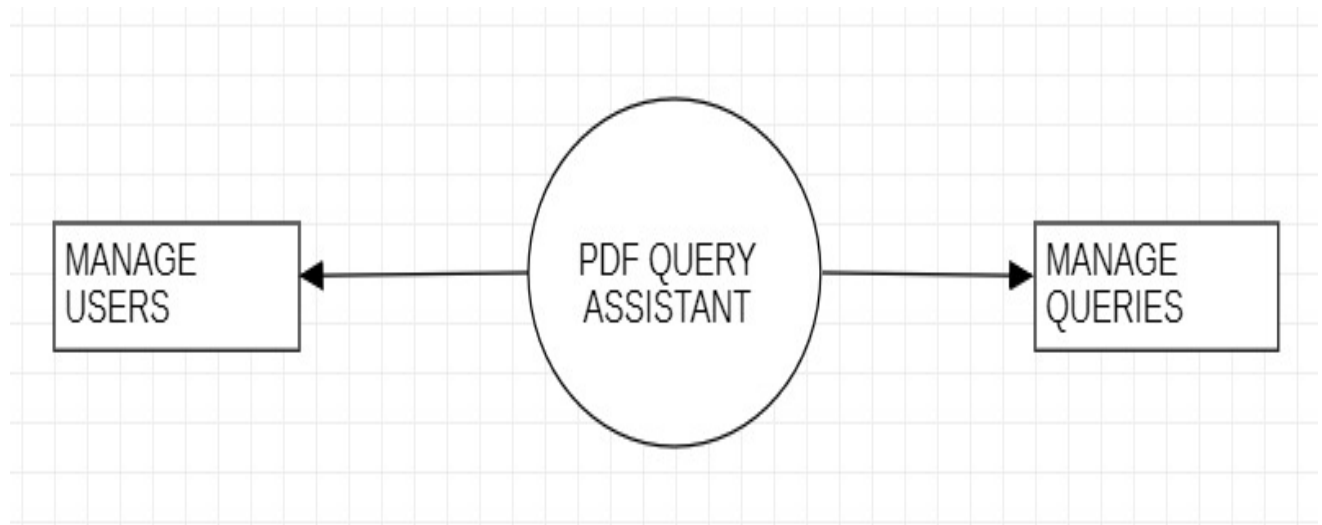
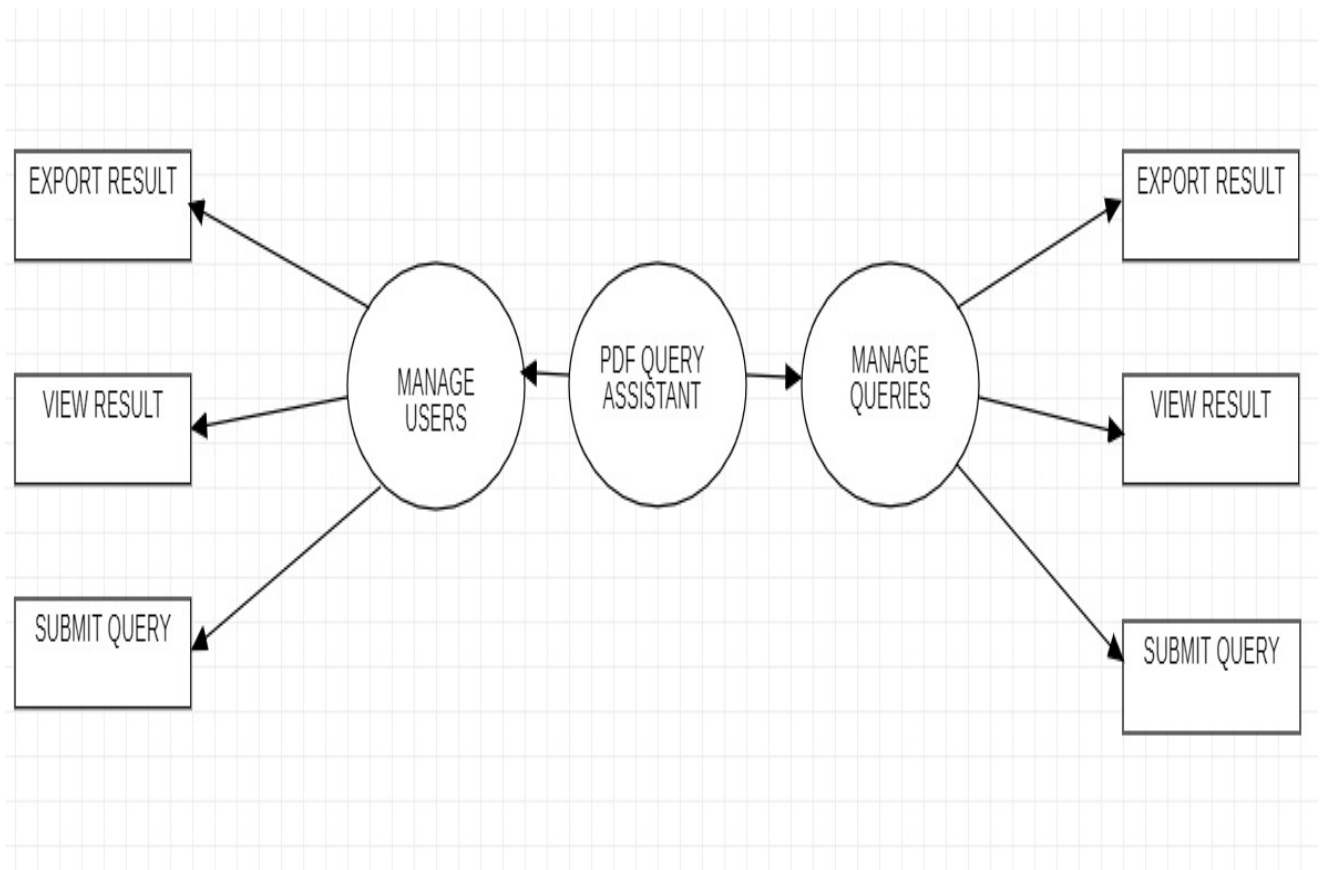


Fig No 3.6.6 Level 0 DFD

3.6.7 LEVEL 1

Fig No 3.6.7 DFD LEVEL 1



3.6.9 SEQUENCE DIAGRAM

- A Sequence Diagram is a type of interaction diagram because it describes how and in what Order a group of objects works together.
- These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.
- Sequence diagrams are sometimes known as event diagrams or event scenarios

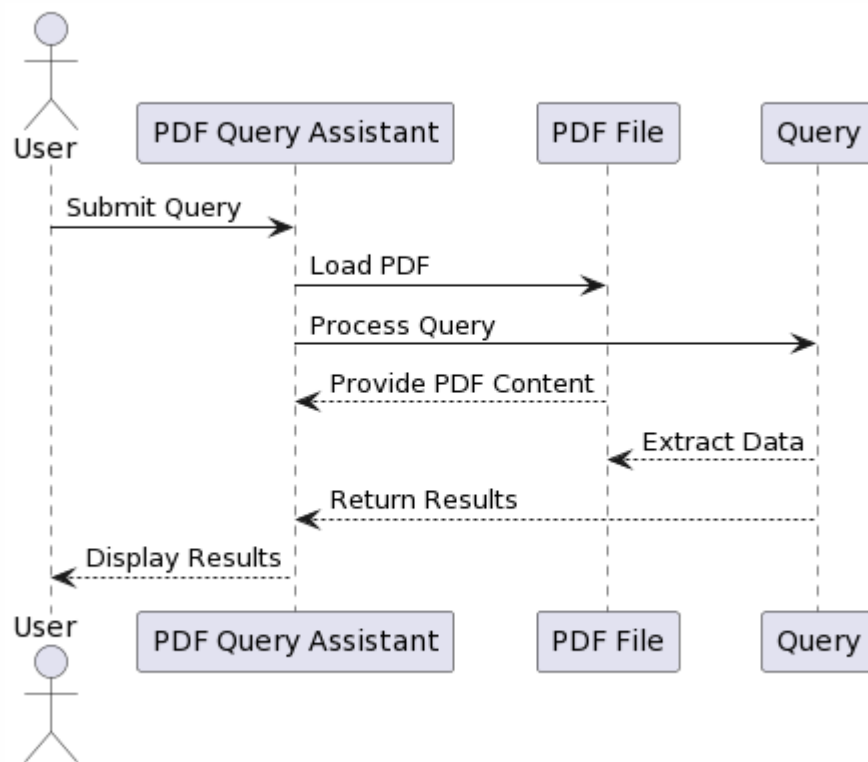


Fig 3.6.9 Sequence Diagram

CHAPTER 4 SYSTEM DESIGN

4.1 BASIC MODULES

MODULES: This project contains 1 modules, that is

1. User Module

1. User Module

The user module enables users to Authenticate, upload pdf's , and chat with the chatbot.

Here's an overview of its functionalities:

1. **Authentication:**

- User have to register themselves
- Users also have to login with valid credentials for using the websites.

2. **Upload pdf;s**

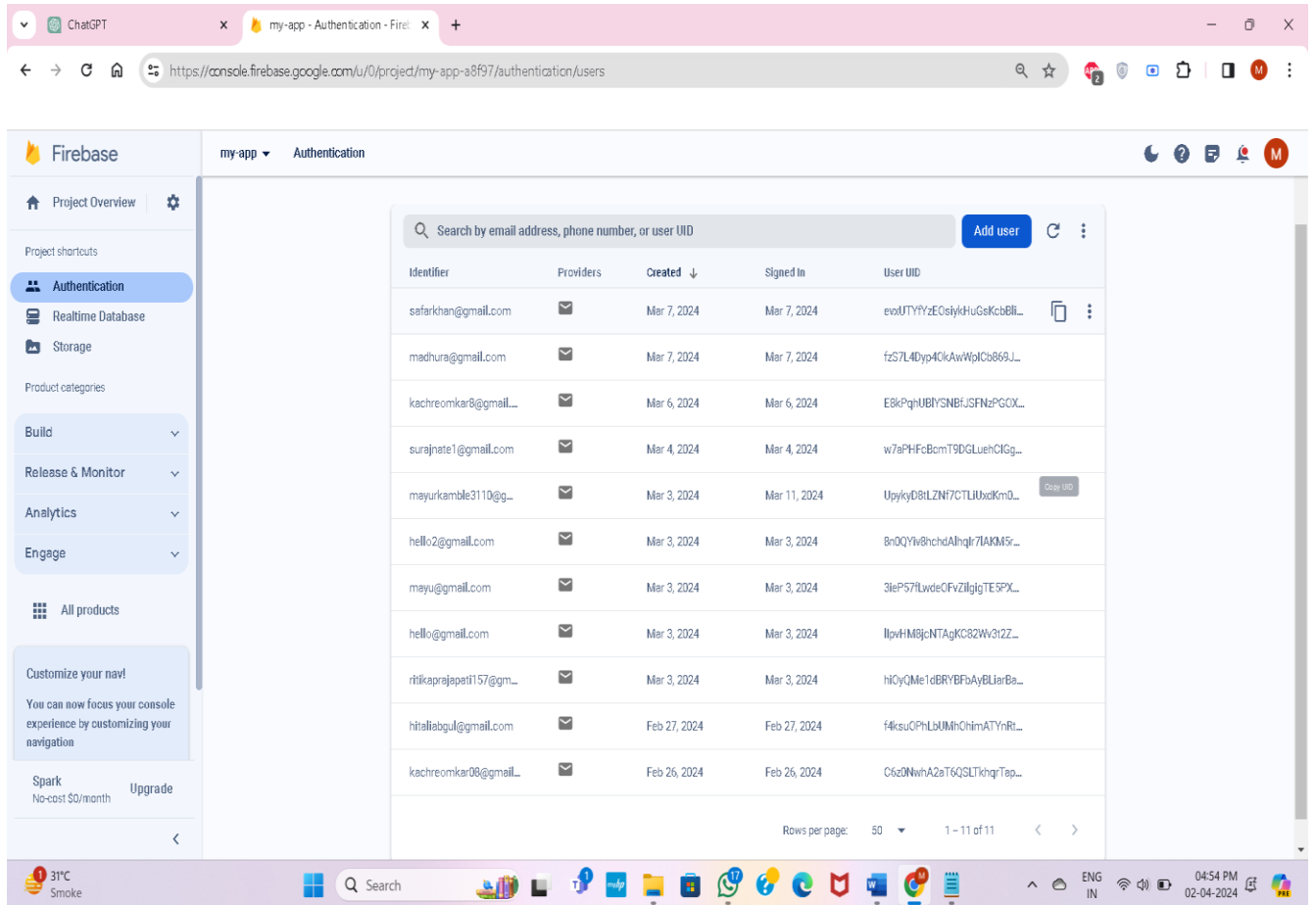
- Users can upload the pfd's ,that are of less than 200 mb
- The Document Uploded must be a pdf only

3. **Chat with Chatbot:**

- User can interact with the chat bot
- Users can ask questions related to the uploaded pdf

This streamlined approach empowers users to conduct private bidding transactions without the need for payment processing or feedback mechanisms within the application. Instead, users can engage directly with each other to negotiate and finalize deals based on their preferences.

4.2 DATA DESIGN



The screenshot displays the Firebase Authentication console for a project named 'my-app'. The left sidebar shows the 'Authentication' section selected. The main area contains a table of users with columns for Identifier, Providers, Created, Signed In, and User UID. A search bar at the top of the table allows filtering by email address, phone number, or user UID. The table lists 11 users, with the first 10 visible. The bottom of the screen shows a Windows taskbar with various application icons and system information.

Identifier	Providers	Created	Signed In	User UID
safarkhan@gmail.com	📧	Mar 7, 2024	Mar 7, 2024	ev0JTYYYzECslykHuGcKcbBil...
madhura@gmail.com	📧	Mar 7, 2024	Mar 7, 2024	fzS7L4Dyp4CkAwWpCb869J...
kachreomkar8@gmail...	📧	Mar 6, 2024	Mar 6, 2024	E8kPqHUBiYSNBJSFNzPGOX...
surajnete1@gmail.com	📧	Mar 4, 2024	Mar 4, 2024	w7ePHFcBcmT9DgUehCIGg...
mayurkamble3110@g...	📧	Mar 3, 2024	Mar 11, 2024	UpkykD8tLZN7CTLUx0Km0...
hello2@gmail.com	📧	Mar 3, 2024	Mar 3, 2024	8n0QYiv8nchdAlnqir7IAKMr...
mayu@gmail.com	📧	Mar 3, 2024	Mar 3, 2024	3leP57LwdeOFvZilgigTE5PX...
hello@gmail.com	📧	Mar 3, 2024	Mar 3, 2024	llpxHMBjcNTAgKC82Wv3i2Z...
ritikprajapati157@gm...	📧	Mar 3, 2024	Mar 3, 2024	hCQyQMe1dBRyBfAyBJarBa...
hitliabgul@gmail.com	📧	Feb 27, 2024	Feb 27, 2024	f4ksuCPHlBUMhChimATYnRt...
kachreomkar08@gmail...	📧	Feb 26, 2024	Feb 26, 2024	C6z0NwhA2aT6QSLtkhgTap...

4.3 PROCEDURAL DESIGN

4.3.1 LOGIC DIAGRAM

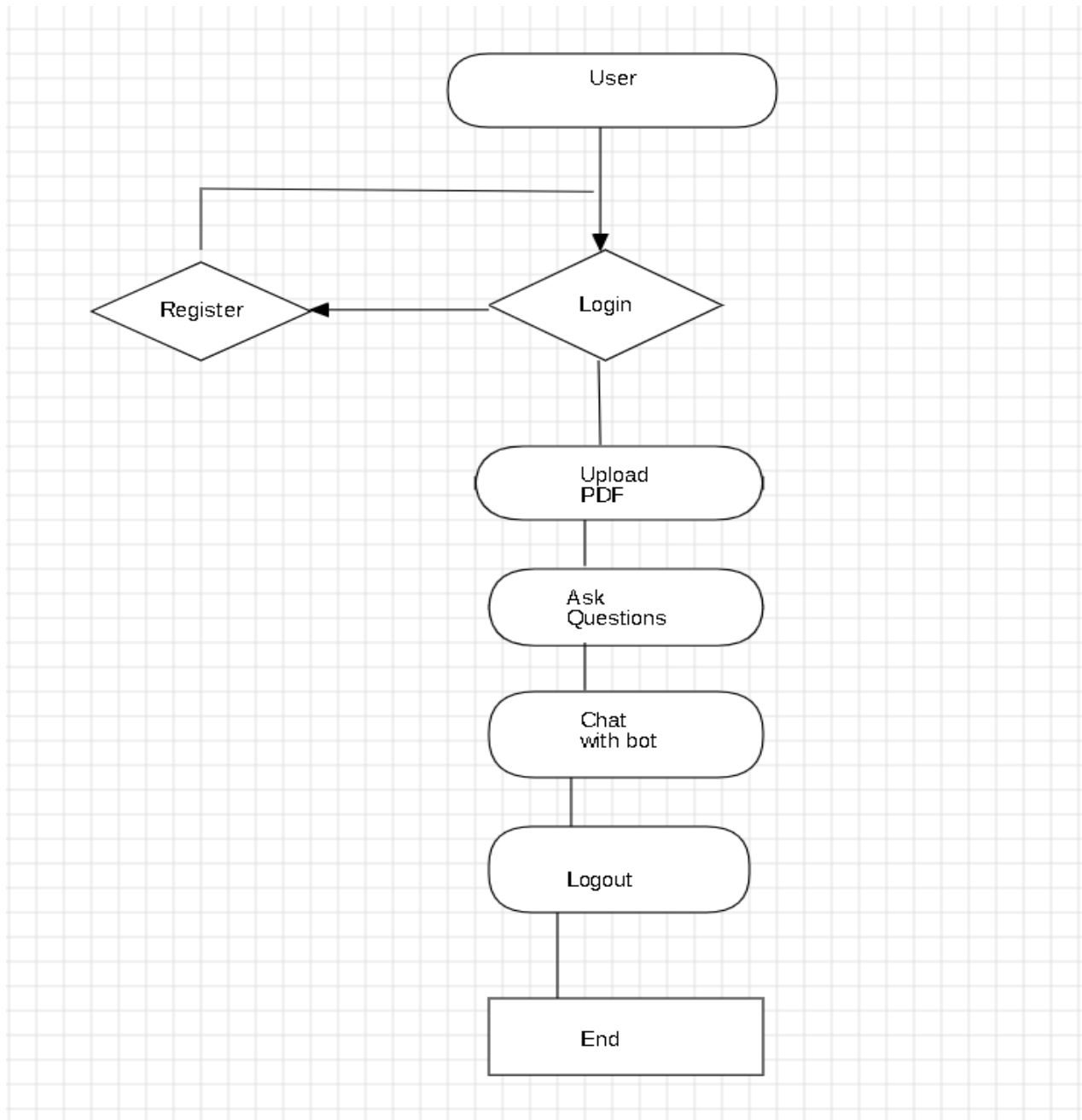


Fig No 4.3.1.1 Logic Diagram

4.3.2 ALGORITHM DESIGN FOR USERS

- Step 1: User can Go to Application directed to home page.
- Step 2: On home page user can see Navigation bar , Home page, About us page and the Logout page
- Step 3: On the home page user can upload pdf and chat with chatbot
- Step 4: User can add pdf upto 200mb size
- Step 5: User can give feedback on the contact us page
- Step 6: Users can verify themselves
- Step 7: User can close the Application. Or simply logout

4.4 USER INTERFACE DESIGN

OUTPUT REPORT FORMS: MAIN PAGE

X

Your documents

Upload your PDFs here and click on 'Process'

Drag and drop files here

Limit 200MB per file

Browse files

Process

Deploy

Chat with multiple PDFs

Ask a question about your documents:

LOGIN PAGE

Welcome to PDF-Query-Assistant

Login/Signup

Login



Please enter your email

Please enter your password



Login

4.5 SECURITY ISSUES

Issues	Solutions
User data	Data will be kept in encrypted form in the database and accessible only by users and owner of database
User password	Access will be given only to the valid users
User privileges	User can assess only his account but those who have access to database can access the account

Table No 4.5 Security issues table

4.6 TEST CASE DESIGN

Sr. No.	Test Case	Input	Expected Output
1.	Enter Full Name	Null / abc@12	Enter valid name
		xyz	Name accepted
2.	Enter Contact number	1245 / 9088rt	Contact number should be 10 digit only and only numeric characters.
		9876543210	Contact number accepted.
3.	Enter Email Id	xyz.gmail.com	Enter valid email.
		abc16@gmail.com	Email accepted

Table No: 4.6 Test case table

CHAPTER 5 IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION APPROACH

† BIG BANG MODEL

The Big Bang model is an SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement.

This Big Bang Model does not follow a process/procedure and there is a very little planning required. Even the customer is not sure about what exactly he wants and the requirements are implemented on the fly without much analysis. Usually, this model is followed for small projects where the development teams are very small. The Big Bang Model comprises of focusing all the possible resources in the software development and coding, with very little or no planning. The requirements are understood and implemented as they come. Any changes required may or may not need to revamp the complete software.

This model is ideal for small projects with one or two developers working together and is also useful for academic or practice projects. It is an ideal model for the product where requirements are not well understood and the final release date is not given. The advantage of this Big Bang Model is that it is very simple and requires very little or no planning. Easy to manage and no formal procedure are required.

However, the Big Bang Model is a very high-risk model and changes in the requirements or misunderstood requirements may even lead to complete reversal or scraping of the project. It is ideal for repetitive or small projects with minimum risks. The advantages of the Big Bang Model are as follows –

- This is a very simple model
- Little or no planning required
- Easy to manage
- Very few resources required

5.2 CODING DETAILS AND CODING EFFICIENCY

Login.py

```
import pyrebase

import streamlit as st


# Firebase Configuration
firebaseConfig = {
    'apiKey': "AIzaSyAUZrT-S2V48F2a5wihJ6aOQ7kvXX2sBwk",
    'authDomain': "my-app-a8f97.firebaseio.com",
    'projectId': "my-app-a8f97",
    'databaseURL': "https://my-app-a8f97-default-rtdb.europe-west1.firebaseio.com",
    'storageBucket': "my-app-a8f97.appspot.com",
    'messagingSenderId': "656510339673",
    'appId': "1:656510339673:web:adbeb7f1d31be09364bf48",
    'measurementId': "G-LM8F2GFQNJ"
}


# Firebase Initialization
firebase = pyrebase.initialize_app(firebaseConfig)
auth = firebase.auth()
db = firebase.database()


# Login function
def login(email, password):
    try:
        user = auth.sign_in_with_email_and_password(email, password)
        handle = db.child(user['localId']).child("Handle").get().val()
```

```

    if handle is not None:
        st.subheader('Welcome ' + handle)
        st.info('Navigate using the menu on the left.')
        st.session_state.username = user['localId']
        st.session_state.usermail = user['email']
        st.session_state.signout = True
        st.session_state.selectbox = False

    except :
        st.error("Login Failed. Error: ")

def logout():
    st.session_state.signout = False
    st.session_state.username = False #"
    st.session_state.usermail = "

def app():
    st.title("Welcome to PDF-Query-Assistant")

    if 'username' not in st.session_state:
        st.session_state.username = ""
    if 'usermail' not in st.session_state:
        st.session_state.usermail = "None"
    if 'signout' not in st.session_state:
        st.session_state.signout = False

    if not st.session_state.signout:
        choice = st.selectbox("Login/Signup", ['Login', 'Signup'])

```

```

email = st.text_input("Please enter your email")
password = st.text_input("Please enter your password", type='password')

if choice == 'Signup':
    handle = st.text_input("Enter a unique username", value='Default')
    submit = st.button("Signup")
    if submit:
        try:
            user = auth.create_user_with_email_and_password(email, password)
            db.child(user['localId']).child("Handle").set(handle)
            st.success("Your account is created successfully")
            st.balloons()
        except :
            st.error("Signup Failed. Error: " )

if choice == 'Login':
    if st.button("Login"):
        login(email, password)

if st.session_state.signout:
    if st.button('Signout'):
        logout()

app()

```


main.py

```
import streamlit as st
from streamlit_option_menu import option_menu
import home, about, contact, login

class MultiApp:

    def __init__(self):
        self.apps = []

    def add_app(self, title, func):

        self.apps.append({
            "title": title,
            "function": func
        })

    def run():
        # app = st.sidebar(
        with st.sidebar:
            app = option_menu(
                menu_title='PDF Query Assistant ',
                options=['Profile','Home','About','Contact'],
                icons=['house-fill','person-circle','trophy-fill','chat-fill','info-circle-fill'],
                menu_icon='chat-text-fill',
                default_index=0,
                styles={
```

```

        "container": {"padding": "5!important","background-color":'black'},
        "icon": {"color": "white", "font-size": "23px"},
        "nav-link": {"color":"white","font-size": "20px", "text-align": "left", "margin":"0px", "--
hover-color": "blue"},
        "nav-link-selected": {"background-color": "#02ab21"},}
    )
    if app == "Profile":
        login.app()
    if app == "Home":
        home.main()
    if app == "Contact":
        contact.app()
    if app == 'About':
        about.app()
run()

```

```

Home.py
import streamlit as st

from dotenv import load_dotenv # used to load the api key from .env file
from PyPDF2 import PdfReader
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings import OpenAIEmbeddings, HuggingFaceInstructEmbeddings
from langchain.vectorstores import FAISS
from langchain.chat_models import ChatOpenAI
from langchain.memory import ConversationBufferMemory
from langchain.chains import ConversationalRetrievalChain
from htmlTemplates import css, bot_template, user_template # used to access the htmlTemplet.py
contents

from langchain.llms import HuggingFaceHub

def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader = PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text

def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )

```

```

    )
    chunks = text_splitter.split_text(text)
    return chunks

def get_vectorstore(text_chunks):
    # embeddings = OpenAIEmbeddings()
    embeddings = HuggingFaceInstructEmbeddings(model_name="hkunlp/instructor-xl")
    vectorstore = FAISS.from_texts(texts=text_chunks, embedding=embeddings)
    return vectorstore

def get_conversation_chain(vectorstore):
    # llm = ChatOpenAI()
    llm=HuggingFaceHub(repo_id="google/flan-t5-xxl",model_kwargs={"temprature":0.5,"max_length":512})

    memory = ConversationBufferMemory(
        memory_key='chat_history', return_messages=True)
    conversation_chain = ConversationalRetrievalChain.from_llm(
        llm=llm,
        retriever=vectorstore.as_retriever(),
        memory=memory
    )
    return conversation_chain

def handle_userinput(user_question):
    response = st.session_state.conversation({'question': user_question})
    st.session_state.chat_history = response['chat_history']

```

```

for i, message in enumerate(st.session_state.chat_history):
    if i % 2 == 0:
        st.write(user_template.replace(
            "{{MSG}}", message.content), unsafe_allow_html=True)
    else:
        st.write(bot_template.replace(
            "{{MSG}}", message.content), unsafe_allow_html=True)

def main():
    load_dotenv()
    #st.set_page_config(page_title="Chat with multiple PDFs",
    #                    page_icon=":books:")
    st.write(css, unsafe_allow_html=True)

    if "conversation" not in st.session_state:
        st.session_state.conversation = None
    if "chat_history" not in st.session_state:
        st.session_state.chat_history = None

    st.header("Chat with multiple PDFs :books:")
    user_question = st.text_input("Ask a question about your documents:")
    if user_question:
        handle_userinput(user_question)

    with st.sidebar:
        st.subheader("Your documents")
        pdf_docs = st.file_uploader(
            "Upload your PDFs here and click on 'Process'", accept_multiple_files=True)

```

```
if st.button("Process"):
    with st.spinner("Processing"):
        # get pdf text
        raw_text = get_pdf_text(pdf_docs)

        # get the text chunks
        text_chunks = get_text_chunks(raw_text)

        # create vector store
        vectorstore = get_vectorstore(text_chunks)

        # create conversation chain
        st.session_state.conversation = get_conversation_chain(vectorstore)

if __name__ == '__main__':
    main()
```

```

htmlTemplates.py
import streamlit as st

from dotenv import load_dotenv # used to load the api key from .env file
from PyPDF2 import PdfReader
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings import OpenAIEmbeddings, HuggingFaceInstructEmbeddings
from langchain.vectorstores import FAISS
from langchain.chat_models import ChatOpenAI
from langchain.memory import ConversationBufferMemory
from langchain.chains import ConversationalRetrievalChain
from htmlTemplates import css, bot_template, user_template # used to access the htmlTemplet.py
contents

from langchain.llms import HuggingFaceHub

def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader = PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text

def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )

```

```

)

chunks = text_splitter.split_text(text)

return chunks


def get_vectorstore(text_chunks):
    # embeddings = OpenAIEmbeddings()
    embeddings = HuggingFaceInstructEmbeddings(model_name="hkunlp/instructor-xl")
    vectorstore = FAISS.from_texts(texts=text_chunks, embedding=embeddings)

    return vectorstore


def get_conversation_chain(vectorstore):
    # llm = ChatOpenAI()

    llm=HuggingFaceHub(repo_id="google/flan-t5-
xxl",model_kwargs={"temprature":0.5,"max_length":512})


    memory = ConversationBufferMemory(
        memory_key='chat_history', return_messages=True)
    conversation_chain = ConversationalRetrievalChain.from_llm(
        llm=llm,
        retriever=vectorstore.as_retriever(),
        memory=memory
    )

    return conversation_chain


def handle_userinput(user_question):
    response = st.session_state.conversation({'question': user_question})
    st.session_state.chat_history = response['chat_history']

```



```

for i, message in enumerate(st.session_state.chat_history):
    if i % 2 == 0:
        st.write(user_template.replace(
            "{{MSG}}", message.content), unsafe_allow_html=True)
    else:
        st.write(bot_template.replace(
            "{{MSG}}", message.content), unsafe_allow_html=True)

def main():
    load_dotenv()
    #st.set_page_config(page_title="Chat with multiple PDFs",
    #                    page_icon=":books:")
    st.write(css, unsafe_allow_html=True)

    if "conversation" not in st.session_state:
        st.session_state.conversation = None
    if "chat_history" not in st.session_state:
        st.session_state.chat_history = None

    st.header("Chat with multiple PDFs :books:")
    user_question = st.text_input("Ask a question about your documents:")
    if user_question:
        handle_userinput(user_question)

    with st.sidebar:
        st.subheader("Your documents")
        pdf_docs = st.file_uploader(
            "Upload your PDFs here and click on 'Process'", accept_multiple_files=True)

```

```
if st.button("Process"):
    with st.spinner("Processing"):
        # get pdf text
        raw_text = get_pdf_text(pdf_docs)

        # get the text chunks
        text_chunks = get_text_chunks(raw_text)

        # create vector store
        vectorstore = get_vectorstore(text_chunks)

        # create conversation chain
        st.session_state.conversation = get_conversation_chain(vectorstore)

if __name__ == '__main__':
    main()
```

Contactus.py

```
import streamlit as st
```

```
def app():
    st.write("you can contact us on mayurkamble3110@gmail.com")
```

5.3 TESTING APPROACH

Integration Testing: -

System Integration Test#1:

User System to application server, 3 tire architecture

Purpose	To ensure that the client Presentation successfully connects to the application server
External Dependencies	Ensure that the client presentation is connected to the internet.
Test Description	Run the test using a client system running on Windows OS and make sure it can reach to the application server to browse to catalogue.
Expected Results	User should be able to view the full list of items presented in the category, no error messages to appear.

5.3.1 System Integration Test 1

System Integration Test#1.1

User system to application server, 3 tier architecture

Purpose	The purpose of this test is to ensure that the client Presentation successfully connects to the application server
External Dependencies	Ensure that the client presentation is connected to the internet
Test Description	Run the test using a client system running on android OS and make sure we can reach to the application server to browse to catalogue.
Expected Results	The user should be able view the full list of items presented in the catalogue, no error messages to appear.

5.3.2 System Integration Test 1.1

System Integration Test#2.0

Application server to database server, 3 tier architecture

Purpose	To ensure that the application server successfully connects to the database server
External Dependencies	Application and database servers should be online and connected to the internet
Test Description	Run the test using a User system running on Mobile and make sure we are able to reach to the application server category and add the items available for the bidding, this needs the application server to interact with the database server to retrieve the data on availability of the item user requires to check out and an active connection is established between the application and database server.
Expected Results	The user should be able to view the full list of category and able to chat with other user, No error messages to appear

5.3.3 System Integration Test 2.0

System Integration Test# 2.1

Application server to database server, 3 tier architecture

Purpose	To ensure that the application server successfully connects to the database server
External Dependencies	Application and database servers should be online and connected to the internet
Test Description	Run the test using a client system running on android OS and make sure we are able to reach to the application category and add the items available for the bidding, this needs the application server to interact with the database server to retrieve the data on availability of the item user requires to check out and an active
	connection is established between the application and database server.
Expected Results	The user should be able to view the full list of category and able to chat with other user, No error messages to appear

5.3.4 System Integration Test 2.1

System Integration Test# 3.0

Client – application server to database server, 3 tier architecture

Purpose	To ensure that the system is fully functional to establish the connection between the client Presentation, application server and database servers.
External Dependencies	Ensure that the client, application server and database servers are all connected to the internet.
Test Description	User should be able to log into the system via a client system, browse the category on the application server and should be able to retrieve the data from the database server to add the item for the bidding
Expected Results	The user should be able to successfully navigate through the application to every area from login to log out, no error messages to appear

5.3.5 System Integration Test 3.0

REGRESSION TESTING

Regression testing is a software testing practice that ensures an application still functions as expected after any code changes, updates, or improvements. Regression testing is responsible for the overall stability and functionality of the existing feature.

- While developing we made several changes to add more functionality and fix errors that cropped up while updating and integrating different modules together.
- To ensure the changes didn't result in errors in the functioning part of the website, we subjected the website to strenuous regression cycles to ensure correctness at all times. We implemented this testing on admin module for order detail

5.4 MODIFICATIONS AND IMPLEMENTATION

- During creating and developing this application i made several changes.
- The GUI of the user-interation was changed and modified as per the requirements.
- I have focus on chatbot system because user mainly depends on that
-

5.5 TEST CASES

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

Test Case ID	Test Case Description	Test Steps	Expected Result	Pass/Fail
TC-01	PDF Parsing	1. Provide a text-only PDF for parsing . 3. Provide an encrypted PDF for parsing.	1. Software successfully parses the text-only PDF 3. Software successfully parses the encrypted PDF.	Pass
TC-02	Query Input	1. Enter a natural language query. 2. Enter a keyword-based query. 3. Enter a regular expression query. 4. Enter an invalid query.	1. Software accepts and processes the natural language query. 2. Software accepts and processes the keyword-based query. 3. Software accepts and processes the regular expression query. 4. Software provides a clear error message for the invalid query.	Pass
TC-03	Query Execution	1. Execute a query and check if it returns relevant results. 2. Execute a complex query and check if it returns accurate results. 3. Execute multiple queries simultaneously.	1. The software executes the query and returns relevant results. 2. The software executes the complex query and returns accurate results. 3. The software can handle concurrent queries without errors.	Pass
TC-04	Performance	1. Load a large PDF file and execute a query. 2. Execute multiple queries in rapid succession.	1. The software handles a large PDF file and query execution without crashing or excessive resource usage. 2. The software executes multiple queries without significant performance degradation.	Pass

CHAPTER 6 RESULTS AND DISCUSSIONS

6.1 OUTPUT

Login Page:

Welcome to PDF-Query-Assistant

Login/Signup

Login



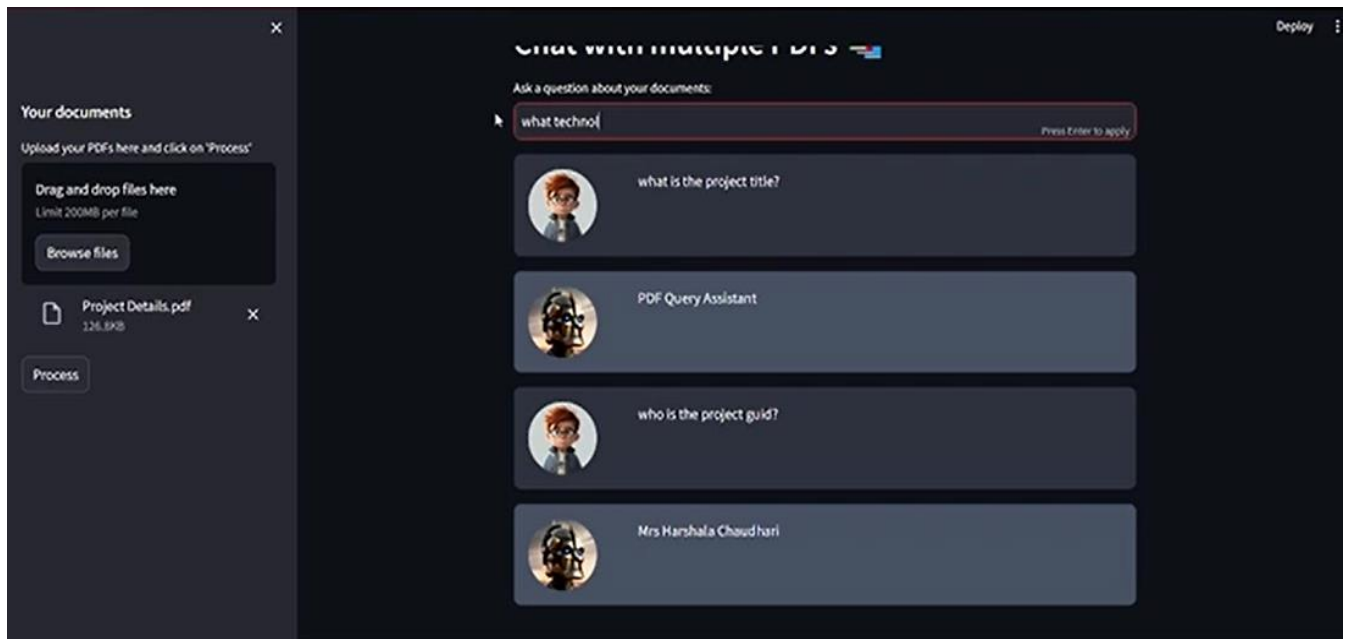
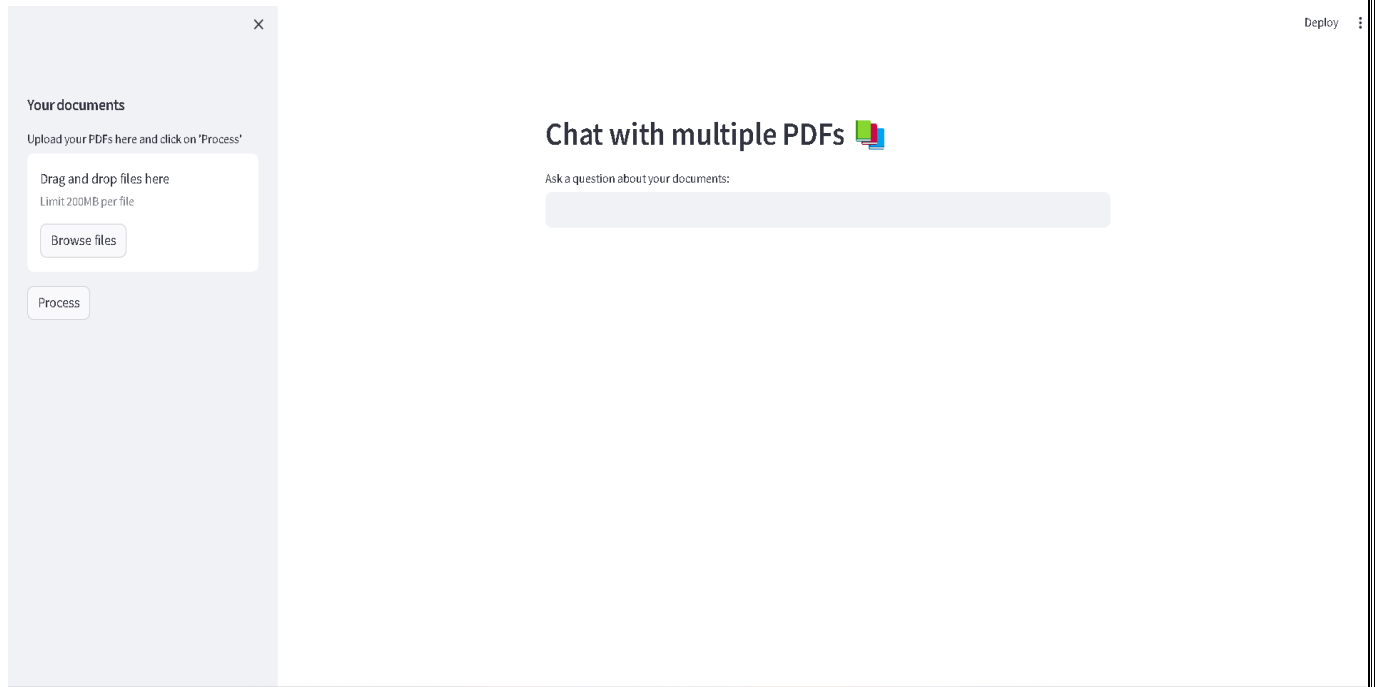
Please enter your email

Please enter your password



Login

Home Page:



6.2 User Documentation

- PDF Query Assistant is design to reduce the manual work in the real Auction
- For one to run this application they should have VS Code
- PDF Query Assistant, as described above, can lead to error free, secure, reliable and fast
Interaction with pdfs

CHAPTER 7 CONCLUSIONS

7.1 CONCLUSION

The PDF Query Assistant project is a comprehensive software application tailored to streamline the manipulation and analysis of PDF documents. Its primary focus lies in parsing and querying PDF files with remarkable efficiency. Users can easily install the software on various operating systems, including Windows, macOS, and Linux, following straightforward installation instructions. It exhibits a versatile PDF parsing capability, enabling the handling of diverse PDF types, from simple text-only documents to scanned images and even encrypted files. Users can input queries in multiple formats, encompassing natural language, keywords, and regular expressions, and the software excels in executing these queries, delivering precise and relevant results. It can simultaneously handle multiple queries without sacrificing performance, making it adept at managing large PDF files. The user interface is designed to be intuitive, ensuring user-friendliness, and all features and functions operate as expected. Security measures are in place to identify and alert users to malicious content within PDFs, enhancing document safety. Comprehensive user documentation and tutorial resources are available to facilitate users in harnessing the software's capabilities for diverse tasks. Ultimately, the PDF Query Assistant project aims to be a versatile, efficient, and user-centric tool for seamlessly interacting with PDF documents.

7.2 LIMITATION OF THE SYSTEM

- In this project there are certain limitations where I am trying to resolve It and trying to implement and modified this project
- User can only ask question related to pdfs
- There is no Admin Page
- User cannot add there profiles which lead to bad user experience

7.3 FUTURE SCOPE OF THE PROJECT

1. **Integration of Payment Gateways:** Implementing payment gateways will allow users to buy premium and upload files which are of large size
.
2. **Feedback and Rating System:** Introducing a feedback and rating system will enable users to provide and receive feedback based on their experiences, fostering trust and accountability within the platform while helping

users make informed decisions.

3. **Advanced Search Algorithms:** By integrating advanced search algorithms, the platform can provide personalized answers and user can search the content of pdf on the websites itself

7.4 REFERENCES

Sr no	Book	Author
1	Foundation of Software testing	Dorothy Graham
2	Learning Python	Mark Lutz

7.4 Reference table

Also:

- <https://towardsdatascience.com/how-to-chat-with-any-file-from-pdfs-to-images-using-large-language-models-with-code-4bcfd7e440bc>
- [1] Devlin et al, “Introduce a new language representation model called BERT” , 2018.
- [2] Rohit Arora, “Identifies which question and answer questions should be used in a vast amount of data”, 2022.
- [3] Rohit Arora, “Developed the automatic question-answer pairs generation system”, 2022.