**REPORT GENERATION IN COPS PORTAL** **&**

**COGNITIVE SEARCH ON DOCUMENTS**

**JYOTSHSNA S**

**Register No: 2034023**

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

M.Sc. (Artificial Intelligence and Machine Learning)

OF ANNA UNIVERSITY



November 2023

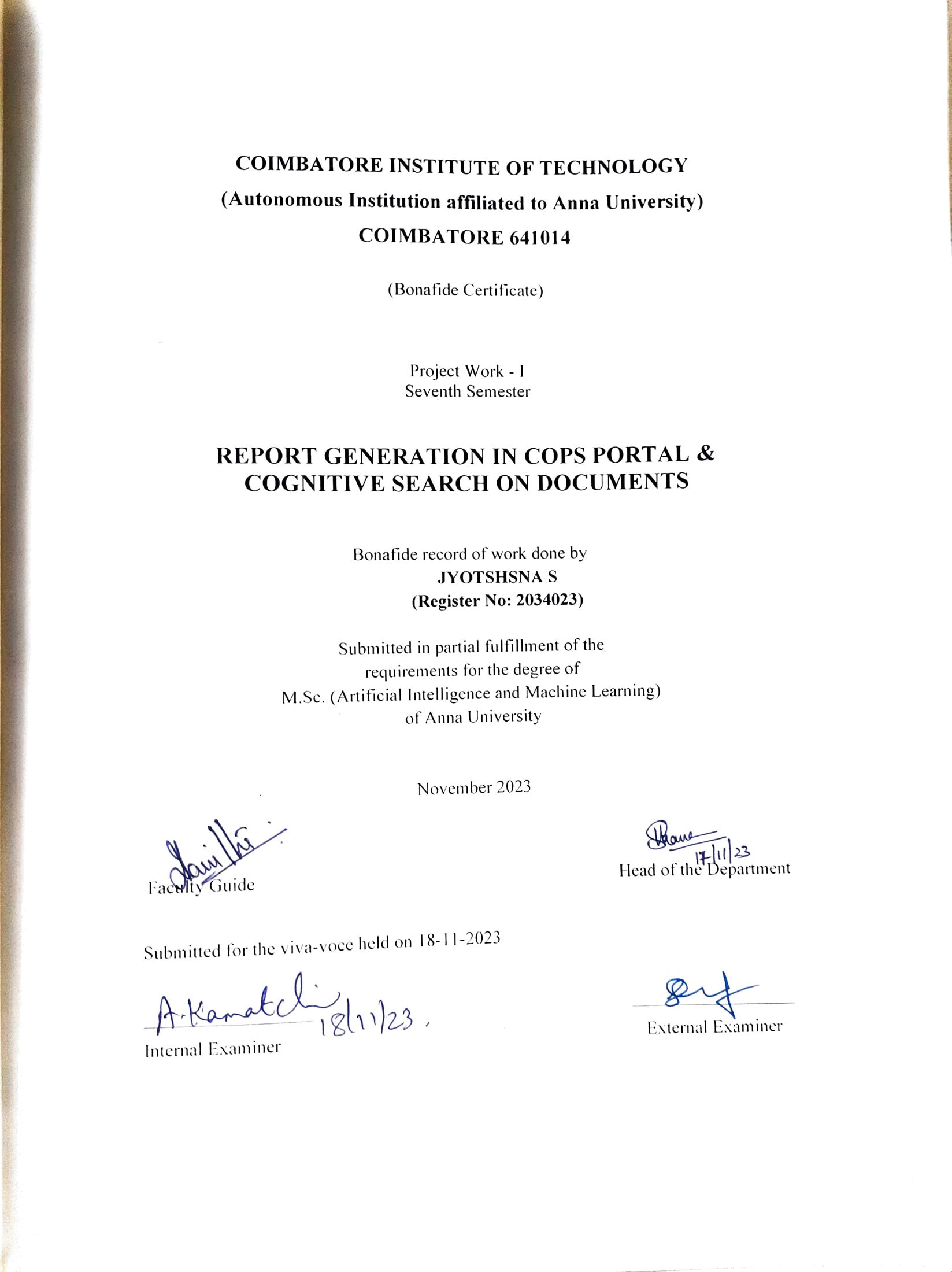
**DEPARTMENT OF COMPUTING**

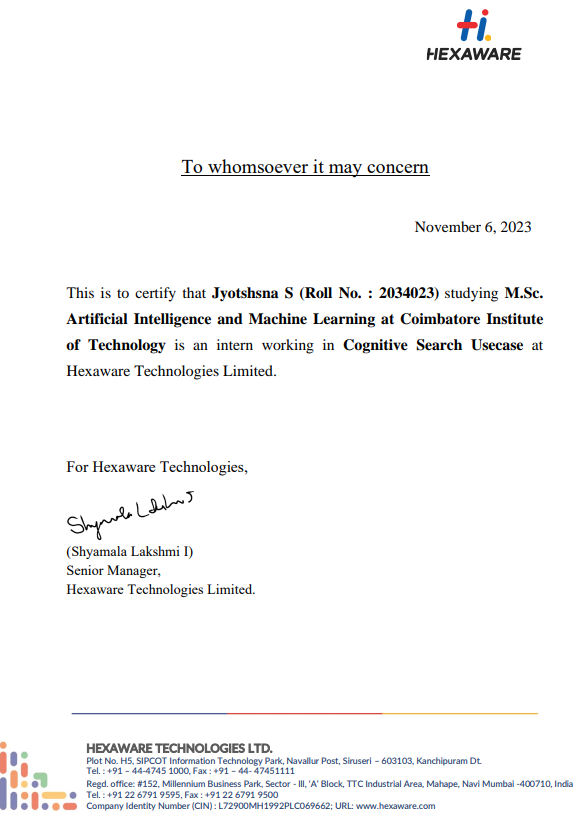
**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**COIMBATORE INSTITUTE OF TECHNOLOGY**

**(Autonomous Institution affiliated to Anna University)**

**COIMBATORE – 641014**



****

**CONTENTS**

**CHAPTER** **PAGE NO**

ACKNOWLEDGEMENT i

SYNOPSIS ii

PREFACE iii

**I INTRODUCTION 1**

1.1 ORGANIZATION PROFILE 1

1.2 PROBLEM DEFINITION 2

1.3 SYSTEM ENVIRONMENT 3

**II SYSTEM ANALYSIS 5**

2.1 SYSTEM DESCRIPTION 5

2.2 USE CASE MODEL 6

2.3 SOFTWARE REQUIREMENTS SPECIFICATION 11

2.4 TEST PLAN 13

**III SYSTEM DESIGN 16**

3.1 ARCHITECTURAL DESIGN 16

3.2 STRUCTURAL DESIGN 19

3.3 BEHAVIOURAL DESIGN 22

3.4 TABLE DESIGN 24

3.5 USER INTERFACE DESIGN 26

3.6 DEPLOYMENT DESIGN 28

3.7 NAVIGATION DESIGN 32

3.8 CODE DESIGN 33

**IV SYSTEM TESTING 35**

4.1 TEST CASES AND TEST REPORTS 36

**V SYSTEM IMPLEMENTATION 40**

**VI CONCLUSION 43**

BIBLIOGRAPHY

**ACKNOWLEDGEMENTS**

I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

First of all, I would like to thank the Principal, **Dr. A. Rajeswari, Principal, Coimbatore Institute of Technology** for her guiding light with her words of inspiration and hard work.

Secondly, I would like to extend my sincere thanks to our departmental head, **Dr. K. Sakthi Mala, Dean of Computing, Coimbatore Institute of Technology** for her generous support and encouraging us to achieve higher goals which would benefit the society.

I extend my heartfelt gratitude to my internal guide **Dr. J. Shana**, **Head In-charge, Department of Computing (Artificial Intelligence and Machine Learning), Coimbatore Institute of Technology** for her continual support and enduring guidance throughout my project tenure.

I pull out my cordial thanks to my internal guide **Dr. N. Vanitha, Assistant Professor, Department of Computing, Coimbatore Institute of Technology** for her continual support and a lasting guidance throughout my project tenure

I would also like to extend my gratitude to **Mrs.** **Hymavathi Kaleeswaran, Principal Consultant, Hexaware Technologies and Mr. Sabarish A K, Technical Architect, Hexaware Technologies** for providing me an opportunity to carry out an internship in their organization.

Finally, I express my thanks to all my teachers, well-wishers and friends who were a constant support throughout my internship period.

**SYNOPSIS**

The Catering Operations Portal also referred as *COPS Portal* is a web-based application which eases the process of managing catering facilities in an aircraft. It is designed to streamline and optimize the process of managing catering services for airlines. This portal offers a range of functionalities that enable airlines to efficiently handle everything from managing in-flight menus and placing orders to tracking inventory stocks for each flight, and ensuring cost transparency.

Designed as an in-flight catering management portal, the application benefits the stakeholders largely as it can manage the details of catering options available at each place. The reports in the web application can be saved in the local machine, shared with others for collaborations and exported to other formats. This application has been developed in React with its backend being MongoDB which is managed by nodejs environment.

The *Cognitive search application* aims to enhance the user's search experience, ensuring effectiveness and utility. This is achieved through a conversational approach, leveraging user-provided data sources. The Cognitive Search Application enables users to engage in dialogue with their data. It goes beyond simple search and answer retrieval, instead, it seeks and synthesizes responses based on user queries. The application maintains a vector store as its database, housing relevant scraped data while filtering out irrelevant information from the web. It utilizes Large Language Models to deliver accurate responses.

The application was initially developed as a web application using Jupyter Notebook and Visual Studio Code Editor as the IDE. The project is implemented with Flask (Python) and React, with ChromaDB serving as the database. The application leverages OpenAI API for creating embeddings and chains, and Langchain for QA retrieval. These systems are faster and more responsive as the usage of JavaScript functions in the system act as an underlying catalyst.

**PREFACE**

1. **INTRODUCTION**

A brief description about the organization and its objective, scope of their products in the society. Also, the objective and scope of the system, its hardware and software requirements, stakeholders and users accessing the system to be proposed here.

1. **SYSTEM ANALYSIS**

Detailed description of the proposed system and its features. The Software Requirements Specifications that deals with functional requirements, non-functional requirements, interfaces, assumptions and dependencies.

1. **SYSTEM DESIGN**

The design models that describe the architecture, behaviour, structure, deployment and the user interface are well depicted.

1. **SYSTEM TESTING**

The various testing factors and techniques along with the implementation of testing in various modules of the system, its expected and actual test results and their description summarizes the testing module of the application.

1. **SYSTEM IMPLEMENTATION**

The type of platform in which the proposed system is to be implemented and it should be accessible for the user is described.

1. **CONCLUSION**

Concluded with the salient features of the proposed systems and its characteristics, we define its future scope and the final implemented system.

**CHAPTER I**

**INTRODUCTION**

A brief description about the organization and its objective, scope of their products in the society. Also, the objective and scope of the system, its hardware and software requirements, stakeholders and users accessing the system to be proposed here. The description of development and deployment diagrams for the proposed system along with the needs that satisfy the user’s requirements.

**1.1 ORGANIZATION PROFILE**

Digital transformation that delivers enterprise-grade solutions at consumer speed impacts the whole organization. **Hexaware** is a global technology and business process services company.

Hexaware supports the Travel industry with profound domain expertise and digital tools that can continuously enhance the customer experience of the traveller by leveraging transformative technologies. It offers Airline IT solutions covering all emerging trends and technologies.

Digital transformation will improve airline functions collaboratively by connecting people, process and technology to provide enhanced customer experiences and seamless passenger journeys with our domain experience and technology expertise.

They support multinational clients and initiatives and package cost-effective Application Development, Machine Learning, Cloud Migration, Data Modernization, Data & Analytics and DevOps & Automation projects.

Hexaware, with their software and operations teams, implement agile and modern development practices with a flexible platform to automate functional releases for faster deployment. They have expertise in controlling cloud costs, forecasting usage and checking utilization rates daily, managing billing and optimizing on-demand resources.

**1.2 PROBLEM DEFINITION**

To develop a web-based platforms that administers catering operations right from orders to reports with less manual intervention using integrated modules and deliver accurate responses for user-provided queries using data source.

***1.2.1 TASK – 1***

***1.2.1.1 PROJECT OBJECTIVE***

To develop a web-based application that will streamline and optimize the catering process for airlines. The ultimate goal is to analyse the consumption trends through an effective report that enhances in-flight inventory management and predict consumption to make informed decisions, reducing food wastage and promoting sustainable practices in the aviation industry.

***1.2.1.2 SCOPE***

This application provides a platform to create, update and manage items in cart according to the inventory levels and route, to track supply-chain management activities in a flight catering service, to maintain the inventory and present/new items included in the service, to give the administrators of the organisation to have a transparent view of all activities being carried out and to produce detailed *reports* on the activities which can be exported as an excel file. It is used internally for organizing catering items to each flight and by finance team to generate reports.

***1.2.1.3 USERS***

The users of the Catering Operations portal are the authorized employees of the client and the client’s customers that is the crew. The users have the access to manage the cart items which are in turn based on the inventory levels and the route. They are also provided with detailed reports of consumption for each specific route so that they make an effective plan for the upcoming journeys. The Users can logout and remove account from the portal.

* + 1. ***TASK – 2***

***1.2.2.1 PROJECT OBJECTIVE***

• To obtain the relevant data from data sources

• To chunk each pdf article and create embeddings

• To build a vector store using the embeddings

• To enable conversational search using Large Language Models

• To create an appropriate UI for extracting and search systems

• To evaluate the answers

***1.2.2.2 SCOPE***

The Cognitive Search Application will make the search option very easy and understandable way. It also has been determined to have a permanent vector storage to store and use for long-time. The main scope is to make the unstructured data from the data sources into strategic information

***1.2.2.3 USERS***

Since the application has been created on the customized requirements of a

particular client, it can only be used by the members of the client’s organization.

*KEY USERS*

• Business analyst

• Technical team members

• Data analyst

**1.3 SYSTEM ENVIRONMENT**

The system should be designed such that it will operate on both Mac and Windows operating systems with lesser delay time. The environment of the system must have a personal computer with a minimum version of 8+ for Windows operating systems. The application operates on the personal computer, if it is connected to a network or wi-fi.

***1.3.1 HARDWARE REQUIREMENTS***

Processor and Speed : Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz

Memory : 8.00 GB or more internal memory

Screen Resolution : Monitor resolution of 1024 x 768 or higher

Storage : 64.00 GB or higher

***1.3.2 SOFTWARE REQUIREMENTS***

IDE : Visual Studio Code

Operating System : Windows 10 and above

***1.3.2.1 TASK – 1***

Front-end : ReactJs – 18.0.2

Back-end : Express Nodejs – 20.9.0

Database : MongoDB – 2.0.2

***1.3.2.2 TASK - 2***

Front-end : ReactJs – 18.0.2

Back-end : Flask (python)

Database : ChromaDB

**CHAPTER II**

**SYSTEM ANALYSIS**

Detailed description of the proposed system and its features, while briefing the limitations. Use case models portrayed as diagrams along with the suitable descriptions that explain the salient features of the system. The Software Requirements Specifications that deal with functional requirements, non-functional requirements, interfaces, assumptions and dependencies. All the functions under testing including test scope, testing functions and its techniques.

**2.1 SYSTEM DESCRIPTION**

***2.1.1 TASK – 1***

***2.1.1.1 ANALYSIS OF EXISTING SYSTEM***

The current catering operations management application which is in use is built using MVC architectural design and other traditional technologies. Abiding to the clients’ request for modernizing the application adapting to the growing technology so that the functions are built as separate microservices which can be easily migrated into cloud environment in the deployment phase. So that new features can be integrated with ease.

**2.1.1.2 *PROPOSED SYSTEM***

The proposed system is a portal which is developed using React which comprises a common client authentication module followed by a screen that displays a list of modules integrated in this application for easy navigation. This is then followed by a screen which has various options like select flight date, station and aircraft type. If an option is chosen from the list of dates the cart items are displayed for the given date. Further if a station is chosen, the system will apply a filter on the specific date depending upon the aircraft type and the filter chosen.

The cart is updated according to the inventory which module consists of a display of a list of cart items and their numbers which can be edited from time-to-time. Finally, the report module consists of a list of options like flight date, station, its type and item name for a detail specific report to be generated. The reports in the web application can be exported to other formats. A new reskinned version of the application is developed to give it a more modern look to match the web application.

***2.1.2 TASK – 2***

***2.1.2.1 ANALYSIS OF EXISTING SYSTEM AND ITS DRAWBACK***

There exist some search applications that are way similar to the Cognitive Search Applications, but they lack some of the main requirements. Such applications do not provide any permanent storage of data that will be further used for search process. Most of the search applications provide limited functionalities, such as semantic keyword search methods. Some other drawbacks are, ensuring the accuracy of the search results, maintaining the code for long time, integrating the various components involved in the search process, some of the apps directly searches the keyword and provides all the answers that may not be relevant to the query of the user.

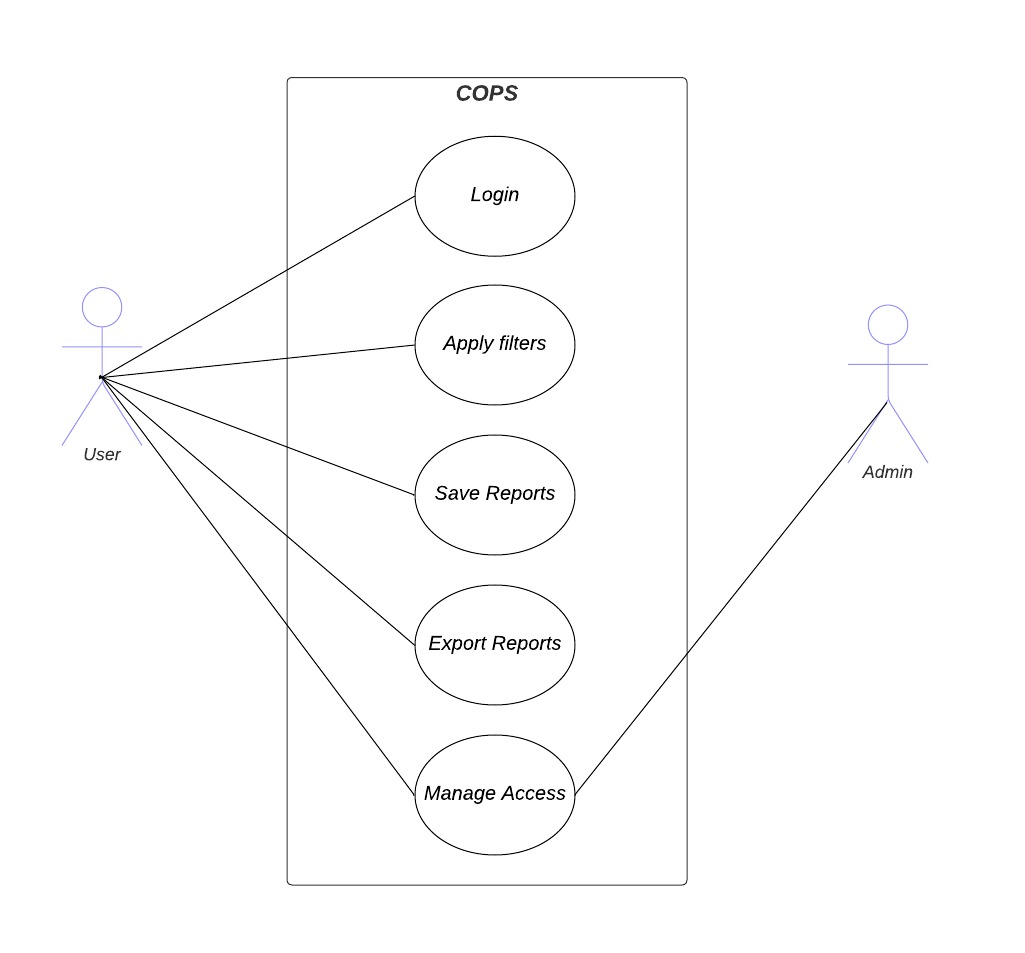
***2.1.2.2 PROPOSED SYSTEM AND ITS FEATURES***

In order to solve the drawbacks in the existing system this system has been created in a way that the data sources will be provided by the user and the information will be scraped and stored in a vector database. Large Language Models like GPT-3.5 has been used to provide answers and synthesize them in a conversational format. The answers are evaluated through manual testing method and Large Language Model assisted evaluation, to ensure the correctness of the results.

**2.2 USE CASE MODEL**

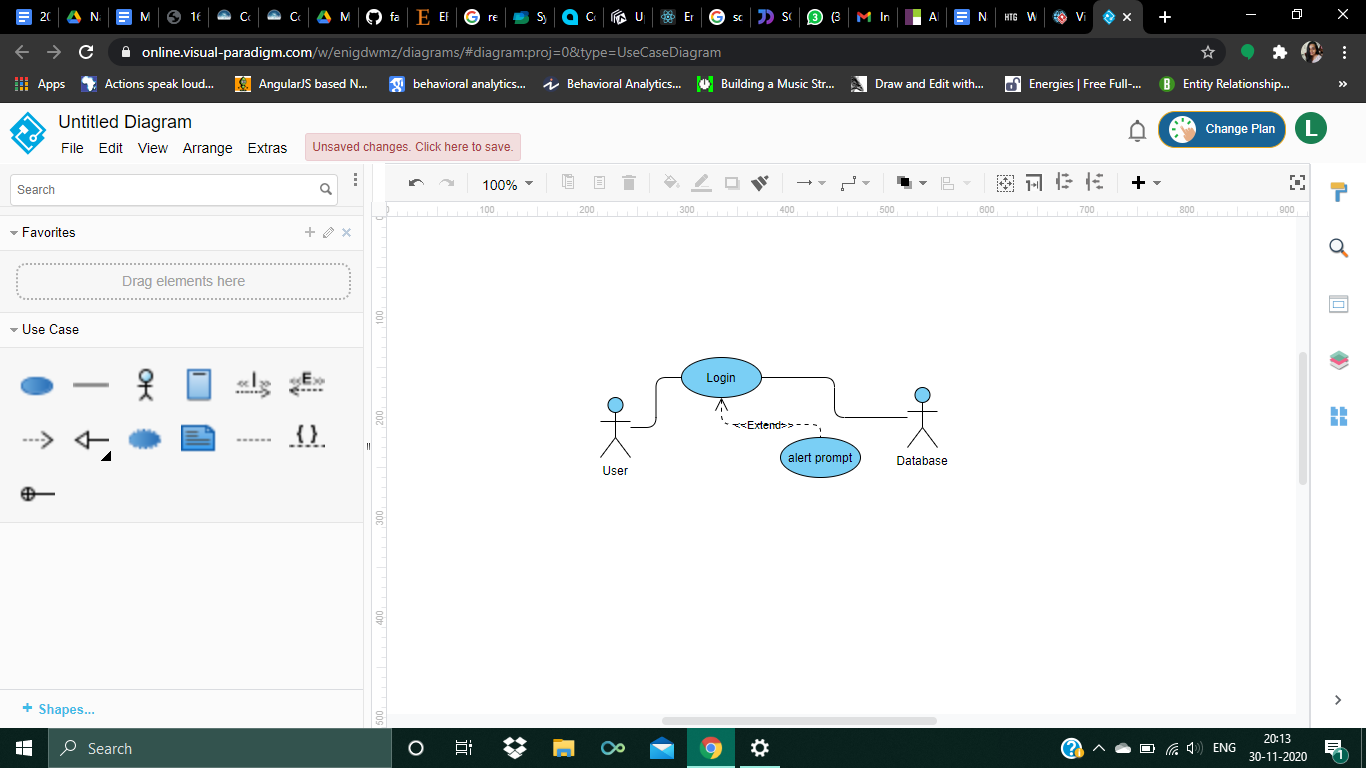
A UML use case diagram is the primary form of system/software requirements for a new software program under development. Use case diagrams, model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In addition, use case diagrams provide a good high-level analysis for outlining the system. Figure 2.1 shows the use case model of the catering operations portal.

***2.2.1 TASK - 1***

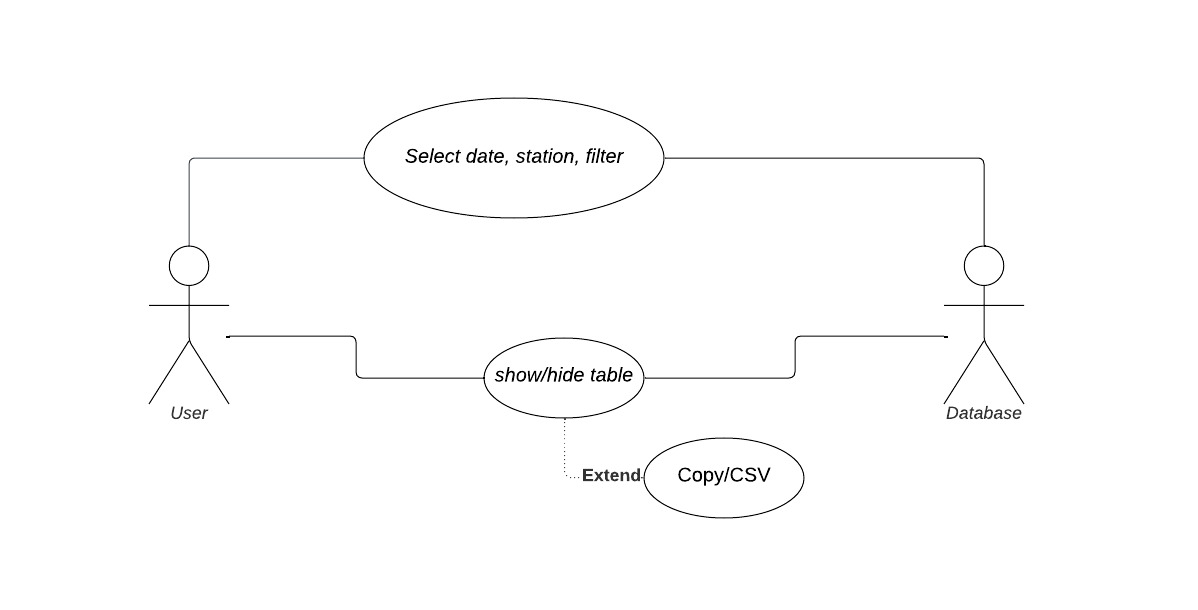


**Figure 2.1 Use case diagram of COPS application**

Figure 2.1 depicts the overall picture of the application.



**Figure 2.2 Authentication**



**Figure 2.3 Reports**

Figure 2.2 depicts the authentication procedure and the description for the use case is shown in table 2.2

Figure 2.3 depicts the function of table display over the webpage and the description for the use case is shown in table 2.3

Table 2.1 shows the description of the use case model portrayed in figure 2.1

**Table 2.1 Description of COPS Application’s use case model**

|  |  |
| --- | --- |
| Goal | To apply the vector layers corresponding to the option(s) chosen |
| Actor | User, Admin |
| Description | 1. login into the portal – report module 2. select a date, station, type and filter 3. view table 4. select item name 5. convert the table into excel |
| pre-condition | Restricted for unregistered users |
| post-condition | Logout option |

**Table 2.2 Description of COPS Application’s Authentication use case model**

|  |  |
| --- | --- |
| Goal | To provide authentication for a user |
| Actor | User, Database |
| Description | 1. credentials for username and password 2. authentication provided through the database |
| pre-condition | Restricted for unregistered users |
| post-condition | Logout option |

**Table 2.3 Description of COPS Application’s Reports use case model**

|  |  |
| --- | --- |
| Goal | To apply the vector layers corresponding to the option(s) chosen and show/hide table |
| Actor | User, Database |
| Description | 1. choose option(s) 2. show/hide table 3. Save the excel file 4. search for a data |
| pre-condition | Only for options available under filters |
| post-condition | Logout option |

***2.2.2 TASK – 2***



**Figure 2.4 Use case diagram of Cognitive Search application**

The Figure 2.1 describes the actors and their respective use cases according to the cognitive search application and the description for the use case is given in Table 2.4

**Table 2.3 Description of Cognitive Search use case model**

|  |  |
| --- | --- |
| Goal | To provide answer for user query |
| Actor | Client, Backend |
| Description | 1. login into the application 2. Input PDF 3. Enter query |
| pre-condition | Restricted for unregistered users |
| post-condition | Logout option |

**2.3 SOFTWARE REQUIREMENTS SPECIFICATION**

***2.3.1 TASK - 1***

***2.3.1.1 FUNCTIONAL REQUIREMENTS***

* A function to select the particulars like flight date, station and aircraft type.
* Buttons to show or hide the table.
* By default, date is the default filter and current date is chosen by default.
* Onclick of an item name on a particular filter, a table to be displayed with information.
* Option to convert the table data into a excel file.

***2.3.1.2 NON-FUNCTIONAL REQUIREMENTS***

* Portability - This system is portable such that it requires only a minimum level of software requirements to be met.
* Accessibility - This system is accessible on Windows, Linux and mac devices.
* Maintainability - The functionalities have been scripted in Javascript. So, changes are made quicker and easier.
* Security / Authentication - Users who are only recognised and registered from the organization are able to sign-in.
* Responsiveness - Open-source library (react) and feature-packed library, hence lessening the time taken for the system to apply filters.
* Reliability - All the data is previously stored in the MongoDB database, so there is no possibility of data failure and the filters can be applied.
* Performance - The usage of modern tools like react and node further speeds up the application along with the user interface being compatible for user.

***2.3.2 TASK – 2***

***2.3.2.1 FUNCTIONAL REQUIREMENTS***

The following are the functional requirements related to the search application:

* The system must be able to extract the text from the given PDF.
* The system should use Large Language Models (LLMs) like GPT-3.5 to generate embeddings for each text chunk.
* The system should create and maintain a vector database (chromaDB) to store the generated embeddings for efficient retrieval during search queries.
* The system should take user queries as input and retrieve relevant embeddings from the vector database using nearest neighbour search techniques.
* The answer must be synthesized and returned to the user.
* The platform must have an interface that allows clients to input their PDF.
* The user interface must have separate sections for Custom extracting and Conversational Search.

***2.3.2.2 NON-FUNCTIONAL REQUIREMENTS***

* Performance - The system should respond to user queries with minimal latency, ensuring a smooth and real-time conversational experience and the search and retrieval processes from the vector database should be optimized for high-speed performance, even with a large volume of data.
* Maintainability - The codebase should be well-structured and modular to facilitate ease of maintenance and future enhancements. Proper documents and code comments should be provided to aid future development and troubleshooting.
* Accuracy and Relevance **-** The system must provide accurate and relevant answers to user queries based on the retrieved embeddings. The search algorithm and similarity measures should be carefully chosen to ensure precision.
* Cost-Effectiveness **-** The platform should be cost-effective to operate, considering factors such as API usage, server hosting, and storage expenses.

***2.3.3 INTERFACE***

The interface is built on React as it supports development of light-weight applications and also effective cross-platform development. The code is more legible with readable and testable code. It is mainly used in developing Single Page Applications. Bugs can also be easily detected and also gives an attractive view for the user for the table displayed matches exactly with the contents of the report.

***2.3.4 ASSUMPTIONS***

The person who signs-in into the system, is assumed to be a member of the client organization who has previously registered for future purposes. The user is also assumed to be knowledgeable about the approximate numerical data.

***2.3.5 DEPENDENCIES***

The system being a web application, it relies on the network and how fast the system can query its values and display them either in pictographically manner or in text form.

**2.4 TEST PLAN**

A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. A plan typically contains a detailed understanding of the eventual workflow.

***2.4.1 TEST SCOPE***

Every functionality of the proposed system is considered important and needs to be tested for the proper working of the system. So, all the functionalities will come under the In-Scope of the system. The scope of testing the application is to make the developers understand the bugs, errors that are occurring in the application. It is also to solve the problems that may occur during the use, or at least identify the errors.

* + 1. ***TESTING PROCESS***
* **Data Testing -** Is done once a feature or bugfix request is completed. Testing is done by comparing the embedded report data with the actual report.
* **Functionality Testing -** Is done once a feature or bugfix request is complete. Testing is done by first deploying the changes to the DEV environment where the developer tests it and then it is deployed to the QA environment where the client’s team test it and the feedback from them is integrated into the next cycle.
* **System Integration Testing -** Is done once each sprint is complete. Testing is done by first deploying the changes to the DEV environment where the developer tests it and then it's deployed to the QA environment where the client’s team test it and the feedback from them is integrated into the next sprint cycle.

***2.4.3 TESTING TECHNIQUES***

* **API Testing -** API testing is a type of software testing where application programming interfaces (APIs) are tested to determine if they meet expectations for functionality, reliability, performance, and security. API testing is intended to reveal bugs, inconsistencies from the expected behaviour of an API.
* **Unit Testing -** It is a level of software testing where individual units or components of the software are tested. The purpose is to validate that each unit of the software performs as designed. Unit testing is used reduces defects in the newly developed features when changing the existing functionality.
* **Functional Testing -** It is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the functional requirements.
* **Integration Testing -** It is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in integration testing.

***2.4.5 TESTING TOOLS***

* **Postman**
  1. Postman is a tool for API developers to share, test, document and monitor APIs. With it one can,
     1. Create Collections of requests
     2. Organize your work within different Workspaces
     3. Split-stack development with Mock Servers
     4. Use Tests & Scripts to add dynamic parameters
     5. Automate tests with Collection Runner
     6. Test performance & behaviour with Monitors
     7. Document & Publish Collections for collaboration
* **QTest**
  1. QTest is used to capture all test cases. It is the process of manually testing software for defects. It requires a tester to play the role of an end-user whereby they use most of the application's features to ensure correct behavior.

Manual Testing is the most primitive of all testing types. The key concept of manual testing is to ensure that the application is error-free and it is working in conformance to the specified functional requirements. Test Suites or cases, are designed during the testing phase and should have 100% test coverage. Basically, this testing checks the quality of the system and delivers a bug-free product to the customer.

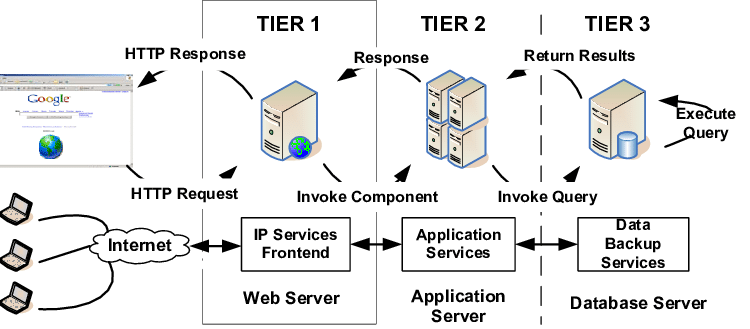
**CHAPTER III**

**SYSTEM DESIGN**

The design models that describe the architecture, behavior, structure, deployment and the user interface are well depicted. The mapping of the proposed structure to the code that is actually written is also described for better understanding. The pseudo code explains the code design added for description of the above models.

**3.1 ARCHITECTURAL DESIGN**

**3.1.1 TASK - 1**



**Figure 3.1 Architectural design (3 - tier architecture) for COPS Portal**

Figure 3.1 depicts the 3-tier architectural model used in the application.

A 3-tier architecture is a common design pattern for developing software applications.

* ***Presentation Tier:***

**Description:** The topmost layer that interacts directly with the end-user. It handles user interactions, displays information, and serves as the entry point for the application.

**Components:** React code.

**Functionality:** It provides the graphical user interface, allowing users to interact.

* ***Application Tier:***

**Description:** The application logic tier is responsible for processing and managing data, business logic, and user requests.

**Components:** API

**Functionality:** To implement the core business logic, perform data processing, handle user requests and ensure data is processed and transformed according to the requirements.

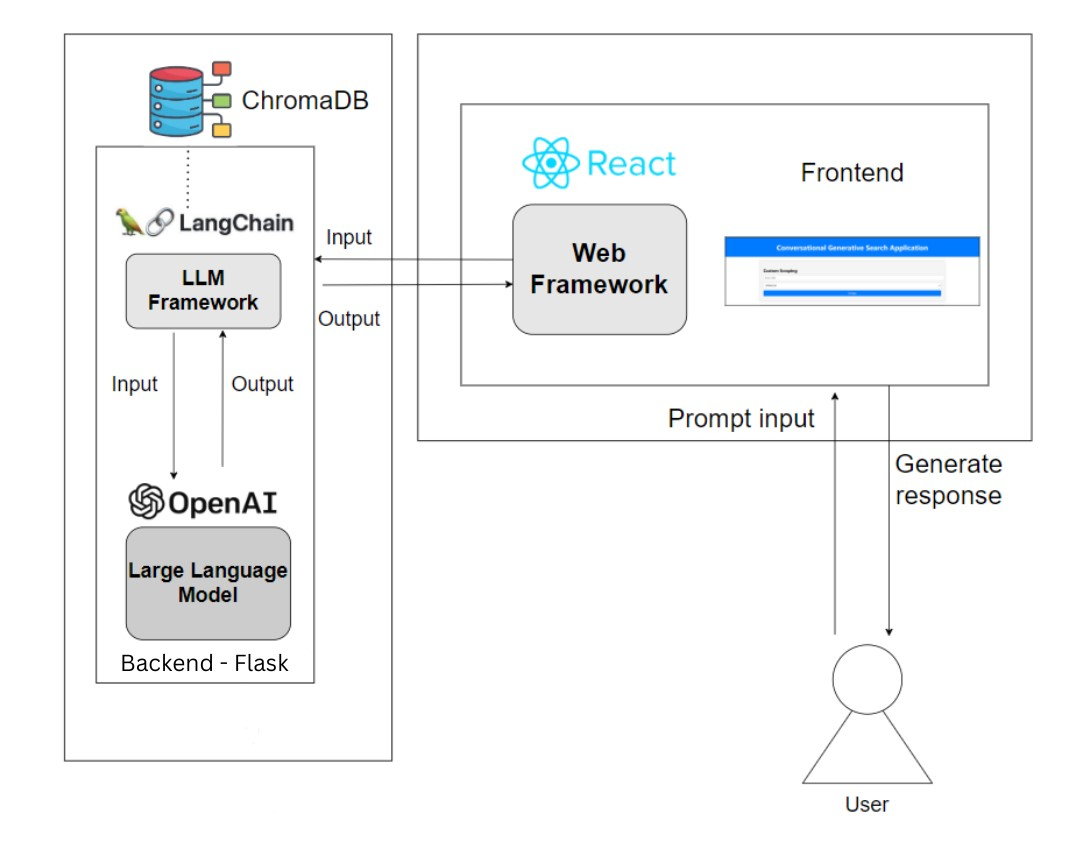
* ***Data Storage Tier:***

**Description:** It is responsible for storing and managing the application's data. It provides a structured and efficient way to store, retrieve, and manipulate data.

**Components:** MongoDB

**Functionality**: The primary function of this tier is data storage and retrieval. It maintains data consistency, durability, and availability for the application.

**3.1.2 TASK – 2**



**Figure 3.2 – Architectural Design of the Search Application**

The figure 3.2 briefly represents the system architecture of the cognitive search application.

* ***User Interface (UI) Layer:***

**Description:** This layer serves as the user-facing part of the application, responsible for presenting the interface to end-users and handling their interactions. It encompasses React code.

**Components:** React code.

**Functionality:** The User Interface (UI) Layer provides the graphical user interface, allowing users to interact with the system and submit search queries.

* ***Application Layer:***

**Description:** The Application Layer acts as the central hub of the system, where it manages user queries, and communication with the backend services. It is powered by the Flask backend code.

**Components:** Flask backend code.

**Functionality:** This layer takes care of processing user requests, and establishing connections with the backend services. It plays a pivotal role in the overall functioning of the application.

* ***Data Storage Layer:***

**Description**: The Data Storage Layer is responsible for the storage and organization of data, in ChromaDB. These components are hosted on separate instances or databases, serving as repositories for data retrieval.

**Components:** ChromaDB.

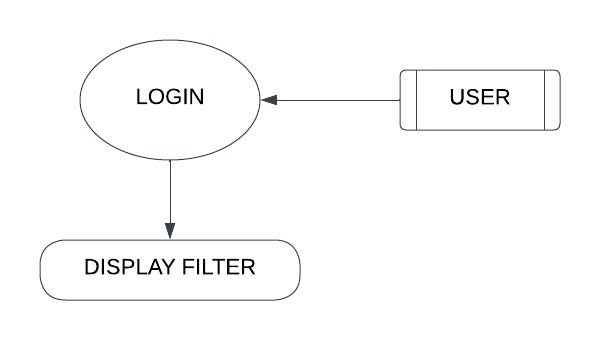
**Functionality:** This layer manages the storage and retrieval of data required to respond to user queries. It ensures efficient data access and management, enhancing the application performance.

**3.2 STRUCTURAL DESIGN**

Structured design is a conceptualization of a problem into several well-organized elements of solution. Structured design is mostly based on 'divide and conquer' strategy where a problem is broken into several small problems and each small problem is individually solved until the whole problem is solved.

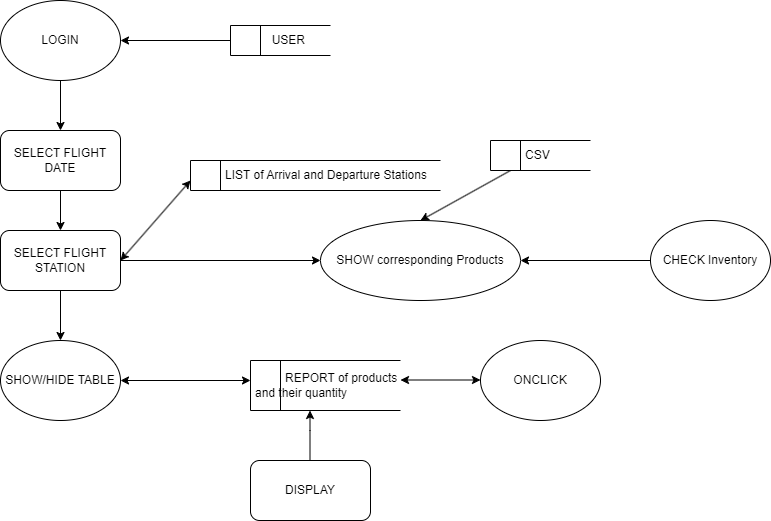
***3.2.1 DATA FLOW DIAGRAM***

Also known as DFD, Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.



**Figure 3.3 Authentication**

Figure 3.3 summarizes the login module where the authentication is done in COPS portal



**Figure 3.4 Layers and tables**

Figure 3.4 describes how table data gets displayed through data retrieval in COPS portal

***3.2.2 MODULE DIAGRAM***

Module diagram is a design approach that subdivides a system into smaller parts called modules that can be independently created and then used in different systems.

The search system is divided into the following modules as shown in Figure 3.4, 3.5

***Extraction Phase:***

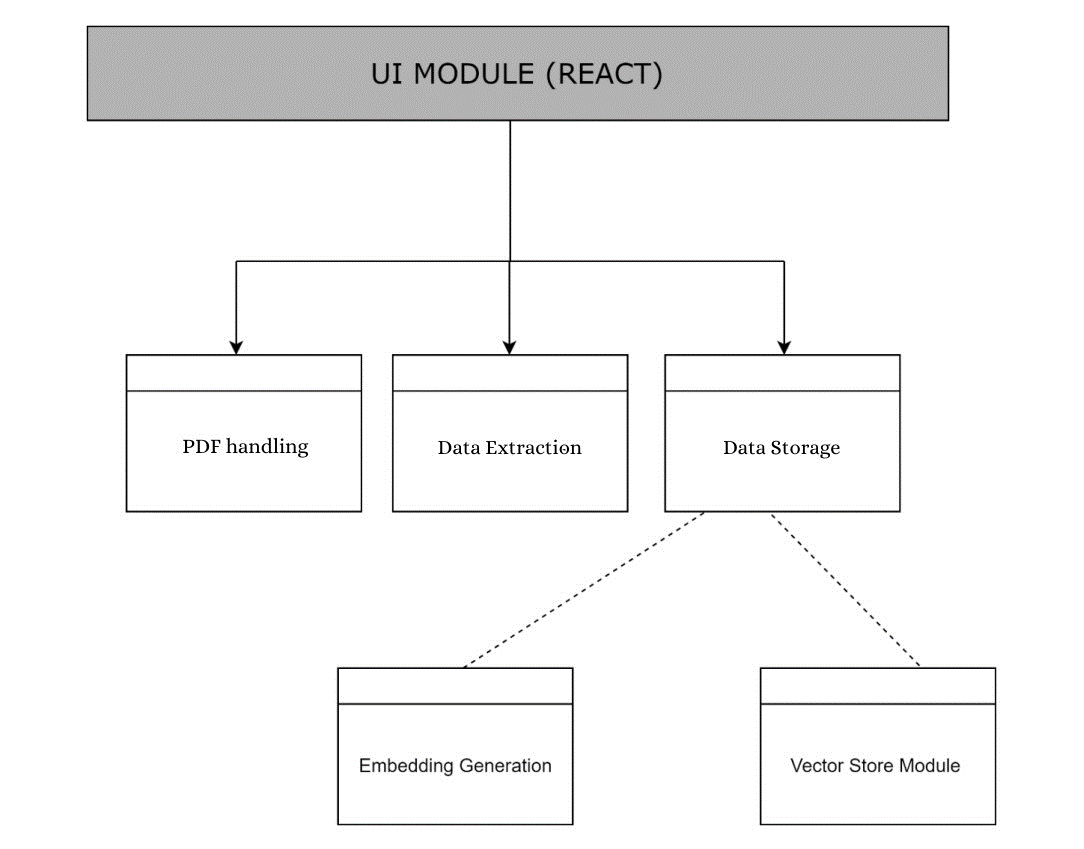
**PDF Handling Module:** This module is responsible for managing and processing the PDFs provided by users. It may validate and prepare the PDFs for the data extraction.

**Data Extractor Module:** The data extractor module is in charge of extracting content from the PDFs.

**Data Storage Module:** After data is extracted, this module handles the storage of the collected information. It could store the data as text files or within a database for further processing.

**Embedding Generation Module:** The embedding generation module utilizes a pre-trained embedding model, often provided by OpenAI, to convert the scraped text data into numerical embeddings.

**Vector Store Module:** The vector store module (e.g., ChromaDB) is responsible for managing and efficiently storing these embeddings. It plays a crucial role in enabling fast and effective similarity searches by allowing retrieval of relevant embeddings during the search process.



**Figure 3.5 – Module Diagram (Extraction phase)**

***Searching Part:***

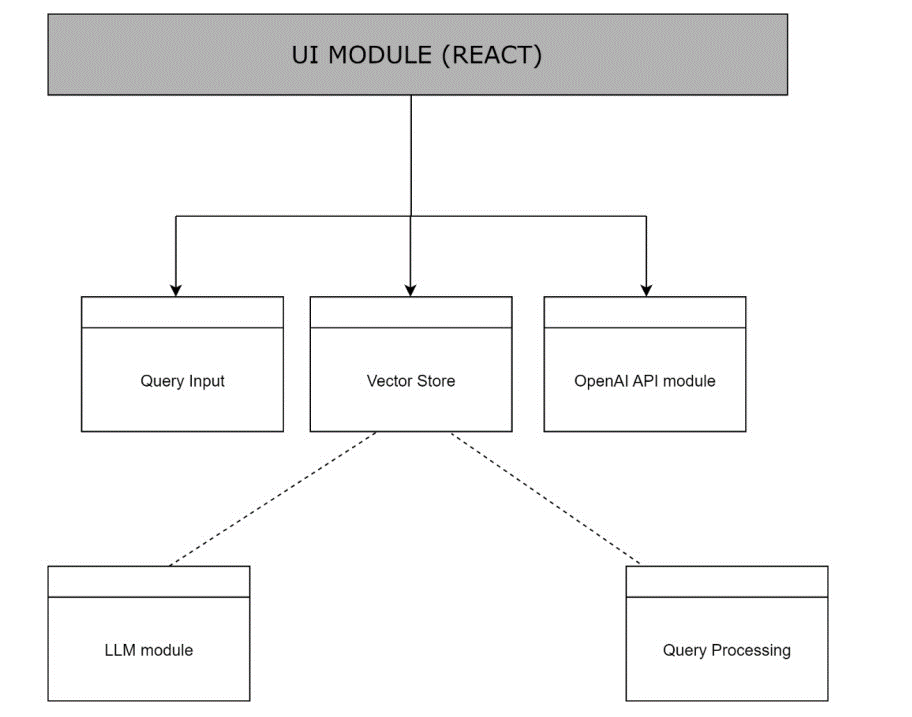
**Query Input Module:** The query input module handles user queries, whether from a web interface, chatbot, or API. It may also include components for query validation and processing.

**Vector Store Module:** This module is shared between the scraping and searching parts of the application and responsible for managing the storage and retrieval of embeddings generated from the scraped data.

**OpenAI API Module:** This module connects to external APIs provided by OpenAI, facilitating communication with large language models like GPT-3.5. It sends queries to the language model and receives responses.

**LLM Module (Large Language Model):** The LLM module connects to a large language model, such as GPT-3.5, and can be used for generating conversational responses based on user queries and retrieved results. It may also be fine-tuned for your application.

**Query Processing Module:** The query processing module plays a central role as it interfaces with the vector store to retrieve relevant embeddings for similarity searches.



**Figure 3.6 – Module Diagram (Searching part)**

The two figures 3.5 and 3.6 represents all the modules involved in the application in Extraction phase and Searching phase respectively.

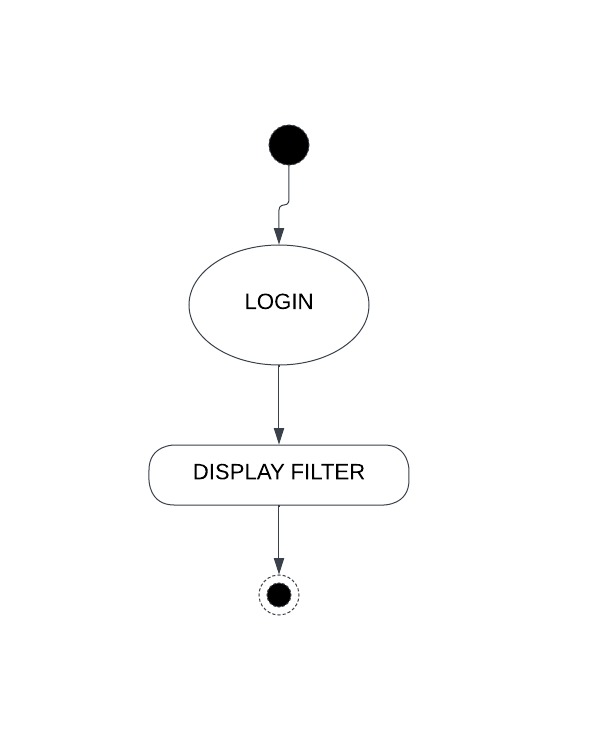
**3.3 BEHAVIOURAL DESIGN**

In software engineering, behavioral design patterns are design patterns that identify common communication patterns among objects. By doing so, these patterns increase flexibility in carrying out communication.

***3.3.1 ACTIVITY DIAGRAM***

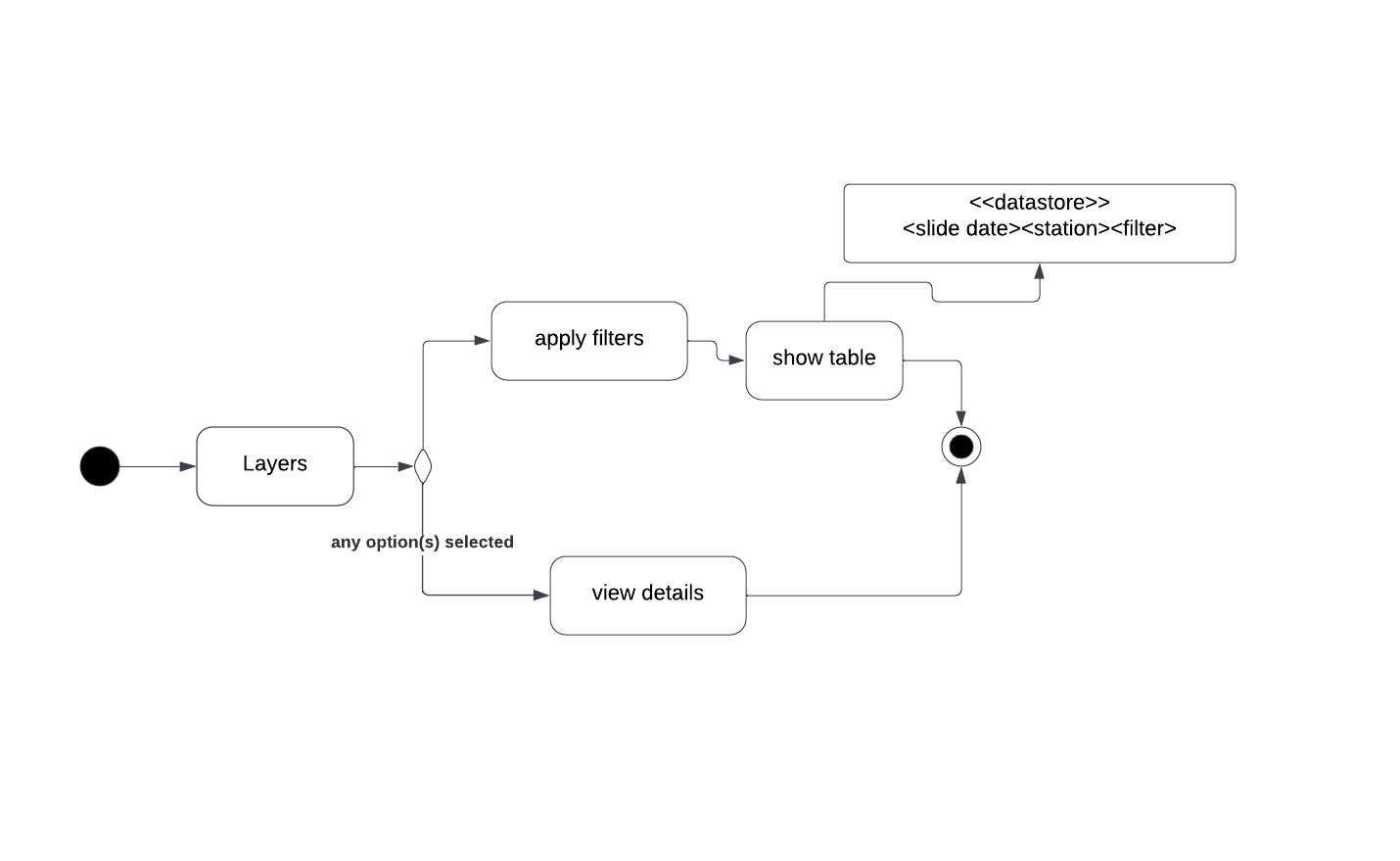
An activity diagram is a behavioral diagram which depicts the dynamic aspects of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

**3.3.1.1 TASK - 1**



**Figure 3.7 Authentication**

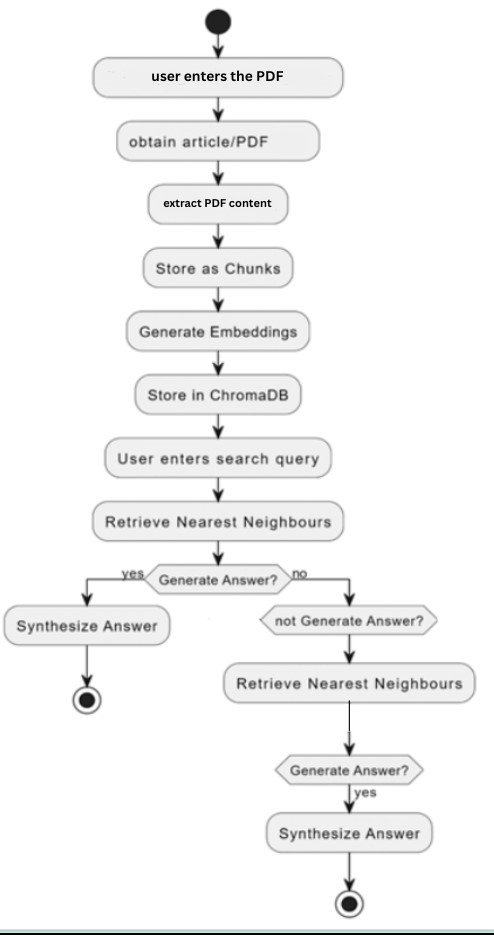
Figure 3.7 provides the authentication module’s outlook.



**Figure 3.8 Layers and table**

Figure 3.8 shows the activities of layers & tables in COPS application.

**3.3.1.2 TASK – 2**

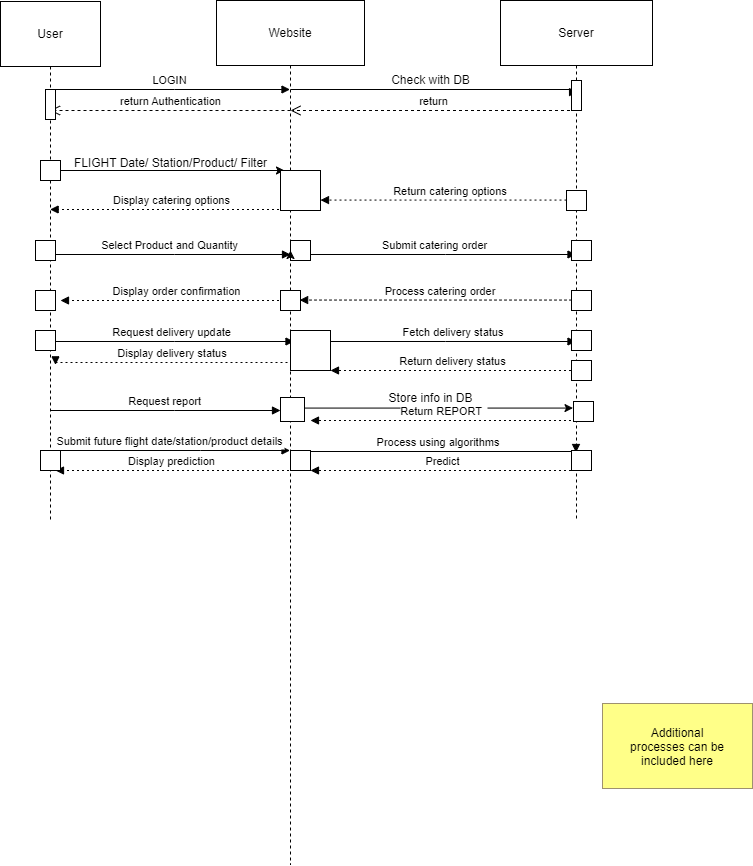


**Figure 3.9 Activity diagram for search application**

The figure 3.9 depicts the overall activity of the search application where the activity starts when the user enters the pdf and ends when the answer is synthesized for user’s query.

***3.3.2 SEQUENCE DIAGRAM***

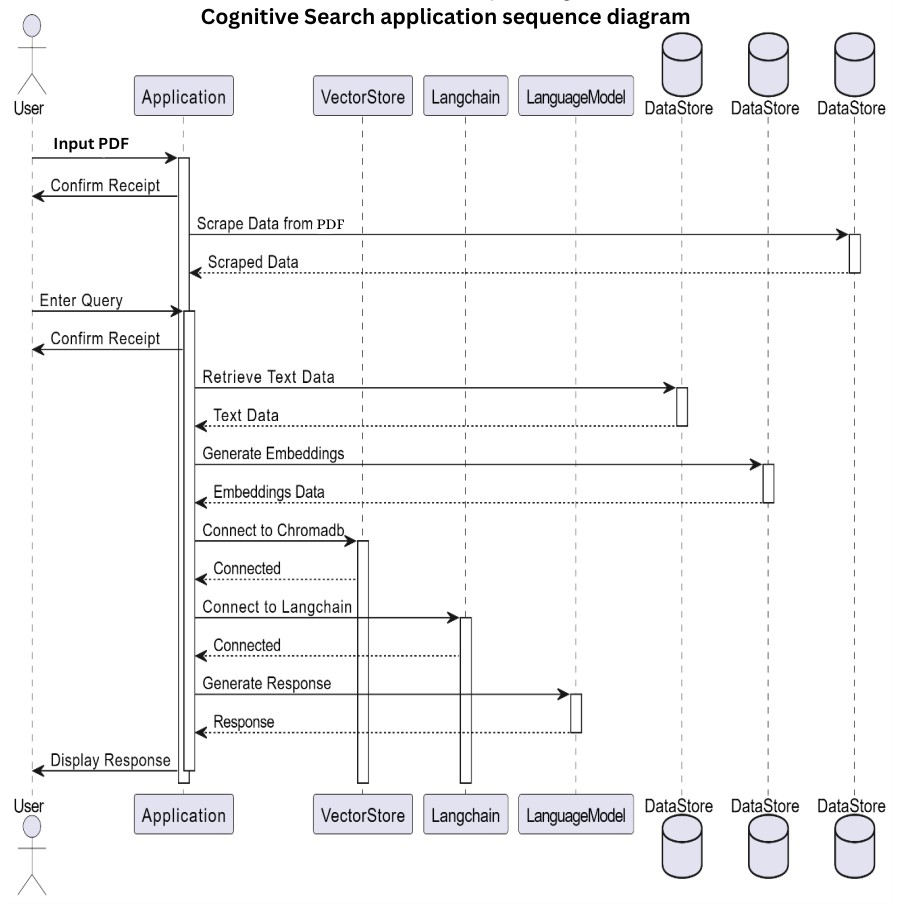
UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. Sequence Diagrams are time focused and they show the order of the interaction visually by using the vertical axis of the diagram to represent time, what messages are sent and when. A sequence diagram is a visual representation used in software engineering to depict the interactions and order of messages between different components or objects within a system over a specific period of time. Typically, sequence diagrams consist of lifelines representing participants, arrows indicating the messages exchanged between the participants, and a timeline that shows the chronological progression of events.

**3.3.2.1 TASK - 1**

**Figure 3.10 COPS application sequence diagram**

Figure 3.10 depicts the flow of interactions & data processing in COPS application

**3.3.2.2 TASK – 2**



**Figure 3.11 – Cognitive Search application sequence diagram**

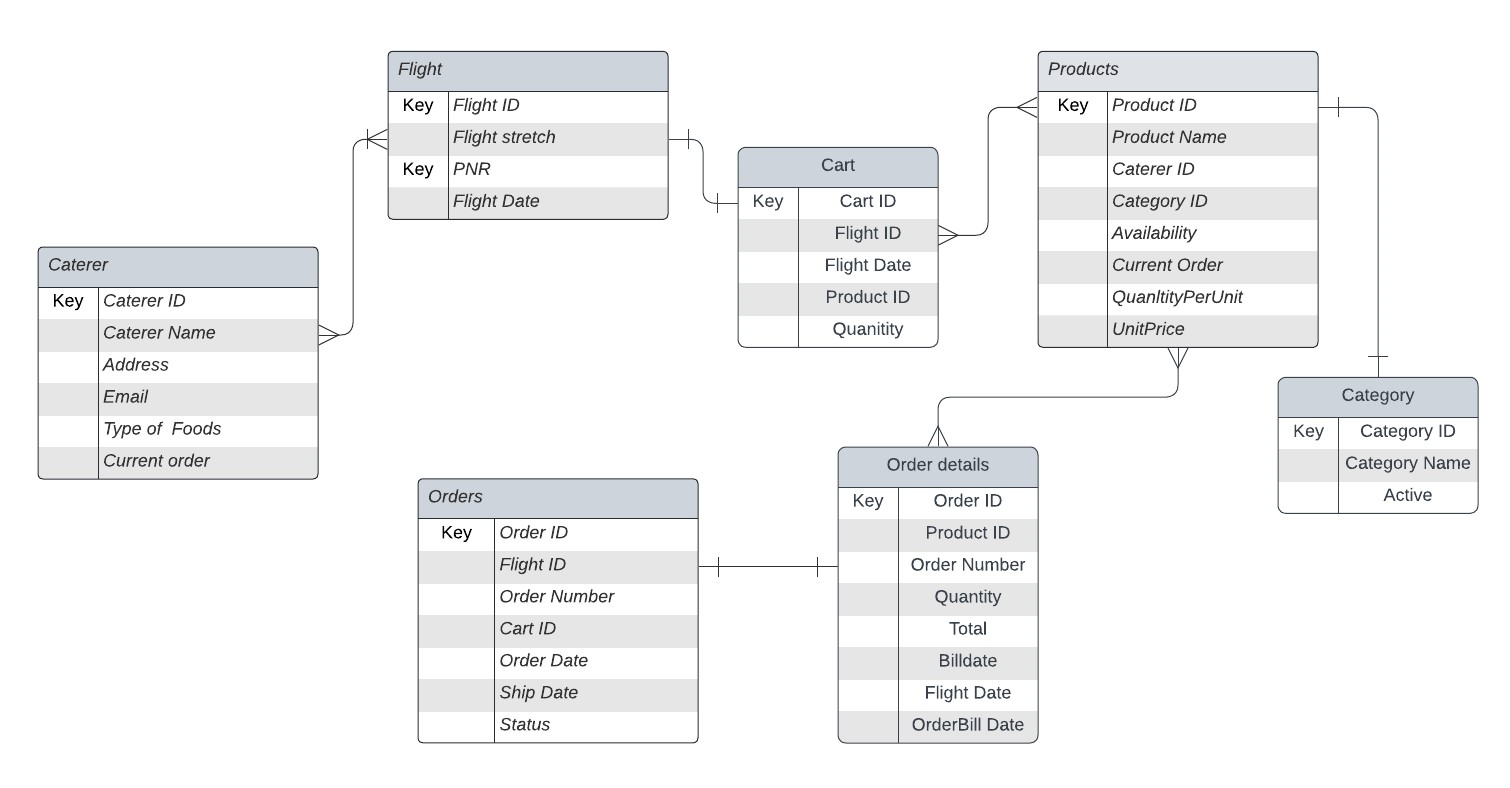
The figure 3.11 depicts user interaction, PDF text extraction, query processing, data analysis, and response generation. This sequence diagram illustrates the flow of interactions and data processing in the Cognitive Search application.

**3.4 TABLE DESIGN**

Table Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed databases are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.

**3.4.1 TASK – 1**

MongoDB is an open source NOSQL database management program. MongoDB is a tool that can manage document-oriented information, store or retrieve information. MongoDB is used for high-volume data storage, organizations use MongoDB for its ad-hoc queries, indexing, load balancing, aggregation and server-side JavaScript execution.

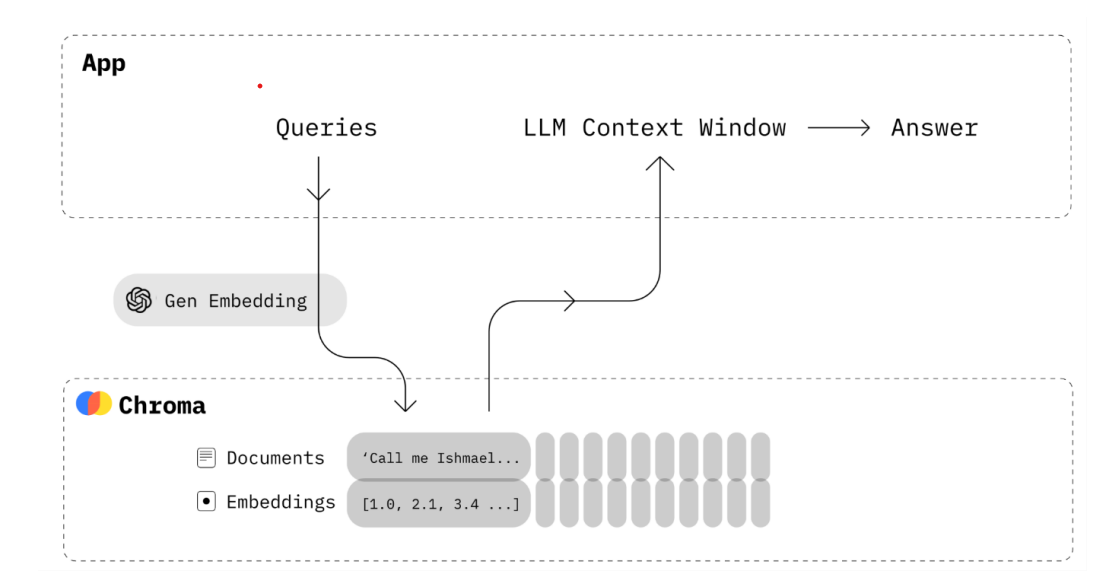


**Figure 3.12 – COPS application database design**

Figure 3.12 depicts the database design of COPS application which comprises of 7 collections each defined by sample entries

**3.4.2 TASK – 2**

Chroma DB is an open-source vector store used for storing and retrieving vector embeddings. Its main use is to save embeddings along with metadata to be used later by large language models. ChromaDB is structured as a database of collections. Each collection contains a set of embeddings and their associated metadata. Embeddings are vectors that represent the meaning of text documents. Metadata can be any type of information about the documents, such as the author, the date, or the topic.



**Figure 3.13 – Structure of ChromaDB**

The contents in the database are associated with the following two attributes:

• page\_content – it stores all the data scraped from data sources in vector store format.

• metadata – the directory of the source file found in the page\_content will be stored here.

The Table 3.1 represents the storage structure of chromadb with the sample data.

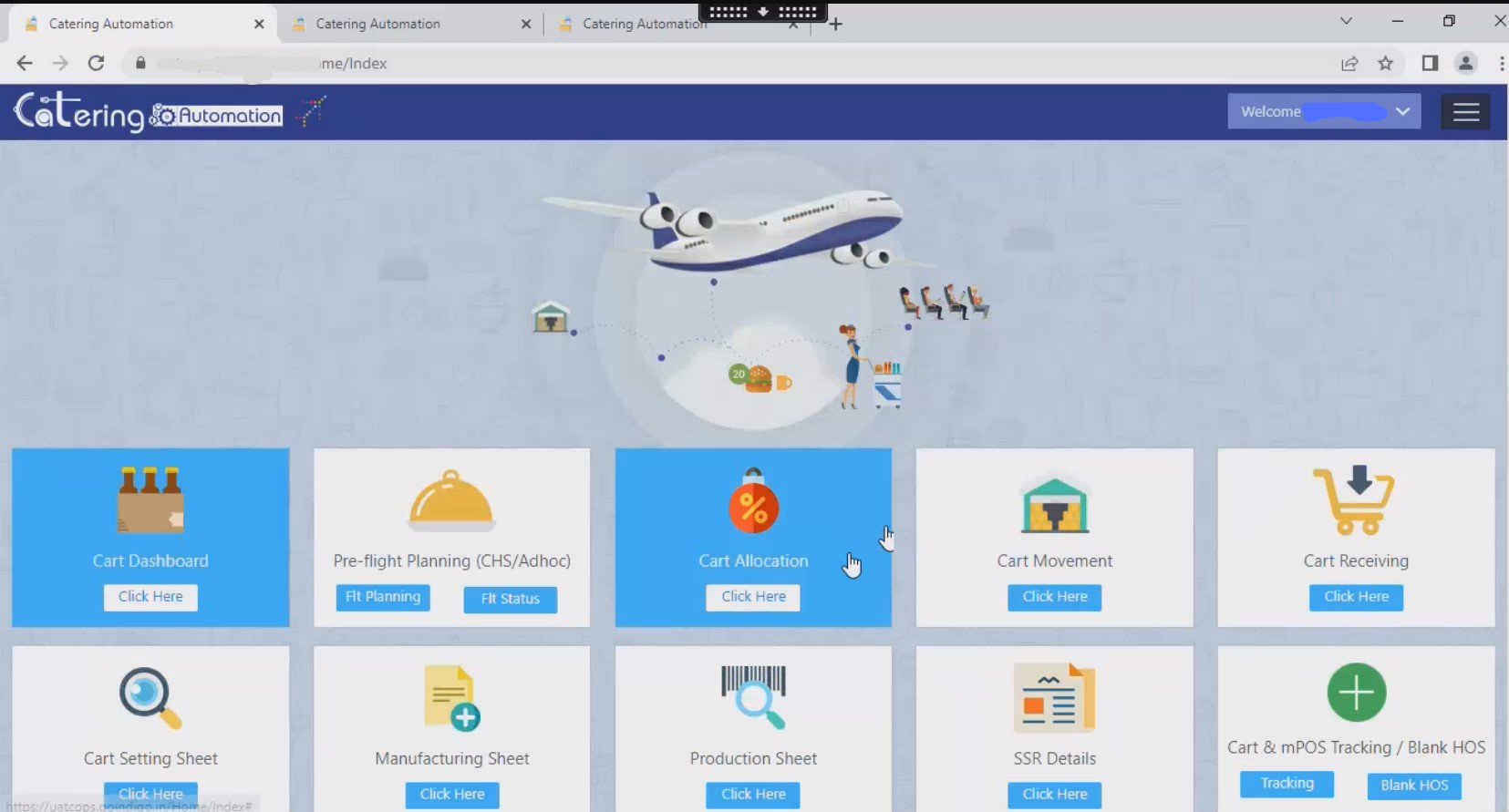
**Table 3.1 – Table design (ChromaDB)**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Description** | **Sample data** |
| **page\_content** | Contains the extracted content | PDF 1 URL: <https://www.aircanada.com/content/dam/aircanada/portal/documents/PDF/en/Code_of_Conduct.pdf> The Corporate Policy and Guidelines on Business Conduct (the “Code”) applies to all directors, officers and employees of Air Canada (the “Company”) |
| **metadata** | Specifies the directory of the Source file | {'source': './pdfs/codeofconduct.txt'} |

**3.5 USER INTERFACE DESIGN**

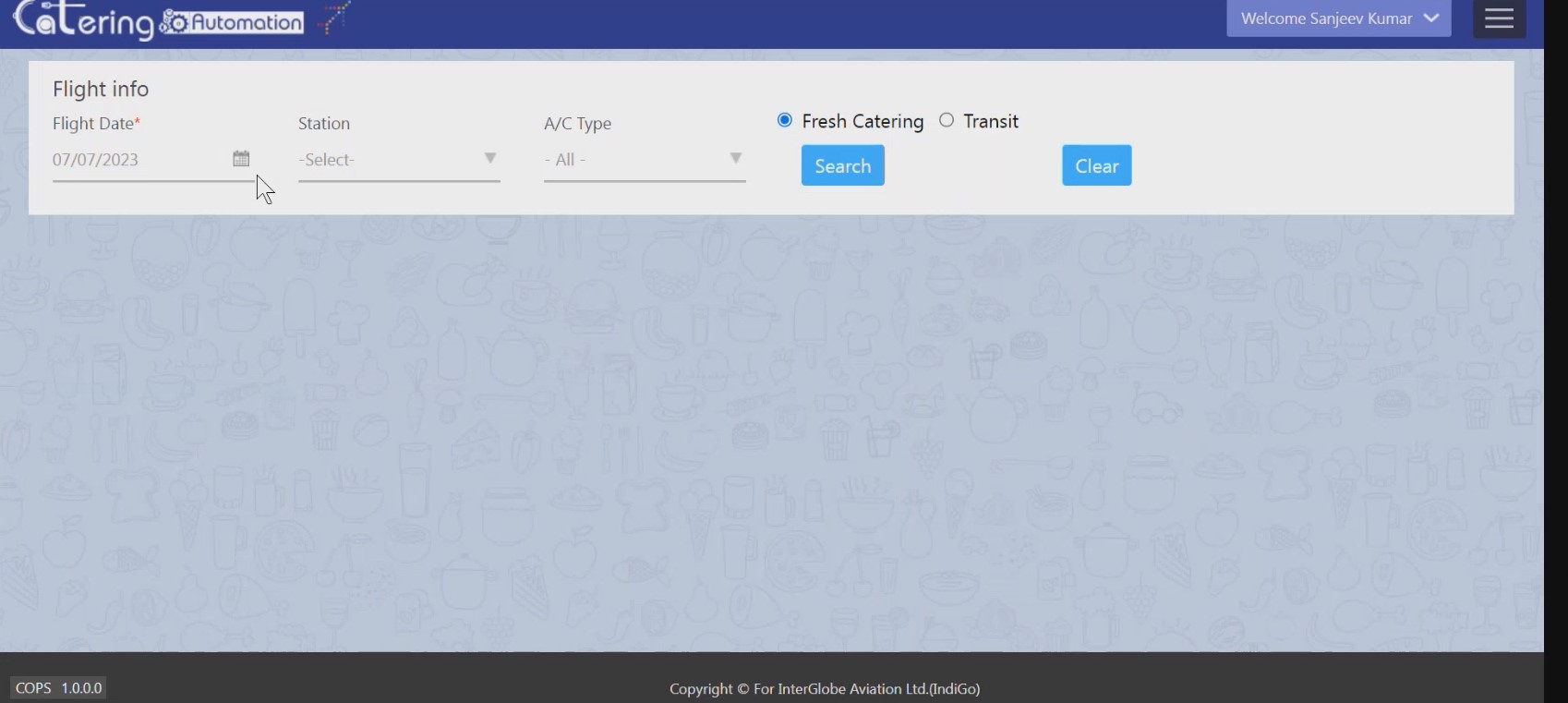
User Interface Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

**3.5.1 TASK - 1**



**Screen 3.1 Main screen**

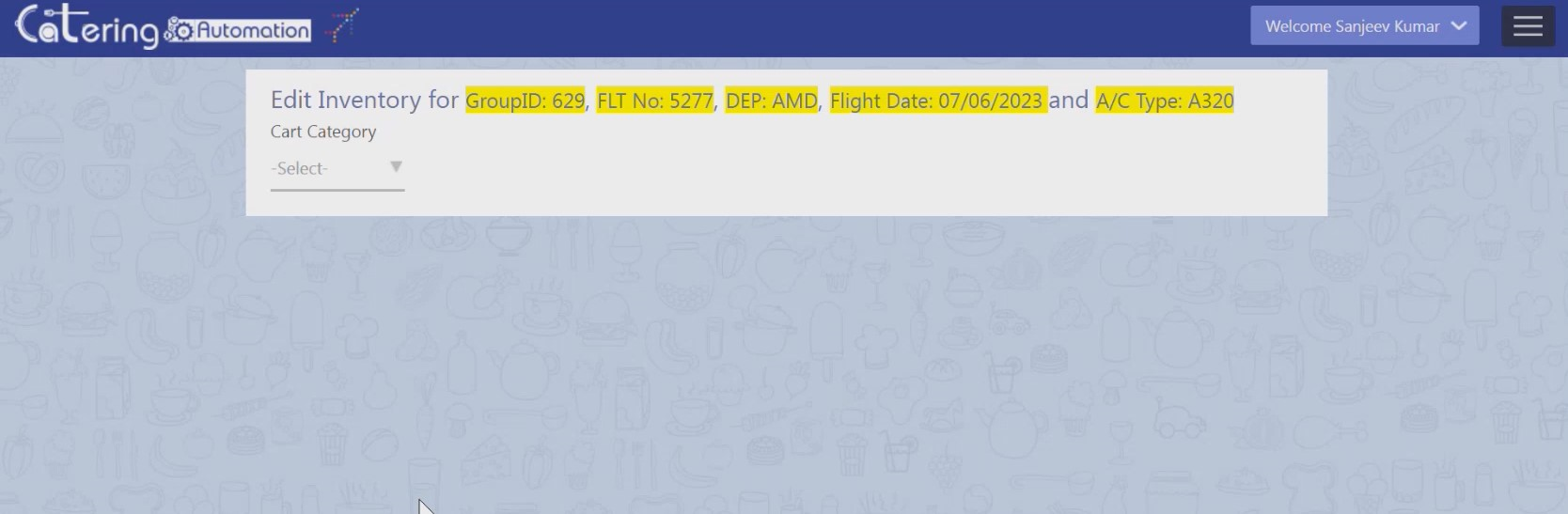
Screen 3.1 displays the main screen of the application which appears after authentication.



**Screen 3.2 After successful authentication of the user**

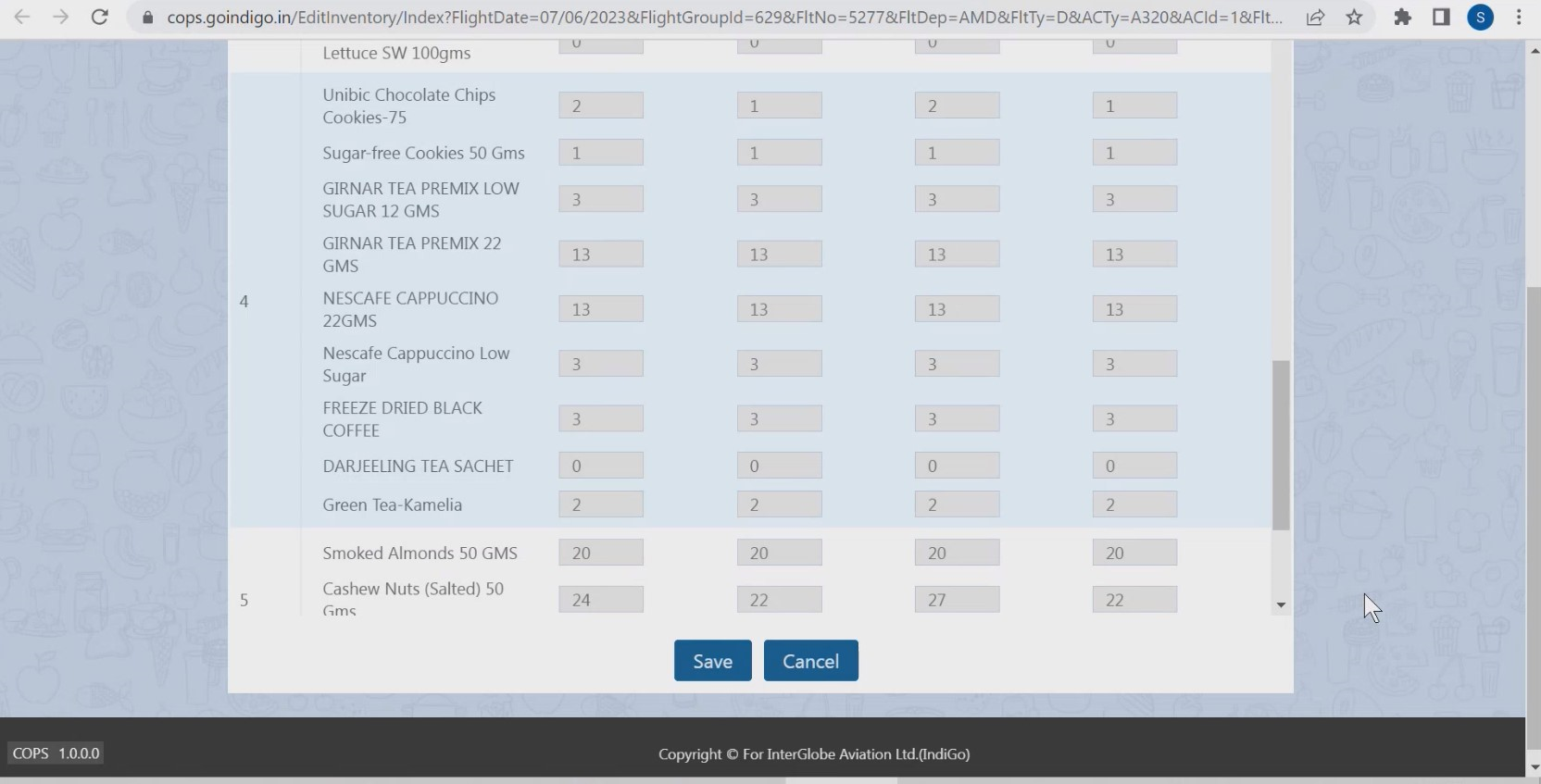
Screen 3.2 depicts options that can be selected

Screen 3.3 edit inventory for specific flight.



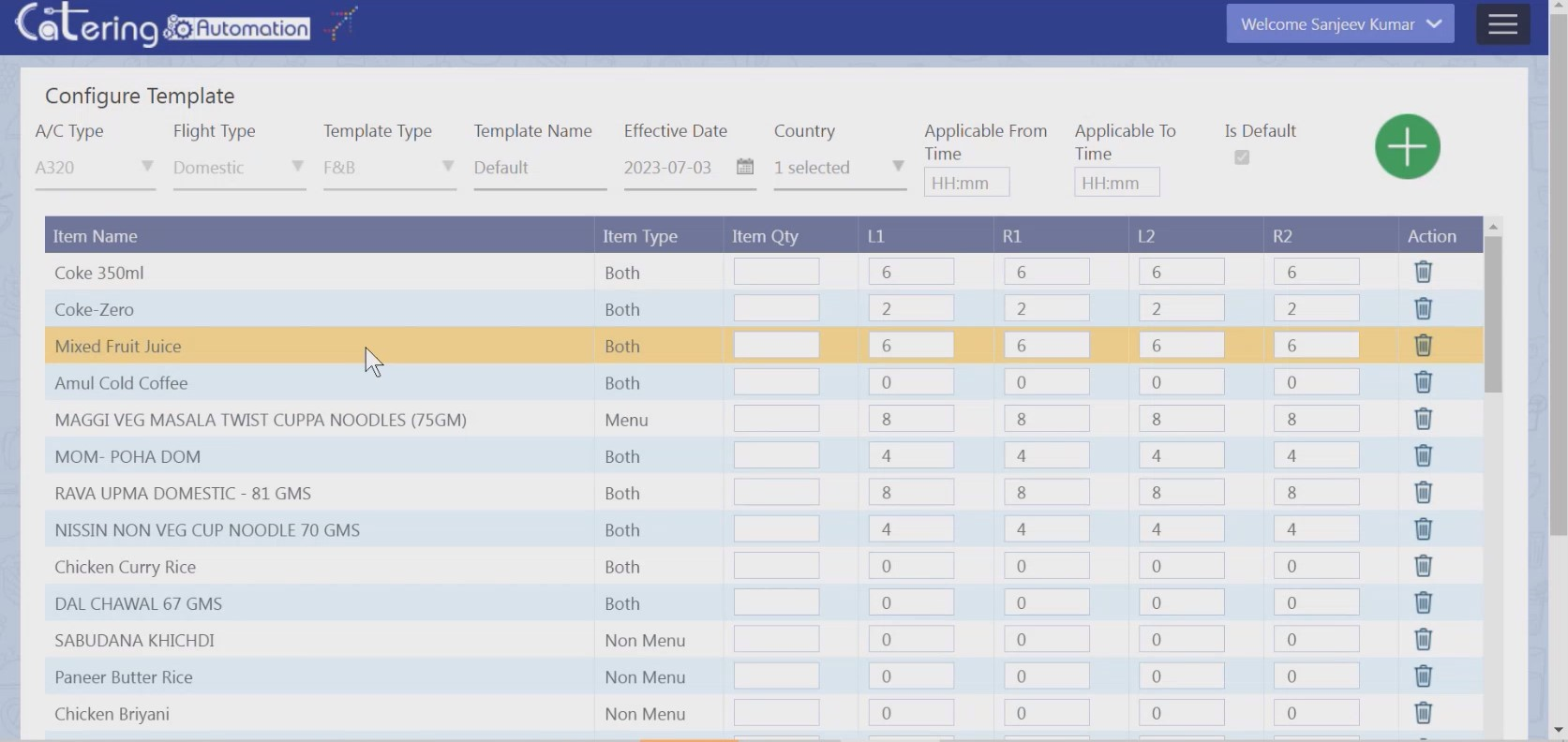
**Screen 3.3 Edit inventory**





**Screen 3.4 Edit inventory after selecting “F&B”**

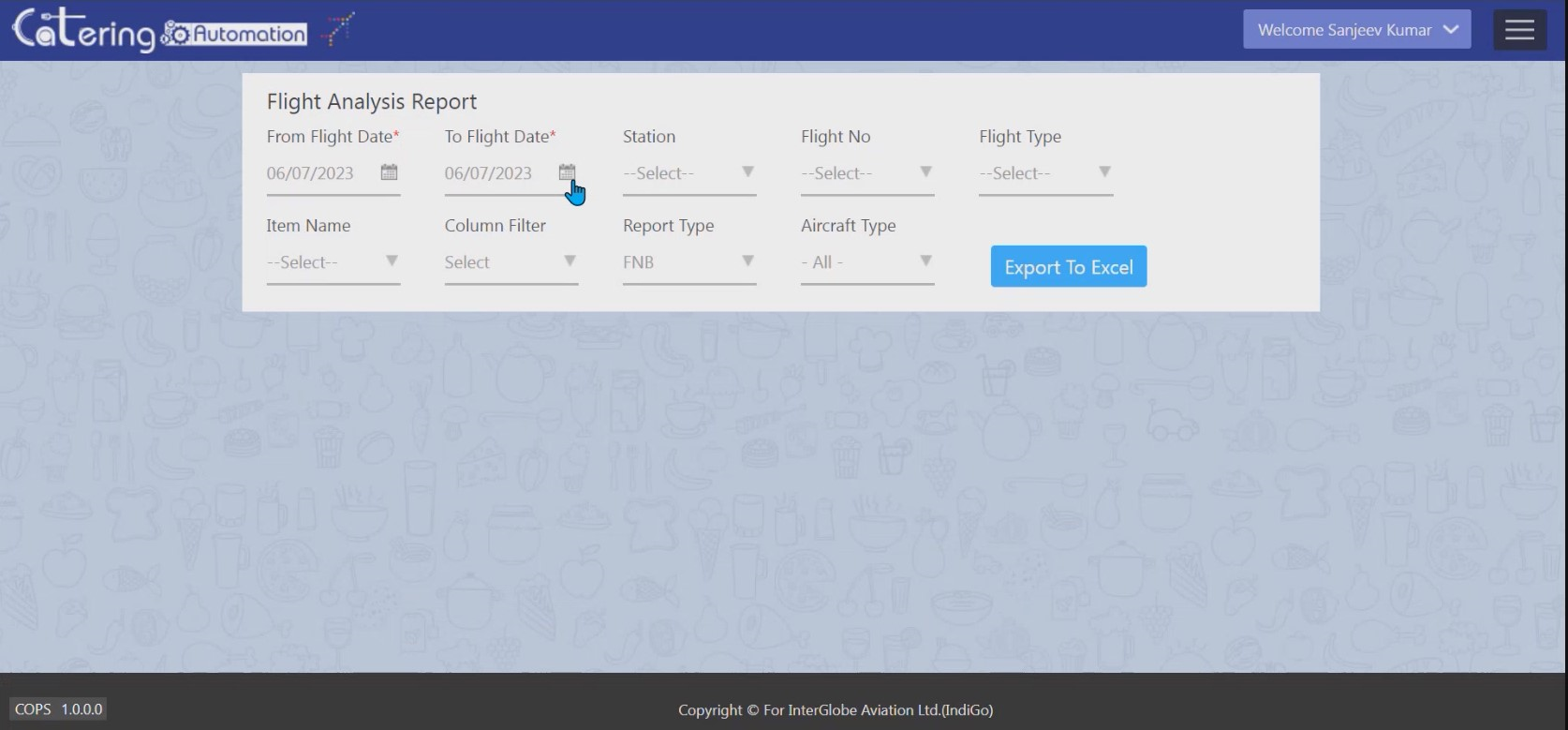
Screen 3.4 depicts inventory items that can be added to the cart



**Screen 3.5 ConfigureTemplate**

Screen 3.5 shows the inventory items which can be edited according to stock levels.

Screen 3.6 depicts the report generator module.



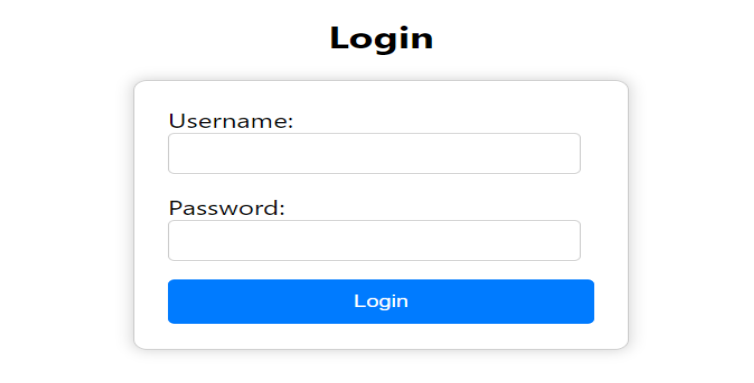
**Screen 3.6 Report generator module**

**3.5.2 TASK - 2**

The Cognitive Search Application requires minimal and necessary features in its User Interface design, login page is necessarily followed by these two sections separately,

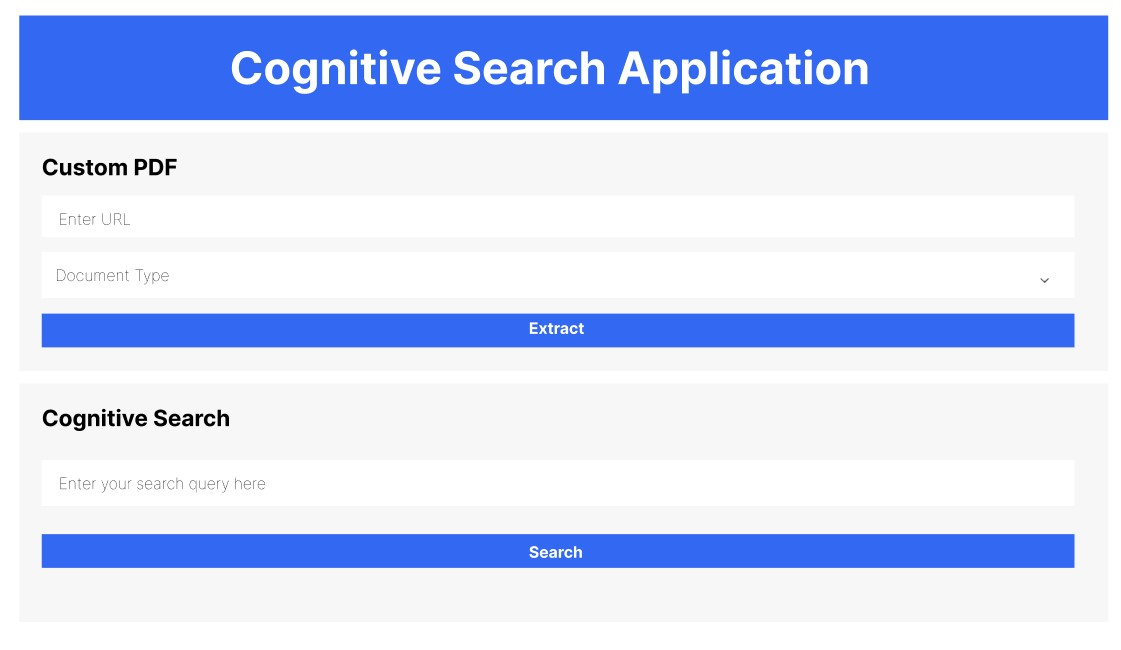
• ***Custom Scraping Section*** – It handles the PDF input from the user and verifies it using some testing techniques and then ensures the completion of scraping process.

• ***Conversational Search Section*** – It has a comparatively large text box which can get large text or queries as input and then displays the answers.



**Screen 3.7 Authentication module**

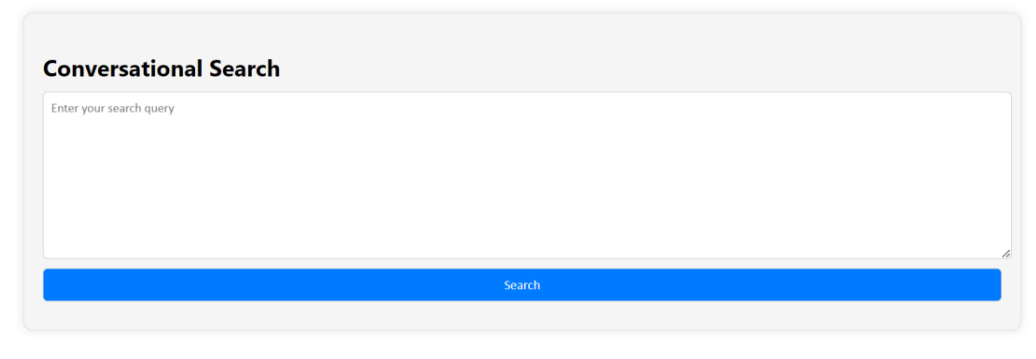
Screen 3.7 depicts the authentication module



**Screen 3.8 Search Application**

Screen 3.8 depicts the search application interface

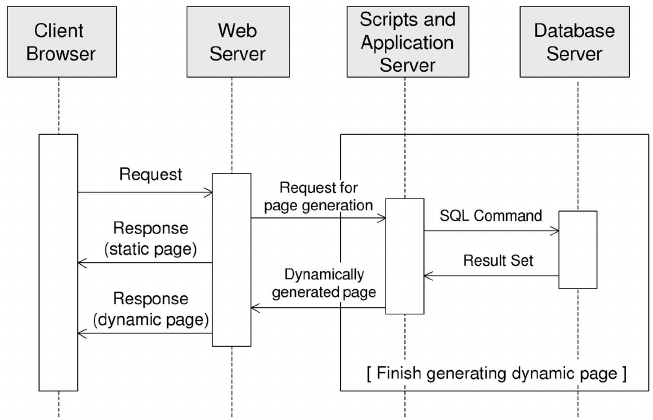
Screen 3.9 depicts the Conversational Search part of the Application

****

**Screen 3.9** **Conversational Search part of the Application**

**3.6 DEPLOYMENT DESIGN**

A deployment diagram falls under the structural diagramming family because it describes an aspect of the system itself. In this case, the deployment diagram describes the physical deployment of information generated by the software program on hardware components. a deployment diagram falls under the structural diagramming family because it describes an aspect of the system itself. In this case, the deployment diagram describes the physical deployment of information generated by the software program on hardware components.

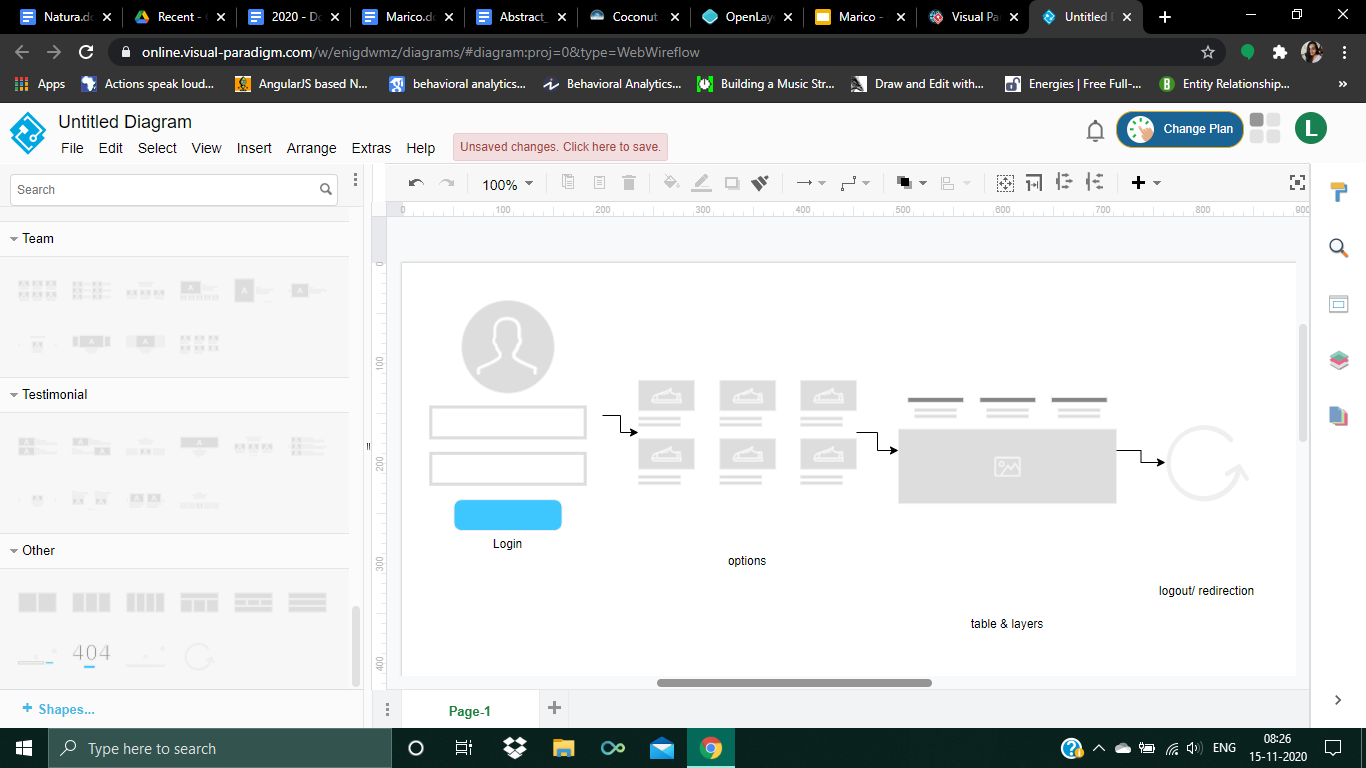
****

**Figure 3.17 Deployment Diagram for both applications**

Figure 3.17 shows the deployment of the application on the web with the backend.

**3.7 NAVIGATION DESIGN**

Navigation design is a design process that utilizes hyperlinks to organize information on a website so that site visitors can navigate, or find, the information they are looking for. Navigation can be text-based or image-based.

****

**Figure 3.18 Navigation design**

Figure 3.18 provides an overview of how one activity of a functionality links to another functionality in the application.

**3.8 CODE DESIGN**

**3.8.1 TASK - 1**

* After successful login by the user, cart module is displayed.
* Options such as Flight date, station, aircraft type and Filter are given, where date and station and filter are static and the rest are dynamic.
* On choosing only the static options, the specific details are displayed and the inventory can be edited for each specification while if any one of the rest of the options are chosen, then the corresponding layer is applied.
* In the report module, on selecting a particular item name gives the information about the consumption trend of that particular item.
* Show/hide button - displays a horizontally sliding table with the information about the layer and its quantitative data.

**3.8.2 TASK – 2**

* + - * Set the OpenAI API key.
      * Load PDF and the text data from "article.txt" using TextLoader. Open the file "article.txt" and read the content of the file.
      * Split the text data into chunks of 3000 characters with a 0-character overlap. Create an empty list to store text chunks. Iterate through the text data, Extract a chunk of 3000 characters and append the chunk to the list of text chunks.
      * Create or load the Chroma vector store from documents. Load the existing database, If a Chroma database already exists. If not: create a new Chroma database from the list of text chunks. Persist the database. Create a retriever from the Chroma vector store - Define search parameters, such as 'k' (number of nearest neighbors).
      * Initialize a RetrievalQA chain with an OpenAI Language Model (LLM). - Configure the chain with the LLM and retriever.
* Define a function to process LLM responses. Accept the LLM response as input. Extract and print the response text. Iterate through the source documents. Extract and print the source metadata (e.g., source of the document).

Loop:

* User provides a query. Confirm the receipt of the user's query. Execute the RetrievalQA chain to generate a response. Process the LLM response using the defined function.
* Retrieves the nearest neighbour from the embeddings. Call the function with the LLM response as input. Display the response to the user.
* - Print the LLM-generated response.

**CHAPTER IV**

**SYSTEM TESTING**

System testing is a generic metric of performing a variety of tests on a system to explore functionality or to identify problems. A series of systematic procedures are referred to test how the system should perform and where common mistakes may be found by entering data that may cause the system to malfunction or return incorrect information. The purpose of testing is quality assurance, verification and validation, or reliability estimation. A Test Plan documents the strategy that will be used to verify and ensure that a product meets its design specifications. The three major elements that constitute the test plan are Test Coverage, Test Methods, and Test Responsibilities.

**4.1 TEST CASES AND TEST REPORTS**

Test cases are built around the requirements and specifications i.e., what the system is supposed to do. Test cases are generally driven from the external descriptions of the framework and design parameters.

Each test case contains item criteria such as:

1. PASS

All expected results are achieved and/or all unexpected events are resolved.

1. PASS WITH EXCEPTIONS

Unexpected events require alternative procedures that have been implemented and those events are called Exceptions.

1. FAIL

The testing process response does not confirm the expected results.

A test case report provides information about the status of test cases, test suites, or test scripts for a given scope. A test case is a document that has a set of test data, preconditions, expected results, and postconditions. It's developed for a particular test scenario in order to verify compliance against a specific requirement.

Table 4.1 describes the various test cases and reports for a few iterations.

**Table 4.1 Test cases and reports for COPS application**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TEST CASE ID** | **TEST CASE** | **TEST STEPS** | **EXPECTED RESULT** | **ACTUAL RESULT** | **STATUS** |
| T\_01 | Website links | Check if able to Navigate through all modules.  Modules integrated and in-sync | Navigation Successful | Navigation Successful | PASS |
| T\_02 | Search functionality | Verify that on searching, all the products satisfying the search criteria are visible on the search result page.  Validate the search functionality by testing different search queries and ensuring relevant products are displayed. | Search criteria satisfied | Search criteria satisfied | PASS |
| T\_03 | Filter working | Verify that filtering functionality correctly filters products based on the filter applied.  Verify if it works correctly on category pages and search result page. | Filtered items displayed | Filtered items displayed | PASS |
| T\_04 | Cart working | Check new items updated in cart.  Check deleted items removed from cart.  Check total number of items updated properly | Updations Successful | Updations Successful | PASS |
| T\_05 | Report generator | Verify report generated based on filtered options.  Verify report downloadable OnClick | Report generation Successful | Report generation Successful | PASS |

**4.2 COMMON TEST CASE SCENARIOS**

* Test with different browsers
* Test with different user access privileges
* For report generation, test for the correctness of the data

Functionalities such as login, extraction and conversational search functionality with the expected and actual results obtained. The Table 4.2, contains the details of the different test cases that are applicable for this search application.

**Table 4.2 Test cases and reports for search application**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASE ID** | **TEST CASE** | **TEST STEPS** | **INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **STATUS** |
| T01 | Verify login functionality with valid username and password | 1. Navigate to the login page  2. Enter username, password  3. Click login | 1. username  2. password | Home page of the search application | Home page visible after logging in | PASS |
| T02 | Verify login functionality with invalid username and correct password | 1. Navigate to the login page  2. Enter username, password  3. Click login | 1. username  2. password | Error message showing "Invalid username" | Alert displaying "Invalid username" | FAIL |
| T03 | Verify login functionality with valid username and wrong password | 1. Navigate to the login page  2. Enter username, password  3. Click login | 1. username  2. password | Error message showing "Invalid password" | Alert displaying "Invalid password" | FAIL |
| T04 | To verify the correctness of uploaded PDF in unsupported document type | 1. Login to the application  2. Upload the PDF  3. Specify the type of source  4. Click Submit | 1. username  2. password  3. PDF | Error displaying "Unsupported document " | Alert showing "PDF & type mismatch" | FAIL |
| T05 | To verify the correctness of uploaded PDF in correct document type | 1. Login to the application  2. Upload the PDF  3. Specify the type of source  4. Click Submit | 1. username  2. password  3. PDF | Message stating "Scraping successful" should be displayed | "Scraping successful" message displayed | PASS |
| T06 | Proper input query and provide relevant answer | 1. Login to the application  2. Enter the search query  3. Click Submit | 1. username 2. password  3. query | After searching, the synthesized answer should be displayed | Relevant answer displayed | PASS |
| T07 | Improper input query and provide uncertain answer | 1. Login to the application  2. Enter the search query  3. Click Submit | 1. username 2. password  3. query | Should display an error "I don't know" | Error message "I don't know" displayed | PASS |
| T08 | Check for the proper input query and provide irrelevant answer | 1. Login to the application  2. Enter the search query  3. Click Submit | 1. username 2. password  3. query | Should display relevant answer | Displayed irrelevant answer | FAIL |

**CHAPTER V**

**SYSTEM IMPLEMENTATION**

Implementation is the process of converting a new or a revised system design into an operational one. It is the most crucial stage in achieving a new successful system and in giving confidence to the new system for the teams that it will work efficiently and effectively. In this phase, one can build the components either from scratch or by composition. Given the architecture document from the design phase and requirement document from the analysis phase, one can build exactly what has been requested. The type of platform in which the proposed system is to be implemented and it should be accessible for the user is described. The type of platform in which the proposed system is to be implemented and it should be accessible for the user is described.

This application can be directly downloaded from the company’s private server onto any PC that meets the minimum configuration. Also, does not require any pre-setting of the database as the application is well-integrated.

**5.1 INSTALLATION PROCEDURE**

***5.1.1 TASK – 1 (COPS Application)***

The following steps must be followed to complete the installation:

1. Installing Node.js

* 1. Download Node.js installer from “https://nodejs.org/en/download/”
  2. Run the downloaded installer and follow the wizard to complete.

2. Installing React

Run the following command on Node.js prompt: “npm -g create-react-app”

3. Installing MongoDB

Download MongoDB Compass from https://www.mongodb.com/try/download/shell

Run the downloaded installer and follow the wizard to complete.

***5.1.2 TASK – 2: (Search Application)***

• Determine the right Integrated Development Environment and the Code Editor

for the execution of the code.

• Create a virtual environment using ‘python -m venv myenv’

• Install the packages using ‘pip install -r requirements.txt’

• Navigate to the flask directory in the project

• Activate the virtual environment by ‘venv\Scripts\activate’

• Run the development server using,

• python manage.py runserver

• Make necessary migrations whenever the changes are made to models

• Navigate to the ‘frontend’ directory

• Then install the necessary packages for react using ‘npm install’

• Start the development server using the following command:

‘npm start’

• Run the application.

***5.1.2.1 ENVIRONMENT:***

python venv (Virtual Environment) is the environment to be set by following above steps.

***5.1.2.2 PACKAGES:***

• chroma-vectorstore==0.4.0

• langchain==0.6.0

• openai==0.27.6

• Flask==3.2.9

• react

• react-router-dom==6.2.1

• react-redux==7.2.5

• react-hook-form==7.16.2

**CHAPTER VI**

**CONCLUSION**

Concluded with the salient features of the proposed systems and its characteristics, we define its future scope and the final implemented system.

**6.1 CONCLUSION**

***6.1.1 TASK - 1***

The COPS portal is a powerful solution that addresses the complexities and challenges of managing catering services for airlines. The integration with flight route is the customized feature that enables the portal to fulfill its requirements.

***6.1.2 TASK – 2***

In a fast-growing world, data is present everywhere. There are no efficient systems to extract useful information from the raw data. The Cognitive Search Application will be a boon to the organizations who are in need of a system to efficiently chat with their data. The main advantage of the search application is that it uses Large Language Models to generate answers. This application ensures the answer generation for the given query.

**6.2 FUTURE ENHANCEMENTS**

In the future, it may be necessary to include some additional features that will be beneficial to both administrators and customers.

* COPS application can be enhanced by using AI to suggest menu options based on passenger preferences, flight schedules, and previous orders. Develop advanced reporting capabilities to provide insights through interactive dashboards.
* Cognitive Search application in future can be made available to accept all formats of documents (e.g.: docs, xlsx, ppt) and improve through user feedback.

**BIBLIOGRAPHY**

**REFERENCE BOOKS**

1. React 18 Design Patterns and Best Practices, Fourth Edition by Carlos Santana Roldán, 2023.
2. MERN Projects for Beginners: Create Five Social Web Apps Using MongoDB, Express.js, React, and Node, First Edition by Nabendu Biswas, 2021.

**WEBSITES**

1. "Documentation · MongoDB", mongodb.com[Online].

Available: <https://www.mongodb.com/docs/compass/current/>

1. "Documentation: JavaScript", developer.mozilla.org. [Online].

Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

1. "Documentation - React", react.dev. [Online].

Available: <https://react.dev/learn>.

1. “Documentation - nodejs”, nodejs.org. [Online]

Available: https://www.w3schools.com/nodejs/

1. “Documentation -Flask”

Available: <https://flask.palletsprojects.com/en/3.0.x/tutorial/>

1. “Documentation-Langchain” <https://python.langchain.com/docs/use_cases/question_answering>
2. “Documentation -ChromaDb” https://docs.trychroma.com/
3. “Documentation -OpenAI API” https://platform.openai.com/docs/guides/embedding