AI SOLVEXPERT

MUHAMMAD FURQAN

HASEEB UL HASSAN

DEPARTMENT OF COMPUTER SCIENCES
COMSATS UNIVERSITY ISLAMABAD, WAH CAMPUS
WAH CANTT – PAKISTAN

SESSION 2021-2025

AI SOLVEXPERT

Undertaken By

MUHAMMAD FURQAN
REG. NO. CIIT/FA21-BCS-003/WAH

HASEEB UL HASSAN
REG. NO. CIIT/FA21-BCS-025/WAH

Supervised By

DR MUSSARAT ABDULLAH

A DISSERTATION SUBMITTED AS A PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELORS IN COMPUTER SCIENCE / SOFTWARE ENGINEERING

DEPARTMENT OF COMPUTER SCIENCES
COMSATS UNIVESITY ISLAMABAD, WAH CAMPUS
WAH CANTT – PAKISTAN

SESSION 2021-2025

FINAL APPROVAL

Certified that the project report titled, "AI SolveXpert" Application, completed by MUHAMMAD FURQAN (CIIT/FA21-BCS-003/WAH) and HASEEB UL HASSAN (CIIT/ FA21-BCS-025/WAH) is of sufficient standard, in our judgment, to warrant its acceptance by COMSATS University Islamabad, Wah Campus for the award of Degree of Bachelor of Science in Computer Science / Software Engineering.

1. External Examiner Name _____

2. Supervisor _____

Dr Mussarat Abdullah
CUI Wah Campus

3. Head of Department _____

Dr. Sheraz Anjum
Head of Department
CUI Wah Campus

DEDICATION

The successful completion of this project is attributed to the blessings and grace of Allah Almighty, who bestowed upon us the strength, knowledge, skills, wisdom, and ability to accomplish this task. We would also like to extend our heartfelt gratitude to our parents for their unwavering support and guidance throughout our lives, and especially during the completion of this project. Without their valuable advice, this project would not have reached its successful conclusion.

We express our utmost appreciation to our teachers for their continued inspiration and support during these challenging times. Their dedication and commitment to our academic progress have been instrumental in shaping our academic and personal growth. We are also deeply grateful to our supervisor, **Dr Mussarat Abdullah**, whose invaluable guidance and mentorship have led us towards the successful completion of our target project. Her insights and expertise have played a crucial role in shaping the course of this project, and we are truly indebted to her.

Muhammad Furqan

Haseeb Ul Hassan

ACKNOWLEDGEMENT

Our deepest gratitude goes to Almighty Allah, the most Merciful and Compassionate, who provided everything necessary to make this project a reality. We are very grateful to our teacher, especially **Dr Mussarat Abdullah**, for allowing us to work under her supervision. She created a conducive and comfortable environment that facilitated constructive discussions and nurtured the creative process. Moreover, her ability to motivate and push us to strive for excellence has been a significant factor in the success of this project.

We recognize and acknowledge the significant contribution of our teacher in helping us to improvise creative ideas and enhancing our critical thinking skills. Her guidance and mentorship have been an invaluable asset throughout the course of this project. We are immensely grateful for her time, effort, and commitment to our academic success and personal development.

Muhammad Furqan

Haseeb Ul Hassan

PROJECT BRIEF

PROJECT NAME	AI SOLVE
ORGANIZATION NAME	COMSATS UNIVERSITY ISLAMABAD, WAH CAMPUS
OBJECTIVE	AI-Powered Learning.
UNDERTAKEN BY	MUHAMMAD FURQAN HASEEB UL HASSAN
SUPERVISED BY	DR MUSSARAT ABDULLAH DEPARTMENT OF COMPUTER SCIENCE COMSATS UNIVERSITY ISLAMABAD, WAH CAMPUS
STARTED ON	OCTOBER 2024
COMPLETED ON	MAY 2025
COMPUTER USED	HP CORE i7, 11th GENERATION SOURCE
LANGUAGE	DJANGO
OPERATING SYSTEM	WINDOWS
TOOLS USED	VISUAL STUDIO CODE

ABSTRACT

SolveXpert is an intelligent, web-based platform designed to assist users in solving academic problems with ease and efficiency. It caters to students, educators, and lifelong learners by combining advanced technologies such as Optical Character Recognition (OCR), AI-powered query solving using Gemini API, document embedding with ChromaDB, and text-to-speech avatar responses. With a user-friendly interface, SolveXpert allows users to upload questions directly from images or documents, and receive instant, clear explanations along with spoken responses.

One of the platform's key strengths lies in its ability to create a seamless interaction between user queries and AI-powered solutions. It maintains a persistent chat history for easy reference and supports copyable text outputs, enhancing both usability and learning retention. SolveXpert also ensures that all user interactions are efficiently stored and managed, making it a reliable academic assistant.

SolveXpert aims to transform how students and professionals approach problem-solving by integrating intelligent automation with intuitive design. It simplifies the learning process, encourages user engagement, and serves as a powerful tool in educational environments where quick, accurate, and interactive assistance is essential.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This chapter provides an overview of the project, highlighting its motivation, purpose, and scope. It begins by discussing the challenges students face in understanding complex academic content, especially in subjects like mathematics. The chapter introduces the proposed solution—an AI-powered platform that offers math problem-solving capabilities through natural language interaction, voice output, and document-based Q&A. Key features such as avatar-based responses, image input support, and persistent chat history are briefly outlined. This section also sets the stage for the chapters to follow by establishing the project's relevance, objectives, and technological foundation.

1.1 System Introduction

The system developed in this project is an AI-powered web application designed to solve math problems and provide contextual question answering from uploaded documents. It is primarily aimed at students and learners who seek accurate, step-by-step solutions in an interactive format. The platform integrates a fine-tuned language model capable of understanding and solving math-related queries, and a document-based RAG (Retrieval-Augmented Generation) module that enables users to upload academic material and ask questions directly from it.

The system enhances the learning experience through features such as avatar-based responses, real-time text-to-speech output, image-based question input, and persistent chat context. Accessible via a user-friendly web interface, it also includes secure email-based signup and login functionality, and offers a light/dark theme toggle for improved usability. This software bridges the gap between AI-driven solutions and student-friendly learning environments by making complex problem-solving more visual, conversational, and accessible.

1.2 Background of the System

Several educational platforms exist that offer automated solutions to academic problems—popular examples include Photomath, Wolfram Alpha, and Microsoft Math Solver. These

platforms allow users to input math queries or scan images of equations to receive step-by-step solutions. While effective in solving isolated math problems, these systems typically lack contextual continuity, document-based question answering, and interactive feedback features such as voice response or avatar engagement.

The proposed system builds upon these foundational ideas but introduces key enhancements that make it more dynamic and user-centric. Unlike existing solutions, this project integrates a fine-tuned model specifically trained for math questions, supports image-to-text input for printed queries, and offers a RAG-based module that allows users to upload academic documents and ask questions directly from them. Additionally, the use of avatar-based interaction, voice output, chat memory, and theme customization distinguishes the system as a more immersive and personalized learning tool.

1.3 Objectives of the System

The **AI-SolveXpert** is designed with the following key objectives in mind:

- To provide accurate, AI-generated solutions for math-related questions using a fine-tuned language model.
- To enable document-based question answering through Retrieval-Augmented Generation (RAG) and ChromaDB.
- To support image-based input for solving handwritten or printed math problems.
- To deliver answers through an interactive avatar with voice output using text-to-speech technology
- To maintain chat history and context across each user session for continuity
- To offer a user-friendly interface with both light and dark theme options
- To ensure secure signup and login using email verification via Mailer
- To enhance student learning by combining AI with visual and audio-based interact

1.4 Significance of the System

The proposed system holds significant value in the field of digital education by simplifying the way students interact with complex mathematical problems and academic content. By integrating AI-powered solutions, voice interaction, and contextual document-based Q&A, the platform promotes deeper understanding and reduces dependency on traditional tutoring methods. It is particularly beneficial for learners who prefer self-paced study, visual and audio support, or struggle with conventional learning tools.

This system can be effectively applied in various domains such as personalized e-learning platforms, virtual tutoring environments, academic support portals, and smart classroom integrations. Its ability to extract knowledge from uploaded documents and maintain conversational history makes it suitable for students, teachers, and academic institutions aiming to adopt AI-driven educational technologies.

2 REQUIREMENT SPECIFICATIONS

This chapter outlines the functional and non-functional requirements of the proposed AI-based math problem-solving and document Q&A system. It defines what the system is expected to do, the conditions under which it must operate, and the constraints it must satisfy. The chapter includes a detailed breakdown of user interactions, system inputs and outputs, performance expectations, security considerations, and interface requirements. By clearly specifying these requirements, this section serves as the foundation for system design, development, and validation in the later phases of the project

2.1 Product Scope

The scope of this software is centered around providing a web-based platform that assists users in solving math-related problems and answering questions from uploaded documents using advanced AI techniques. The system is designed to handle textual and image-based math queries through a fine-tuned language model and support document-based Q&A via a RAG (Retrieval-Augmented Generation) approach with ChromaDB. Key features include an avatar for interactive voice responses, persistent chat memory, secure email-based authentication, and user interface customization through light/dark themes.

However, the system is not intended to handle queries outside the domain of mathematics or unrelated academic topics. It does not support real-time human tutoring, video conferencing, or integration with external education management systems. The platform is built primarily for self-guided learning and academic assistance, with its functionality limited to the predefined model capabilities and document parsing logic.

2.2 Product Description

2.2.1 Product Perspective

The proposed system is a new, self-contained web application developed to offer intelligent, interactive support for solving mathematics problems and document-based question answering. It is not a replacement for existing systems but rather a modern solution that combines multiple AI-driven features to create an engaging learning environment. The platform consists of three core modules accessible through dedicated pages: a model-based page using user-generated API keys for private LLM access, a math solver page utilizing Google's Gemini Flash 2 model for image and text-based math queries, and a RAG-based page that enables document upload and semantic question answering through ChromaDB.

The system allows users to interact through a web interface with support for typed input, image uploads, and file-based queries. Responses are generated using backend AI services and delivered via text and synthesized voice, accompanied by an avatar. Chat context is preserved across sessions for continuity. Authentication is managed securely through email-based login. The modular architecture ensures each page can operate independently, enabling future integration with external services or educational platforms as needeeds.

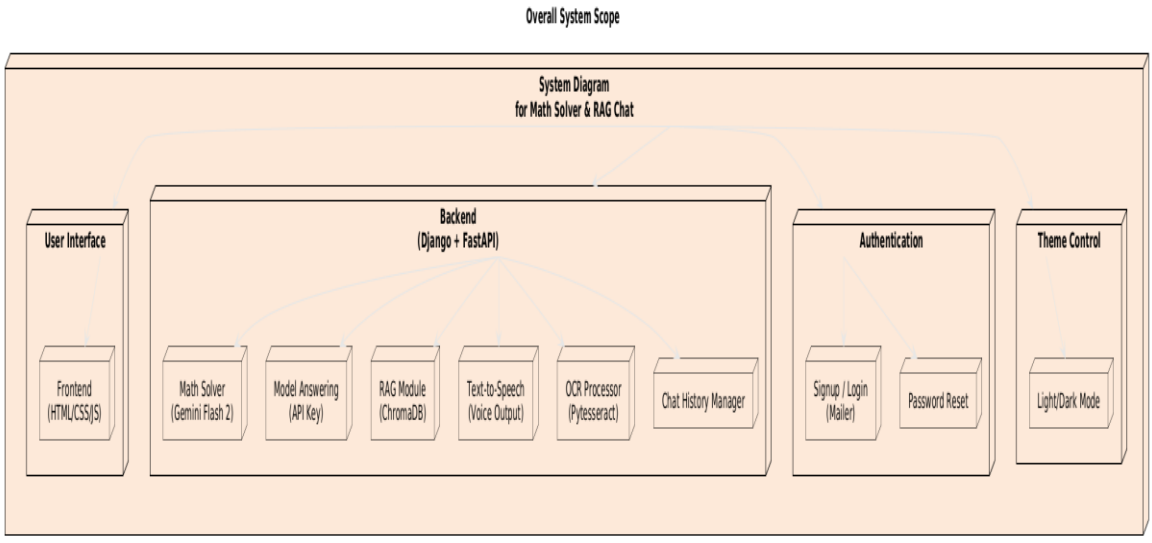

Figure 2.2-1 General Diagram

2.2.2 Product Functionality

- Allow users to securely sign up, log in, and reset passwords via email-based authentication
- Enable users to input math problems through text or upload images for processing
- Solve math problems using:
    - A custom model via user-provided API key (Model Page)
    - Gemini Flash 2 API (Solver Page)
- Accept document uploads and allow users to ask questions based on the document content (RAG Page)
- Extract text from images using integrated OCR (Pytesseract)
- Provide AI-generated responses both in text and spoken format (Text-to-Speech)
- Display avatar-based replies to enhance user engagement
- Maintain chat history and context across user sessions
- Support theme switching between light and dark modes for better accessibility
- Route user inputs to the appropriate backend logic depending on the selected page

2.2.3 Users and Characteristics

The primary users of this system are categorized based on their goals, frequency of use, and interaction

depth. The system is designed to be intuitive and accessible for a wide range of users with varying

technical expertise.

Students (Primary Users)

- **Frequency:** Frequent users
- **Technical Expertise:** Basic to moderate
- **Characteristics:** Typically enrolled in academic institutions; seeking math assistance or subject-specific document queries. They will actively use the math solver, document-based Q&A, and voice/visual outputs.
- **Needs:** Easy-to-use interface, clear step-by-step solutions, document upload functionality, and chat history for review.
- **Social Media Influencers:** High-frequency users with a need for quick and high-quality transformations. Often have higher technical expertise.
- **Support Staff:** Provide assistance and support to users. Need access to all functions for troubleshooting and user management.

Teachers & Tutors (Secondary Users)

- **Frequency:** Occasional
- **Technical Expertise:** Moderate to advanced
- **Characteristics:** May use the system to demonstrate solutions or verify student-submitted work.
- **Needs**: Access to both solver and RAG functions, focus on output clarity and step-wise breakdown.

2.2.4 Operating Environment

The proposed system is a web-based application designed to operate across a variety of modern platforms with internet connectivity. It is built using standard web technologies, ensuring compatibility with major browsers and operating systems.

2.2.4.1 Software Components

- Requires a modern web browser and a stable internet connection for uploading documents, processing math queries, and receiving voice-based responses.

2.2.4.2 Operating Systems

- Compatible with Windows, macOS, Linux, iOS, and Android.

2.2.4.3 Hardware Platform

- Standard personal computers and mobile devices with internet access.

2.2.4.4 Local Server

- Handles backend operations using Django and FastAPI, with MySQL for database management and API-based communication.

2.3 Specific Requirements

2.3.1 Functional Requirements

This section incorporates the requirements that state all the essential functions of the web application.

2.3.1.1 User Registration and Authentication

- The user must be able to create an account with a unique username and password.
- Users shall be able to log in to the platform securely.
- Users must have the ability to change their password.
- Users shall have the option to reset their password if forgotten.

2.3.1.2 Model Page (User API Key Based)

- Users can enter a personal API key to access external AI model services.
- The system should allow users to input math queries as text or upload an image.
- The backend must extract text from uploaded images using Pytesseract OCR.
- The query is sent to the specified model via API key for a solution.
- The result must be displayed along with the option to view chat history.

2.3.1.3 Solver Page (Gemini Flash 2 Integration)

- Users can enter or upload a math question.
- The system extracts content from images using Pytesseract.
- The extracted or typed query is sent to Gemini Flash 2 for solution generation.
- The response must be returned as a step-by-step explanation.
- Chat history should be stored and displayed for user reference.

2.3.1.4 RAG Page (Document-Based Q&A)

- Users can upload a document (e.g., PDF, Word).
- The system indexes the document content using ChromaDB.
- Users can ask questions related to the uploaded document.
- The model retrieves the most relevant sections and generates a response.
- The answer must be converted into speech and played using text-to-speech.

- The avatar should speak the response, enhancing user engagement.

2.3.1.5 Shared Functionalities

- Chat history should be maintained per user and per session for all pages.
- The system must allow switching between light and dark modes.
- The avatar must be consistent across modules and respond with both visual and audio output.
- Session data should persist during login sessions and clear on logout.

2.3.2 Behavioral Requirements

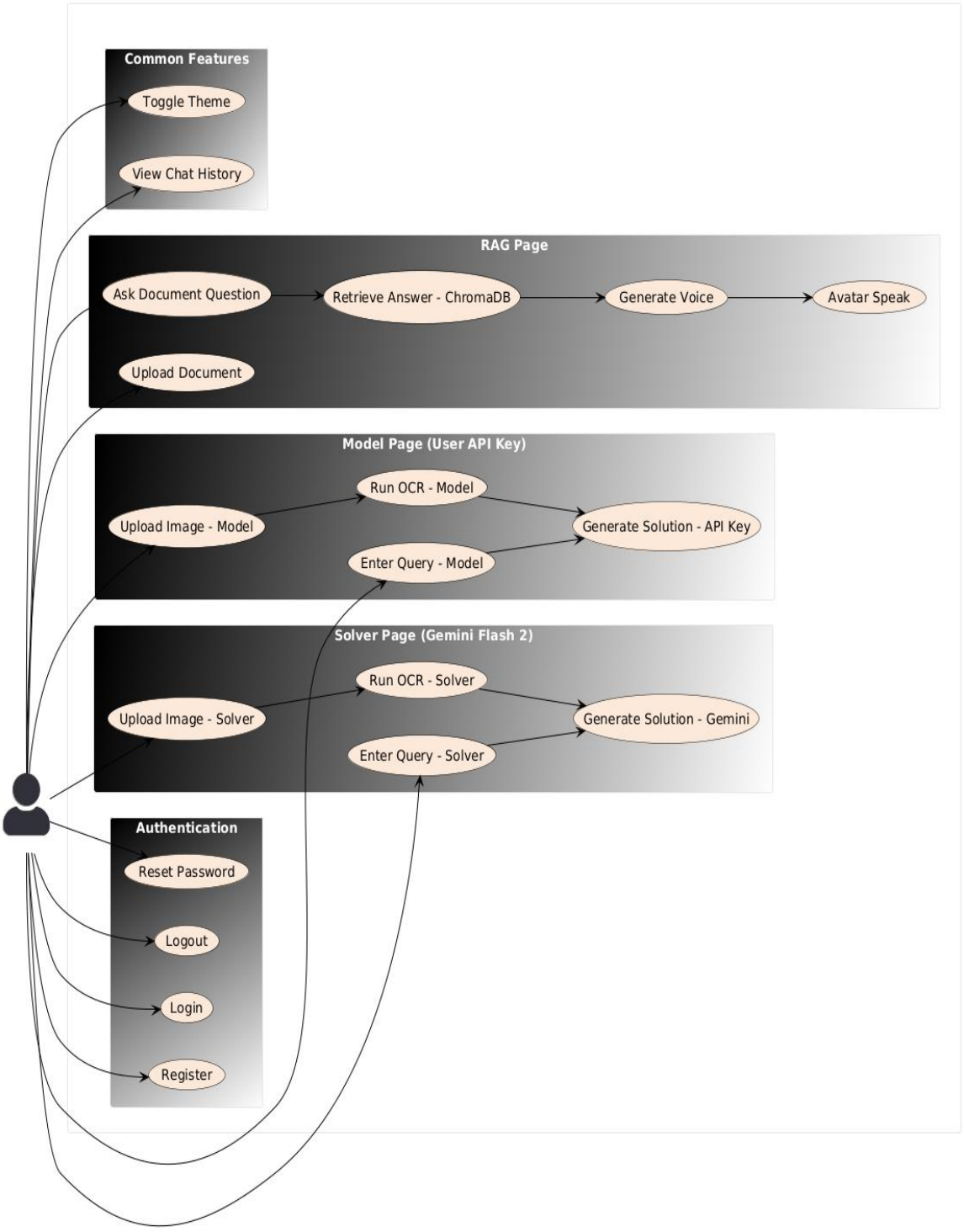

Figure 2.3-1 Use Case Diagram of System

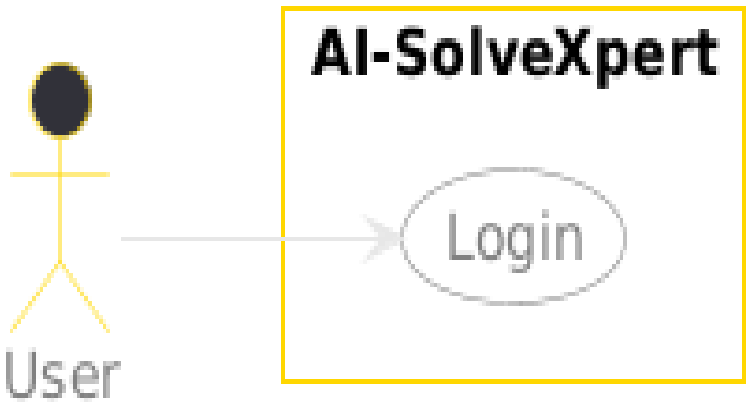

Figure 2.3-2 Use-Case#01 (Login)

Actor	User
Description	User logs into the system.
Pre-Condition	User must have registered an account.
Steps	1. User enters login credentials. 2. System verifies credentials. 3. User is granted access.
Post-Condition	User is logged in.

Table 2-1 Use-Case# 01 (Login) Description

Figure 2.3-3 Use-Case#02 (Registration)

Actor	User
Description	User registers for a new account
Pre-Condition	User must not already have an account.
Steps	1. User navigates to the registration page. 2. User enters required information. 3. User submits the registration form.
Post-Condition	User account is created.

Table 2-2 Usecase#02 (Registration) Description

Figure 2.3-4 Use-Case#03 (Change Password)

Actor	User
Description	Allows the user to update their account password.
Pre-Condition	User must be logged in.
Steps	1. User navigates to password change section. 2. User enters current password and new password. 3. User confirms the new password.
Post-Condition	Password is updated successfully.

Table 2-3 Usecase#03 (Change Password) Description

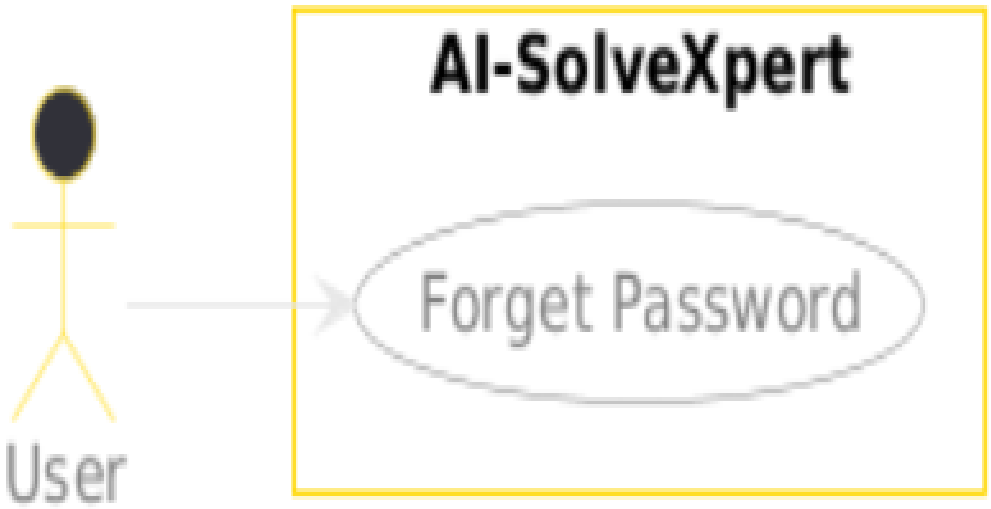

Figure 2.3-5 Use-Case#04 (Forget Password)

Actor	User
Description	Allows the user to recover or reset their password if forgotten.
Pre-Condition	User must have an account on the platform.
Steps	1. User clicks "Forgot Password" option. 2. User provides account recovery information (e.g., email). 3. User follows instructions to reset password.
Post-Condition	User's password is reset.

Table 2-4 Usecase#04 (Forget Password) Description

Figure 2.3-6 Use-Case#05 (Upload image)

Actor	User
Description	Allows the user to upload image of question or Write question for AI-Solver Solution for processing.
Pre-Condition	User must on Upload Solver Page.
Steps	1. User selects a Photo file to upload. 2. User initiates the upload process.
Post-Condition	Image is uploaded successfully.

Table 2-5 Usecase#05 (Upload Image) Description

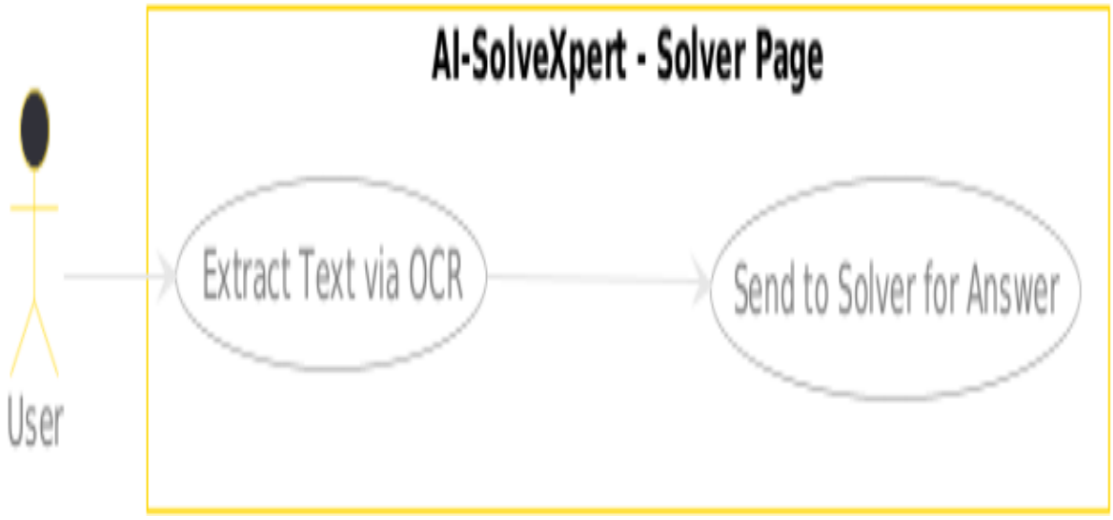

Figure 2.3-7 Use-Case#06 (Extracting text from image)

Actor	System
Description	Extracts text from an uploaded image using OCR and sends it to the AI solver for generating an answer.
Pre-Condition	User uploads an image containing a question.
Steps	1. System receives the uploaded image. 2. System extracts text from the image using OCR. 3. System sends the extracted text to the solver model..
Post-Condition	The user receives a solution to the extracted question from the uploaded image.

Table 2-6 Usecase#06 (Extracting text from image) Description

Figure 2.3-8 Use-Case#07 (System Output)

Actor	System
Description	Prepares the extracted question and AI-generated solution for display and copying.
Pre-Condition	The AI solver has completed processing the question.
Steps	1. System displays the extracted question and its corresponding solution. 2. System provides a copy button to allow the user to copy the solution.
Post-Condition	The solution is visible and available for the user to copy.

Table 2-7 Usecase#07 (System Output) Description

Figure 2.3-9 Use-Case# 8 (Upload Document)

Actor	User
Description	Allows the user to upload a document for processing.
Pre-Condition	The system is ready to accept input.
Steps	1. User selects and uploads a document (e.g., image or PDF). 2. System receives the file and initiates text extraction.
Post-Condition	Document is uploaded successfully, and its content is ready for OCR processing.

Table 2-8 Usecase#08 (Upload Document) Description

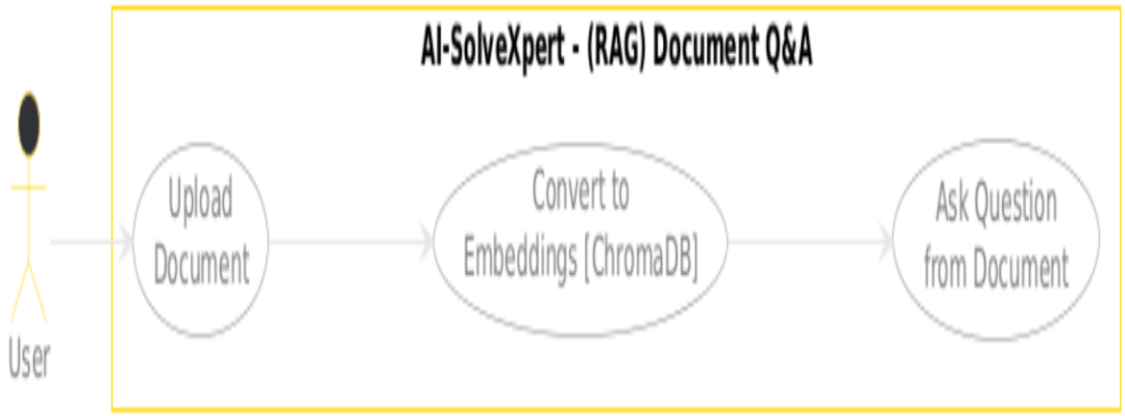

Figure 2.3-10 Use-Case#9 (Creating Embeddings)

Actor	User
Description	Allows the user to convert uploaded documents into vector embeddings for storage and future question answering.
Pre-Condition	Document is uploaded and ready for processing.
Steps	1. User uploads a document (PDF, docx, etc.). 2. System extracts text using OCR or parsing tools. 3. System converts the extracted text into embeddings. 4. Embeddings are stored in ChromaDB for semantic retrieval.
Post-Condition	Document embeddings are stored successfully and are ready for answering user queries.

Table 2-9 Usecase#9 (Creating Embeddings) Description

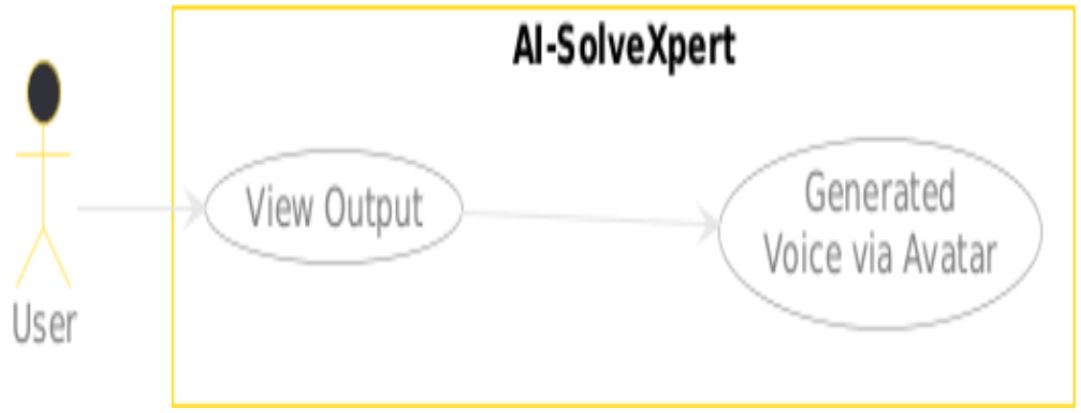

Figure 2.3-11 Use-Case#10 (Voice with Avatar)

Actor	User
Description	Allows the user to listen to the generated answer read aloud using a avatar, with an option to mute the voice.
Pre-Condition	Answer has been generated.
Steps	1. System reads the generated text using an avatar. 2. User has the option to mute or unmute the avatar at any time.
Post-Condition	User listens to or mutes the voice avatar reading the answer.

Table 2-10 Usecase#10 (Voice with Avatar) Description

Figure 2.3-12 Use-Case#11 (Input)

Actor	User
Description	Allows the user to upload image of question or Write question for AI-Solver Solution for processing.
Pre-Condition	User must on Upload Solver Page.
Steps	1. User selects a Photo file to upload. 2. User initiates the upload process.
Post-Condition	Image is uploaded successfully.

Table 2-11 Usecase#11 (Input) Description

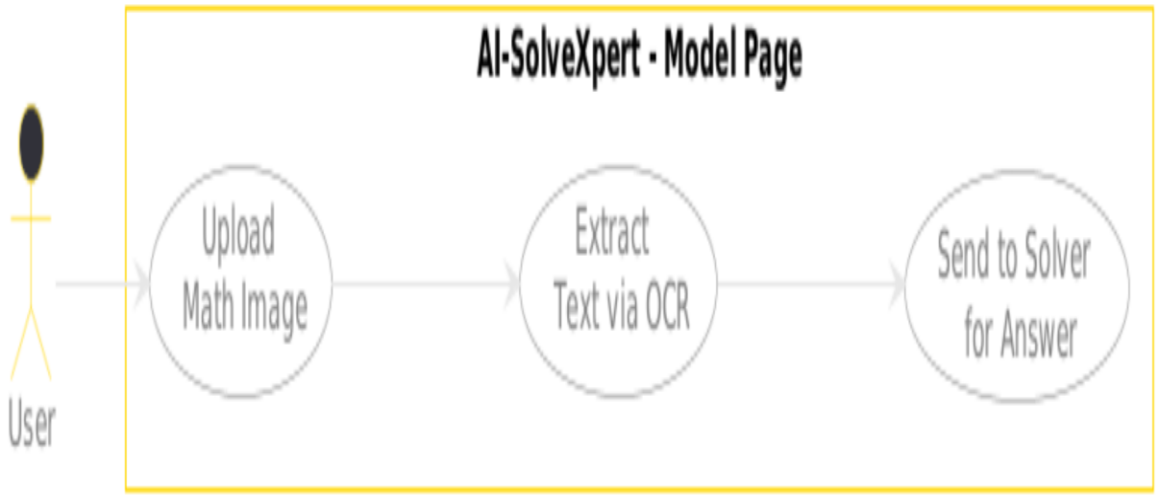

Figure 2.3-13 Use-Case#12 (OCR)

Actor	System
Description	Extracts text from an uploaded image using OCR and sends it to the Finetuned model for generating an answer.
Pre-Condition	User uploads an image containing a question.
Steps	1. System receives the uploaded image. 2. System extracts text from the image using OCR. 3. System sends the extracted text to the finetuned model.
Post-Condition	The user receives a solution to the extracted question from the uploaded image.

Table 2-12 Usecase#12 (OCR) Description

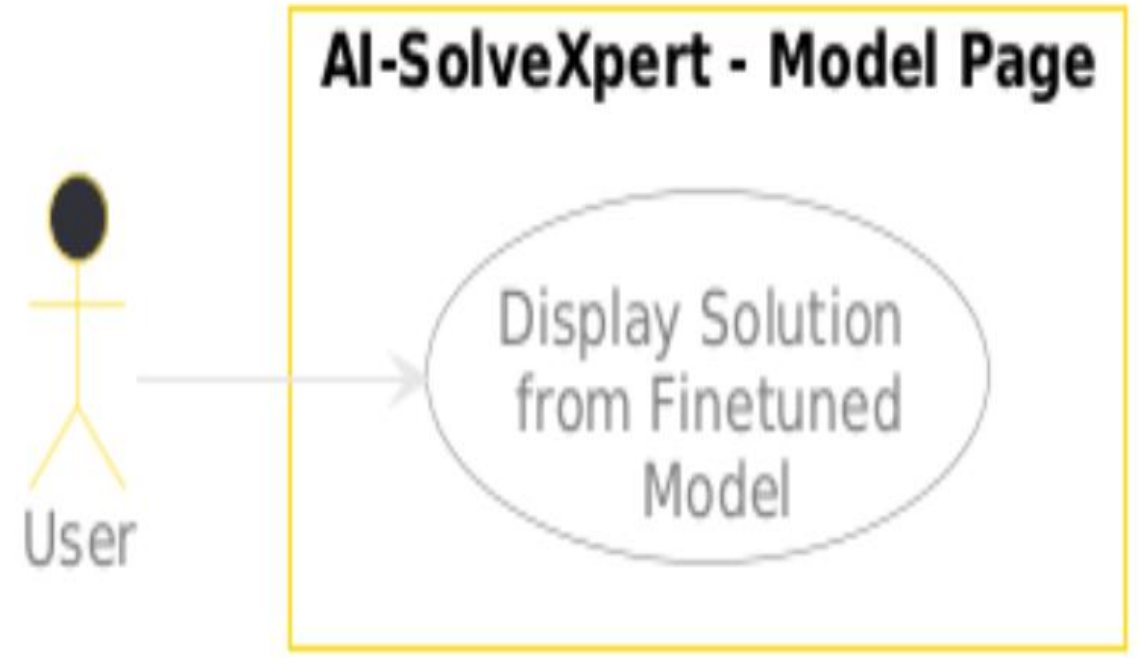

Figure 2.3-14 Usecase#13 (Solution)

Actor	System
Description	Prepares the extracted question and finetuned model solution for display and copying.
Pre-Condition	The finetuned model has completed processing the question.
Steps	1. System displays the extracted question and its corresponding solution. 2. System provides a copy button to allow the user to copy the solution.
Post-Condition	The solution is visible and available for the user to copy.

Table 2-13 Usecase#13 (Solution) Description

2.3.3 External Interface Requirements

2.3.3.1 User Interface

2.3.3.1.1 Home (Extension Interface) Page

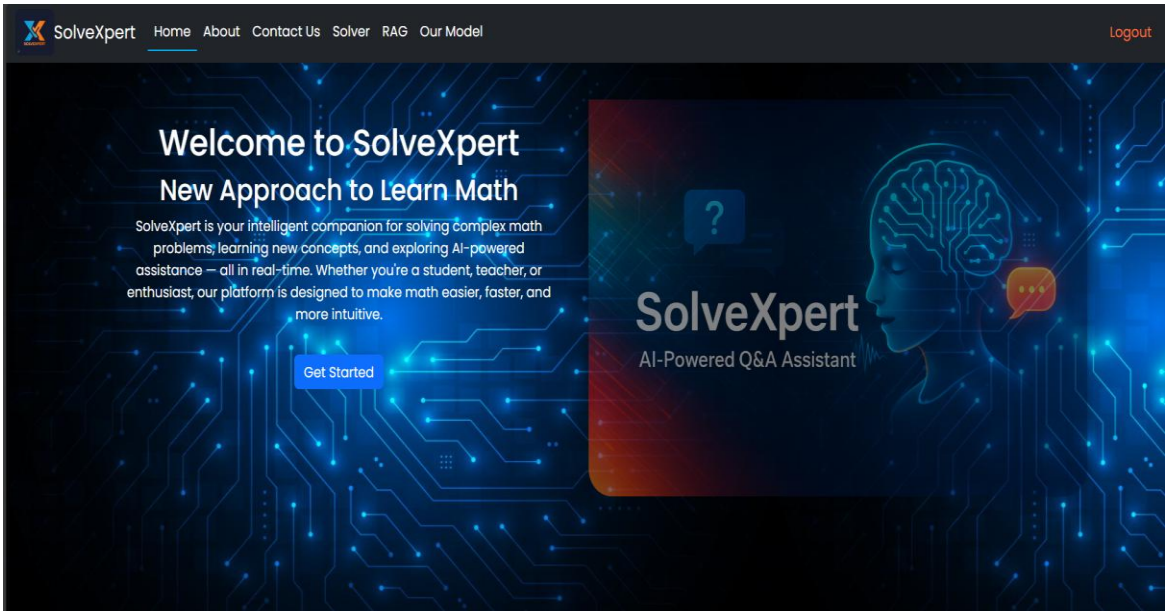

Figure 2.3-15 Home

2.3.3.1.2 About Page

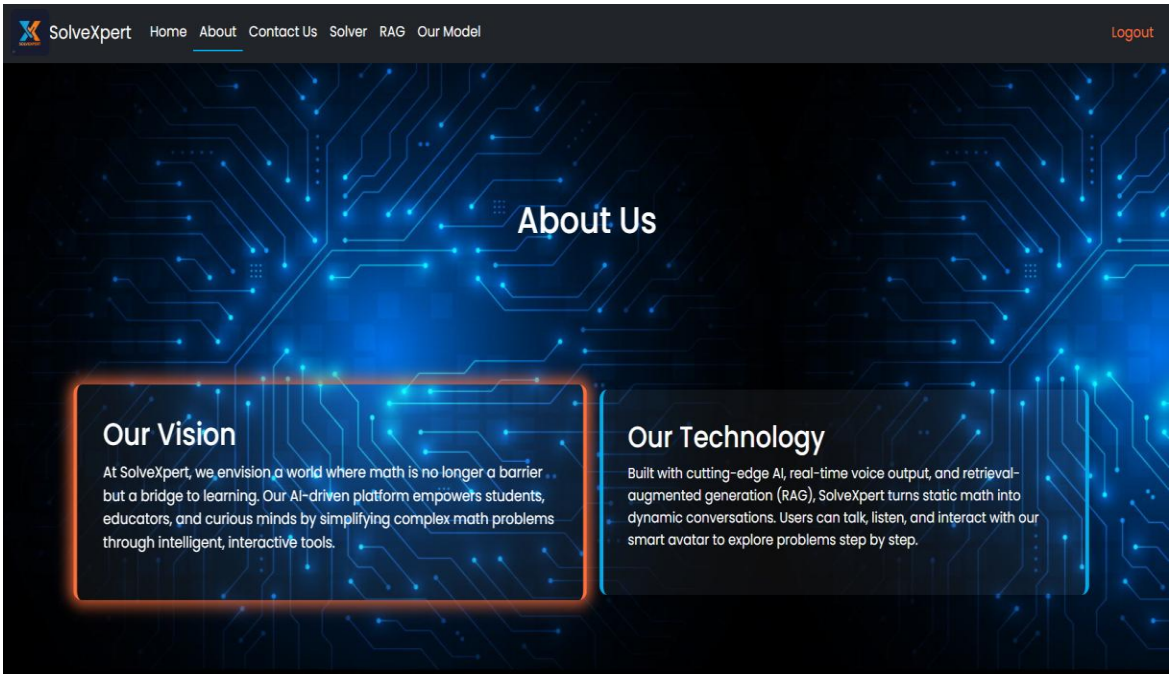

Figure 2.3-16 About Page

2.3.3.1.3 Contact Page

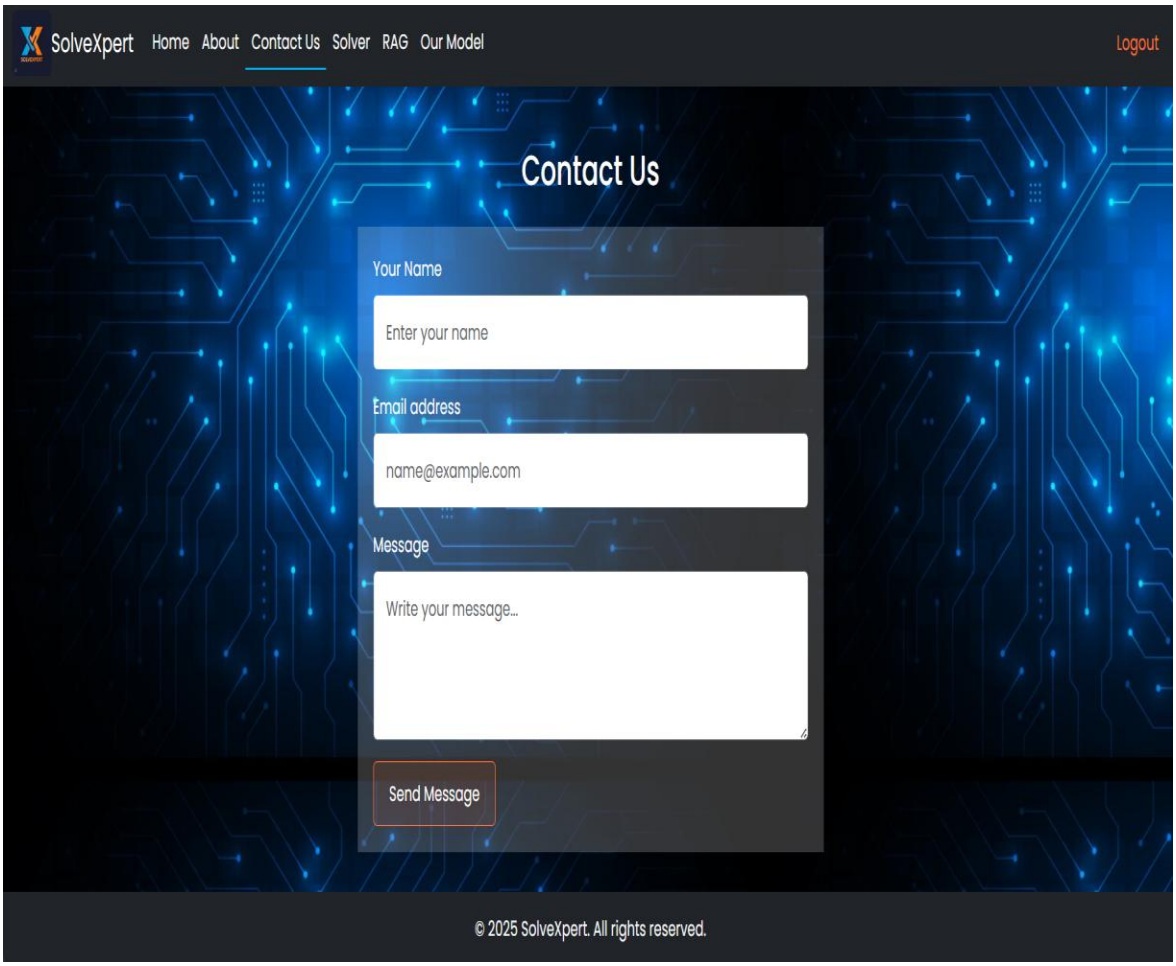

Figure 2.3-17 Contact Page

2.3.3.1.4 RAG Page

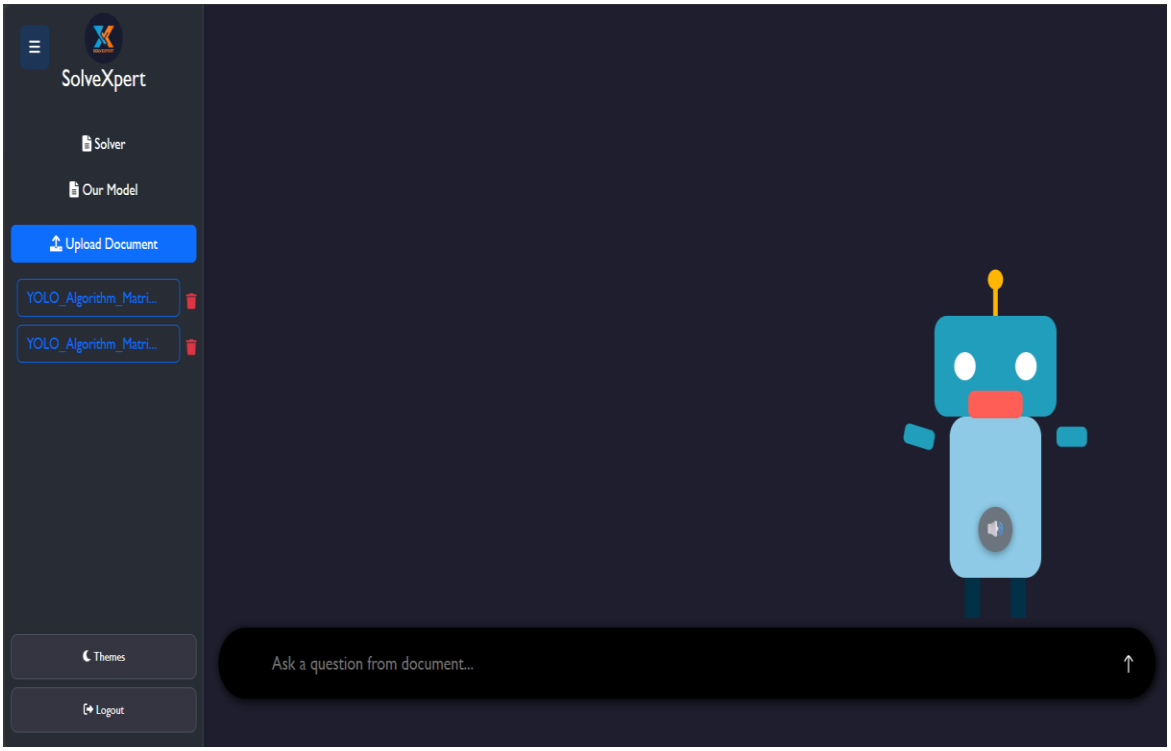

Figure 2.3-18 RAG page

2.3.3.1.5 Solver/Finetuned Model Page

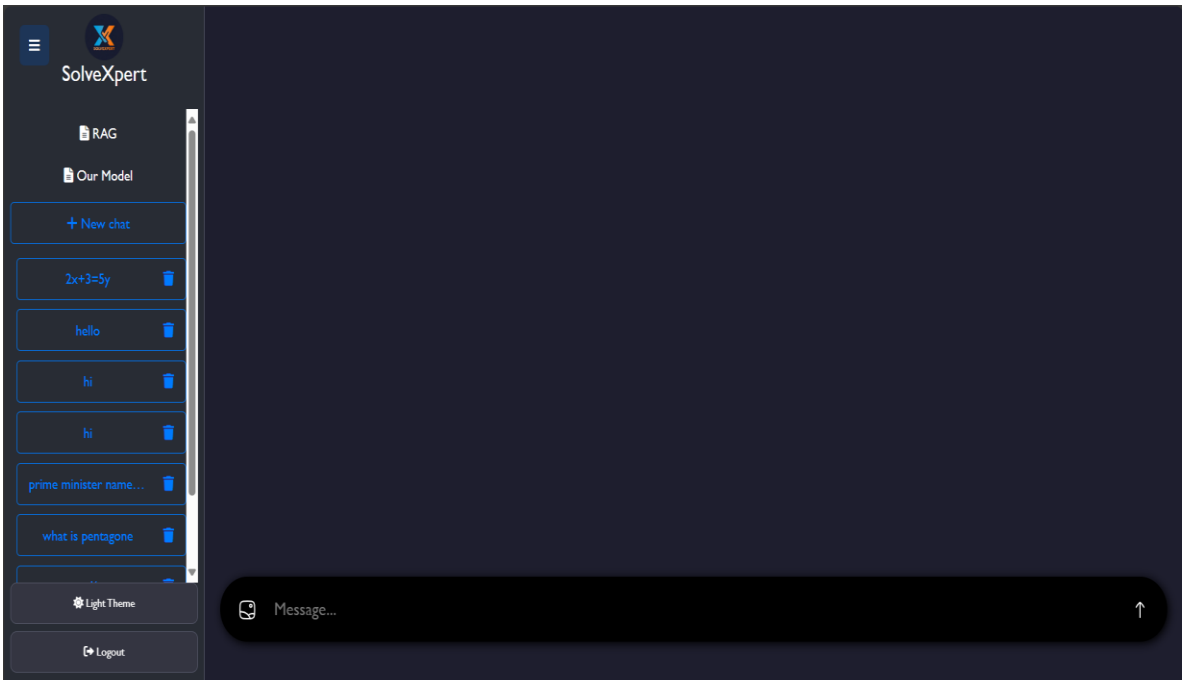

Figure 2.3-19 Solver/Finetuned Model Page

2.4 Non-Functional Requirements

2.4.1 Performance Requirements

- Any user-submitted query (text or OCR-extracted content) must return a response within **10 seconds** under normal load.
- Under peak conditions, the response time must not exceed **15 seconds**.
- The system must extract text from uploaded images or documents within **5 seconds** for standard resolution files (up to 5MB).
- For higher resolution files (5MB–10MB), extraction must complete within **8 seconds**.
- Voice playback of answers must begin within **2 seconds** of answer generation.
- Uploaded documents must be processed and embedded in **ChromaDB** within **10 seconds** for files up to 10 pages.
- For longer documents, processing time must scale linearly but not exceed **30 seconds**.
- All UI interactions (e.g., chat scrolling, theme toggling, button clicks) should complete within **100 ms** to ensure a smooth user experience.
- The system must support **at least 100 concurrent users** without performance degradation beyond specified limits.

2.4.2 Safety and Security Requirements

- **Data Security:** All user inputs, including questions, uploaded documents, and images, must be transmitted over secure HTTPS connections.
- Data stored in the database (e.g., chat history, document embeddings) must be encrypted at rest using standard encryption protocols (e.g., AES-256).
- **Privacy:** Uploaded documents are processed temporarily and retained only for as long as necessary for embedding and response generation.
- Any documents not associated with persistent user sessions must be automatically deleted after processing.
- **Authentication & Session Security:** All users must authenticate via a secure login system.
- Sessions must expire after a defined period of inactivity (e.g., 30 minutes).
- CSRF and XSS protection must be enforced across all forms and inputs.
- **Compliance with Regulations:** The application must comply with relevant data protection laws such as GDPR (for users in the EU) and local data handling policies applicable in the deployment region.
- User consent must be obtained before processing or storing any personal data.
- **User-Controlled Data Management:** Users should have the ability to delete their chat history and uploaded documents on demand.
- A confirmation prompt must be shown before permanent data deletion.
- **Embedding Storage Integrity:** Document embeddings stored in ChromaDB must be isolated per user.
- Access control mechanisms must ensure that users cannot retrieve or query other

users embedded content.

- **Audit and Logging:** All authentication and data access actions must be logged securely to assist in auditing and incident investigation.
- Logs must not contain sensitive user content.

2.4.3 Software Quality Attributes

2.4.3.1 Reliability:

- The system shall be available **99.9% of the time**, excluding scheduled maintenance.
- Automatic error logging and alerting mechanisms shall be implemented to identify runtime failures.
- Redundant server architecture (or fallback responses) shall be used to prevent total service outages.

2.4.3.2 Portability:

- The application shall be deployable across major operating systems (Linux, Windows, macOS) and cloud environments.
- Containerization using Docker shall be used to ensure smooth portability between development, testing, and production environments.

2.4.3.3 Usability:

- The system interface will be intuitive, with clear instructions and tooltips for first-time users.
- Copy buttons, chat history, and avatar voice features will be simple to interact with.
- A maximum of 2 user clicks should be required to reach core functionalities such as document upload or solving queries.

3 DESIGN SPECIFICATIONS

This chapter presents the design architecture of the SolveXpert system from multiple viewpoints to offer a comprehensive understanding of its structure and behavior. These design specifications provide visual and descriptive representations that align with the defined functional requirements. The diagrams and explanations illustrate how different components interact, define system boundaries, and demonstrate how the system's core features—such as question answering, document processing, and voice output—are implemented to ensure clarity, efficiency, and scalability.

3.1 Introduction

This chapter outlines the structural and behavioral design of the SolveXpert system. It includes diagrams such as use case, class, sequence, and activity diagrams to visually represent the system's functionality, data flow, and user interactions. These design elements help clarify how various components work together to meet the project's requirements, ensuring a scalable and maintainable solution.

3.2 Composite Viewpoint

This section presents the high-level composition and modular structure of the SolveXpert system. It illustrates how various subsystems and components interact logically and physically. The composite viewpoint provides a clear understanding of system organization in terms of packages and deployment environments. Below is a brief description of each:

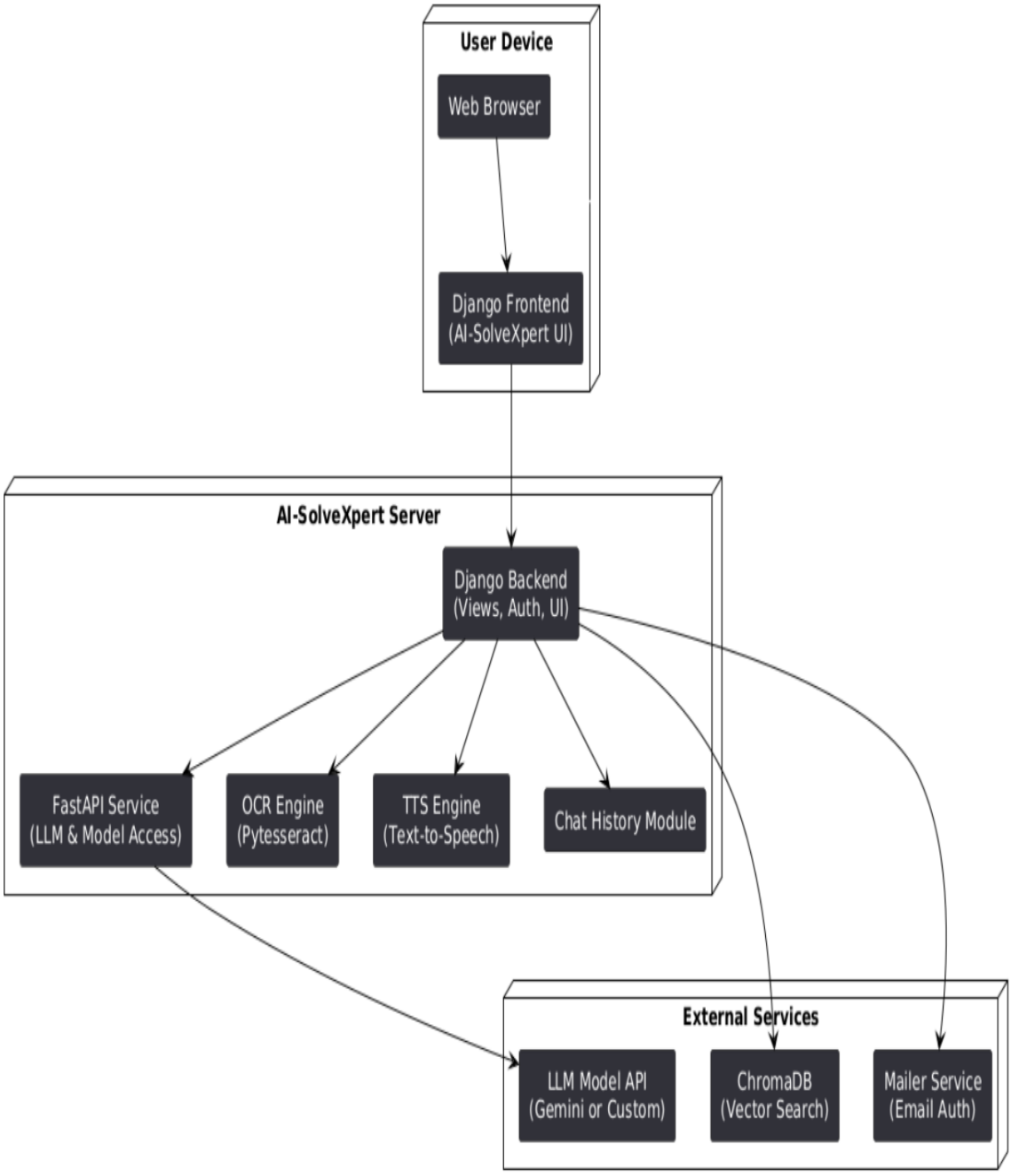

Figure 3.2-1 Deployment Diagram.

3.3 Logical Viewpoint

Shows how the system is broken down into logical subsystems/modules such as:

- **User Interface**: Handles frontend views and interactions.
- **Authentication**: Manages user login, registration, and session handling.
- **OCR & Document Handling**: Manages file upload, text extraction, and formatting.
- **LLM Processing**: Interfaces with the LLM model (e.g., LLaMA3) for solving queries.
- **Database Management**: Stores users, questions, answers, and embeddings.

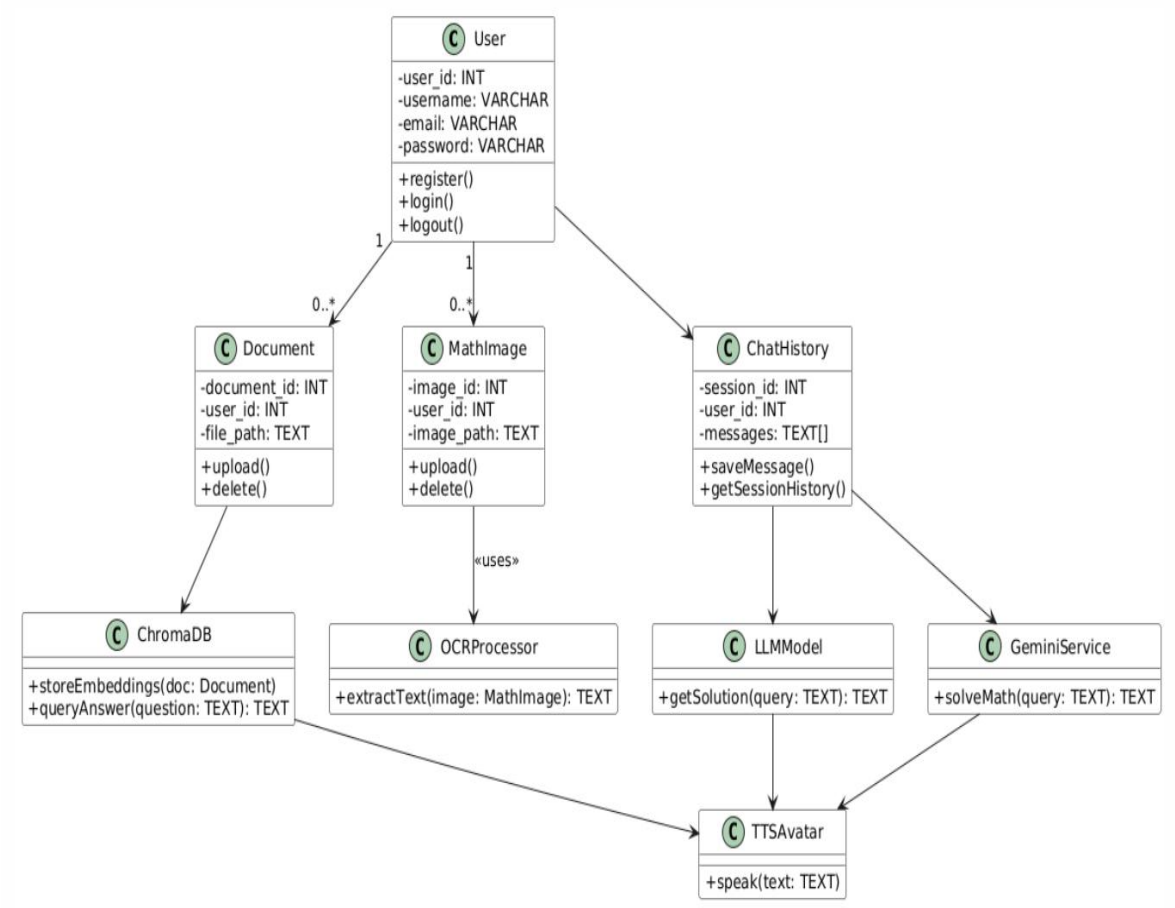

Figure 3.3-1 Class Diagram

3.4 Information Viewpoint

An Entity Relationship Diagram (ERD) is used to represent the data storage requirements for the system.

Figure 3.4-1 ER Diagram

3.5 Interaction Viewpoint

In this diagram we can see the sequence of the functionalities that is performing in the system and what is the system response on the functionality is shown.

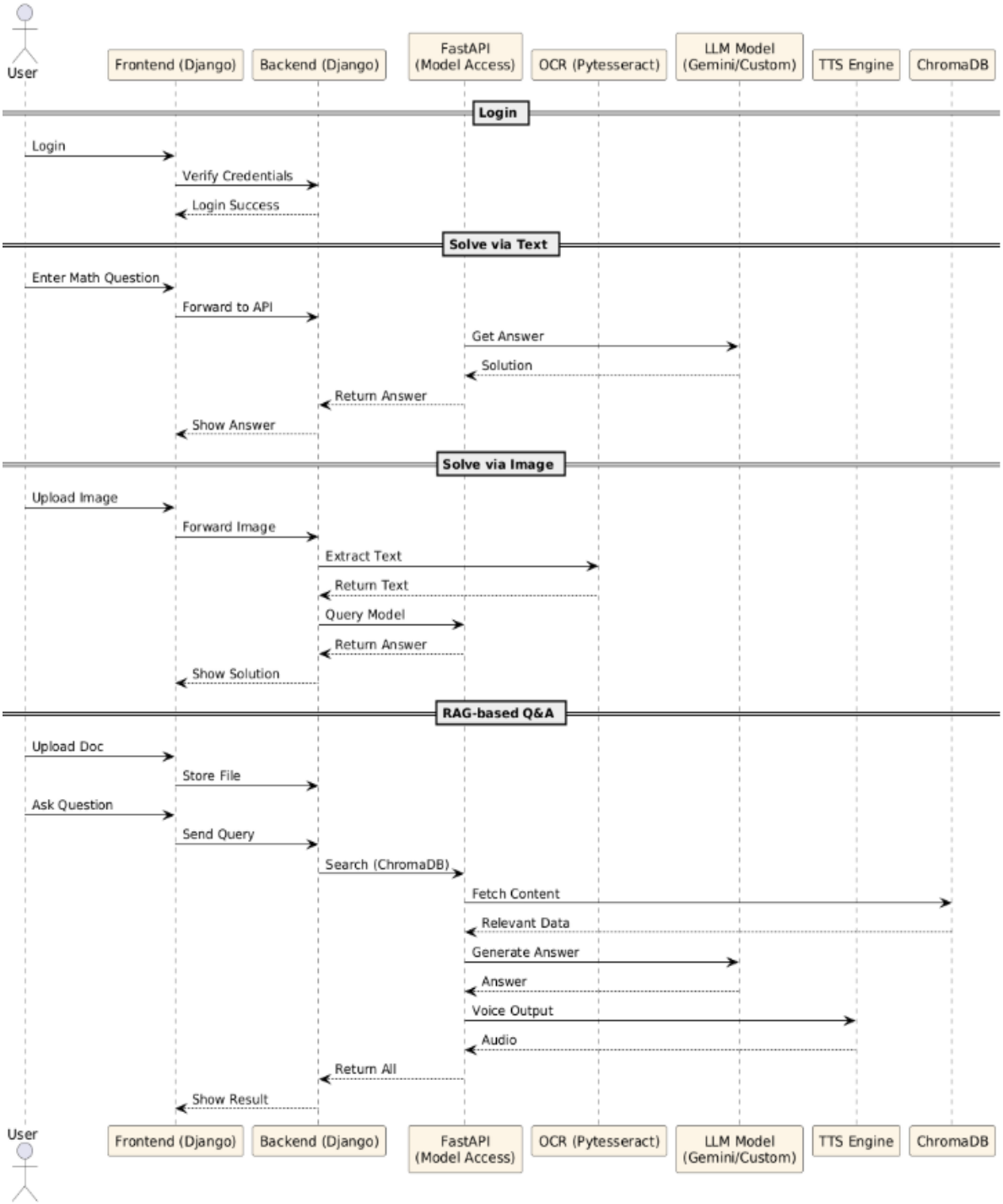

Figure 3.4-2 Sequence Diagram

3.6 Static Dynamic Viewpoint

The State Dynamic Viewpoint illustrates the system's state changes in response to events using a UML State Machine Diagram.

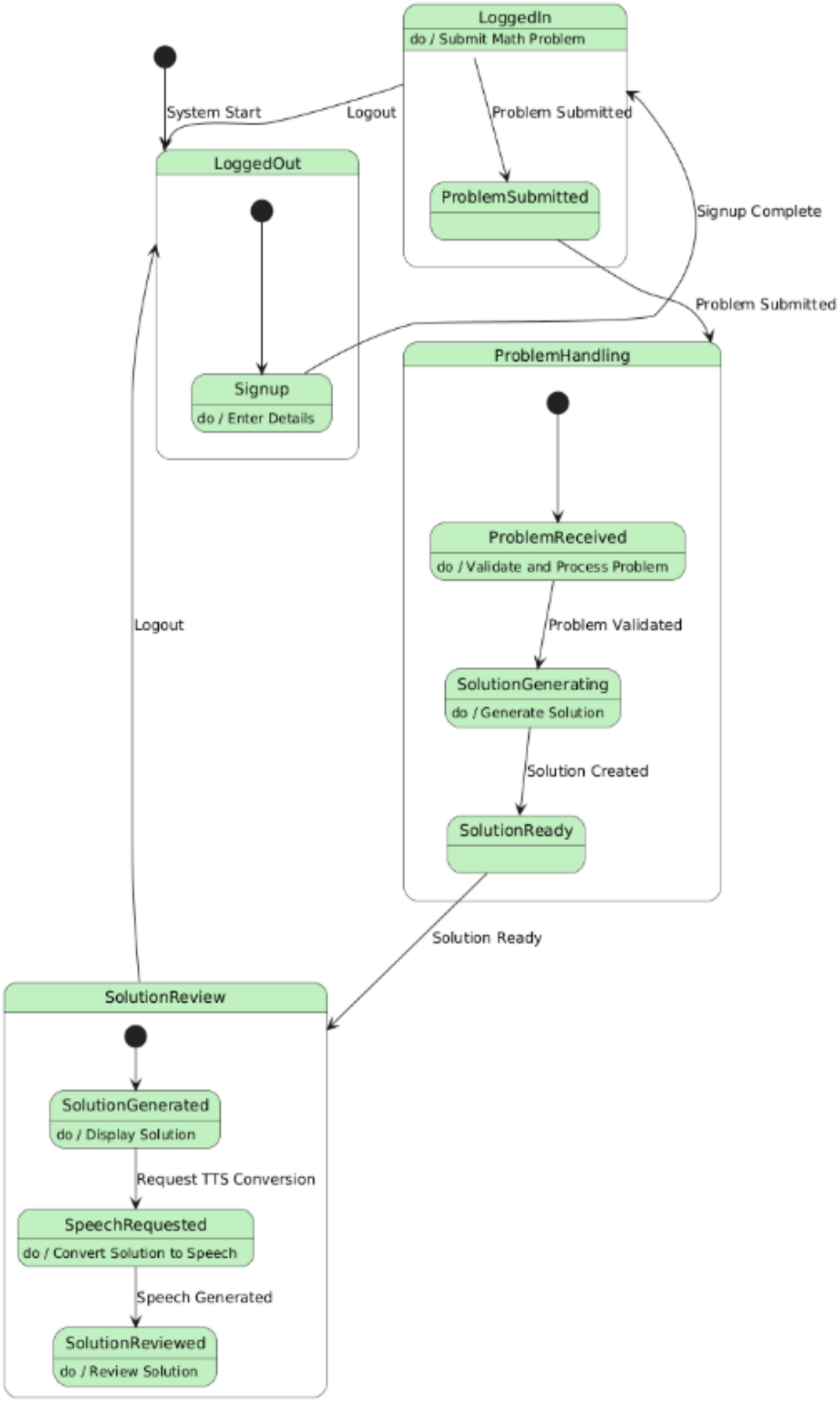

Figure 3.4-3 State Dynamic Viewpoint Diagram

3.7 Algorithm Viewpoint

The Algorithm Viewpoint outlines the operations of the system's algorithms using pseudo code and decision tables.

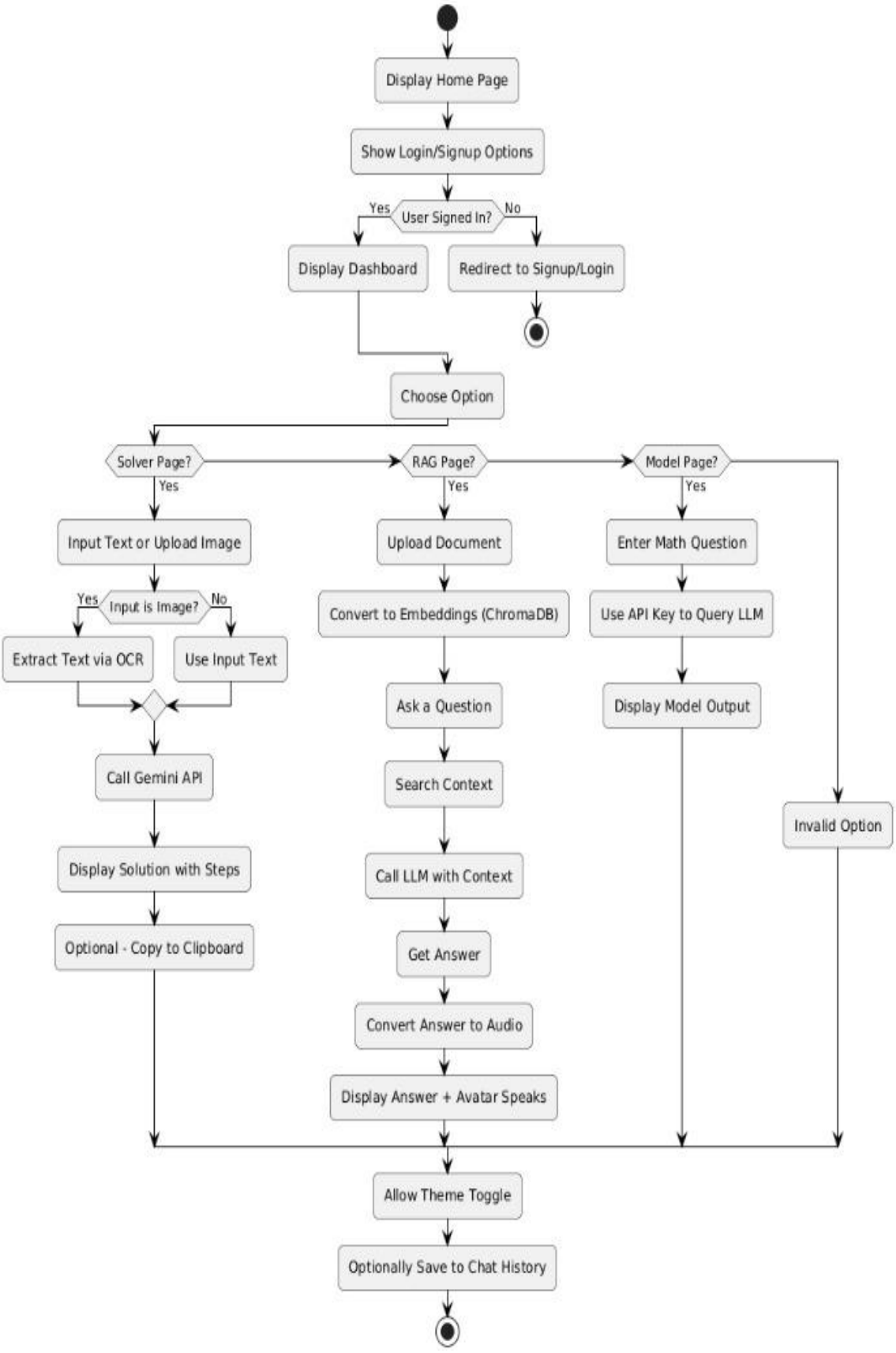

Figure 3.4-4 Algorithmic View

Pseudo Code:

User Signup:

```
Function signup(username, password, email)
  If username, password, email are valid
    If user does not exist
      Create new user with username, password, email
      Return success message "✅ User created successfully"
    Else
      Return error message "⚠️ User already exists"
  Else
    Return error message "❌ Invalid input"
End Function
```

User Login:

```
Function login(username, password)
  If username, password are valid
    If user exists and password matches
      Generate session token
      Return success message "✅ Login successful"
    Else
      Return error message "⚠️ Invalid username or password"
  Else
    Return error message "❌ Invalid input"
End Function
```

Contact Us Form Submission:

```
Function submitContactForm(name, email, message)
  If name, email, and message are not empty
    If email is in valid format
      Save message to database
      Send confirmation or thank-you message to user
      Notify admin about new message
      Return "✅ Your message has been submitted"
    Else
      Return "❌ Invalid email format"
  Else
    Return "⚠️ All fields are required"
End Function
```

Model Page:

```
Function modelPageSolve(api_key, question)
  If api_key is valid
    Send question to FastAPI model using api_key
    Receive response from model
    Save question and response to chat history
    Return model response
  Else
    Return error "❌ Invalid API key"
End Function
```

RAG Page:

```
Function askFromDocument(document, question)
  If document is uploaded
    Split and clean document
    Generate embeddings
    Store embeddings in ChromaDB
  context ← searchChromaDB(question)
  answer ← queryModelWithContext(context, question)
  audio ← convertTextToSpeech(answer)
  Display avatar with voice response
  Save question and answer to chat history
  Return answer + audio
End Function
```

Solver Page:

```
Function solveMath(input)
  If input is an image
    text ← extractTextUsingOCR(input)
  Else
    text ← input
  response ← callGeminiAPI(text)
  Save question and response to chat history
  Return response
End Function
```

Theme Toggle Functionality:

```
Function toggleTheme(currentTheme)
  If currentTheme is "light"
    Set theme to "dark"
    Save preference in user session or database
    Apply dark CSS classes to UI
    Return "Dark mode activated"
  Else If currentTheme is "dark"
    Set theme to "light"
    Save preference in user session or database
    Apply light CSS classes to UI
    Return "Light mode activated"
  Else
    Return error "❌ Invalid theme state"
End Function
```

Copy Answer to Clipboard

```
Function copyToClipboard(answerText)
  If answerText is not empty
    Copy answerText to system clipboard
    Show message "✅ Answer copied successfully"
  Else
    Show warning "⚠️ Nothing to copy"
  End If
End Function
```

Frontend to Backend Communication:

```
Function frontendInteraction(action, parameters)
  Switch (action)
    Case "signup":
      Call signup(username, password, email)
    Case "login":
      Call login(username, password)
    Case "uploadImage":
      Call uploadImage(sessionToken, imageFile)
    Case "solveWithModel":
      Call queryModel(sessionToken, apiKey, question)
    Case "solveWithGemini":
      Call solveMath(sessionToken, text)
    Case "uploadDocument":
      Call uploadDocument(sessionToken, document)
    Case "askFromDocument":
      Call askFromDocument(sessionToken, question)
    Case "playVoice":
      Call generateVoice(sessionToken, text)
    Case "toggleTheme":
      Call toggleTheme(sessionToken, currentTheme)
    Case "copyAnswer":
      Call copyToClipboard(text)
    Case "sendContact":
      Call submitContactForm(name, email, message)
    Default:
      Return "⚠ Unknown Action"
  End Switch
End Function
```

4 DEVELOPMENT AND TOOLS

This chapter provides an overview of the development process, including the tools and technologies used to implement the system. It discusses the choice of programming languages, frameworks, databases, and other software components that support the development of the system. Additionally, it covers the environment setup, integration of various modules, and key tools used to ensure the smooth and efficient development of the product.

4.1 Introduction

This chapter outlines the tools, technologies, and development methodologies employed to build the system. It highlights the programming languages, frameworks, and software tools used for designing, implementing, and testing the system. Additionally, it provides a brief overview of the development environment and the integration of various components that ensure the successful creation and deployment of the product.

4.2 Development Plan

This project is developed by a team of two members.

- MUHAMMAD FURQAN
- HASEEB UL HASSAN

The whole development plan is given below.

Table 4-1 Work Breakdown Structure

Task Name	Duration	Start	Finish
Project Kickoff	3 days	Thu 10/10/24	Mon 10/14/24
Requirements Gathering	10 days	Mon 10/14/24	Fri 10/25/24
System Design	7 days	Mon 10/28/24	Tue 11/5/24
Frontend Development	35 days	Wed 11/6/24	Wed 12/25/24
- Registration Page	2 days	Wed 11/6/24	Fri 11/8/24
- Login Page	2 days	Mon 11/11/24	Wed 11/13/24
- Solver Page (Model API)	10 days	Wed 11/13/24	Wed 11/27/24

Task Name	Duration	Start	Finish
- RAG Page (Document Q&A)	10 days	Thu 11/28/24	Thu 12/12/24
- Output + Avatar Response	11 days	Fri 12/13/24	Wed 12/25/24
Backend & AI Modules	25 days	Thu 12/26/24	Fri 1/31/25
- Gemini API Integration	6 days	Thu 12/26/24	Thu 1/2/25
- ChromaDB Embedding	6 days	Fri 1/3/25	Fri 1/10/25
- TTS + Avatar System	6 days	Mon 1/13/25	Mon 1/20/25
- Chat History Management	7 days	Tue 1/21/25	Fri 1/31/25
Testing	15 days	Mon 2/3/25	Fri 2/21/25
- Unit Testing	3 days	Mon 2/3/25	Wed 2/5/25
- Functional Testing	4 days	Thu 2/6/25	Tue 2/11/25
- Interface Testing	4 days	Wed 2/12/25	Mon 2/17/25
- Final QA & Debugging	4 days	Tue 2/18/25	Fri 2/21/25
Final Documentation	30 days	Mon 2/24/25	Thu 4/10/25
- Review Design & Screenshots	7 days	Mon 2/24/25	Tue 3/4/25
- Write Report Chapters	10 days	Wed 3/5/25	Tue 3/18/25
- Final Compilation	13 days	Wed 3/19/25	Thu 4/10/25
Submission & Closeout	7 days	Fri 4/11/25	Thu 4/17/25

4.3 Development Tool

The tools used for the development of this project are given below:

1. Programming Language
    - Python
    - HTML5
    - CSS3
    - JavaScript
2. Framework
    - Django
3. Database Management System
    - MySQL
4. Development Environment
    - VS Code

5. Libraries and Packages
    - Torch
    - Ffmpeg
    - Numpy
    - Transformers
    - ChromaDB
6. Version Control
    - GIT (for version control)
    - GITHUB (for repository hosting and collaboration)
7. Operating System
    - Windows

4.4 Conclusion and Future Work/Extensions

The current implementation of **SolveXpert** successfully provides essential features like document processing, question answering using AI, and embedding storage via ChromaDB. Potential future enhancements include:

- **Real-Time Answer Generation**: Providing instant answers as users ask questions.
- **Enhanced AI Models**: Further improving the quality and accuracy of the AI responses.
- **Mobile App**: Expanding accessibility with a mobile application for on-the-go use.
- **Multilingual Support**: Enabling support for multiple languages to serve a global audience.
- **User Analytics**: Offering insights into user preferences to enhance the experience and provide tailored recommendations.
- **Integration with More Data Sources**: Enabling the system to interact with additional document formats and data sources for broader applicability.

5 QUALITY ASSURANCE

This section outlines the approach to ensuring the quality of the **SolveXpert** project. It includes the creation of test plans and the execution of test cases to ensure the delivery of a high-quality product. We have focused on maintaining the quality throughout the project development cycle and have documented various test plans and cases that were executed during the development process. These tests are designed to validate the system's functionality, security, and overall performance to meet the expected standards.

Requirement ID	Test Case ID	TC_1	TC_2	TC_3	TC_4	TC_5	TC_6	TC_7	#Test Cases for respective requirement
USR-FR:1		✔							1
USR-FR:2			✔						1
USR-FR:3				✔					1
USR-FR:4					✔				1
USR-FR:5						✔			1
USR-FR:6							✔		1
USR-FR:7								✔	1

Table 5-1 Requirement Traceability Matrix (User FR)

5.1 Introduction

Quality Assurance (QA) is a systematic process aimed at ensuring that the **SolveXpert** application meets the predefined quality standards and user expectations. It involves setting quality benchmarks, planning and conducting tests, documenting results, and iterating for improvement. The QA process in this project focuses on verifying core functionalities such as document upload, question answering, text-to-speech with avatar, and chat history management to ensure they are reliable, secure, and user-friendly.

5.2 Traceability Matrix

The Requirement Traceability Matrix (RTM) maps each functional requirement of the **SolveXpert** system to its corresponding test case(s). This ensures that all specified requirements are tested, validated, and implemented correctly. It provides transparency and accountability, guaranteeing that no requirement is overlooked during development and testing.

5.3 Test Plan

This section outlines the approach, scope, resources, and schedule for testing activities of the **SolveXpert** project. It ensures that all functionalities meet specified requirements and that the system performs reliably under expected conditions.

5.3.1 References

The following documents provide the basis and context for this test plan:

- **Project Plan**: Defines project scope, timelines, and responsibilities.
- **IEEE Test Plan Document**: Serves as a template for structured and standardized test planning.
- **Software Requirements Specification (SRS)**: Provides a detailed list of functional and non-functional requirements to be validated.
- **Detailed Design Document**: Describes the internal structure and logic used to implement functionalities that are to be tested.

5.3.2 Test Items

The systems and modules to be tested in the **SolveXpert** application include:

- **User Module:** Registration, login, and chat session management.
- **Solver Module:** Gemini API integration for answering user questions.
- **Document Module:** Uploading documents, converting them into embeddings using ChromaDB, and querying based on user input.
- **Output Module:** Response display with avatar-based voice output and mute functionality.
- **Chat History Module:** Storing and retrieving past interactions for each user session.

5.3.3 Features To Be Tested

Features to be tested include the following:

- **USR-FR1**: User Registration and Authentication
- **USR-FR2**: Chat Interface and Message Handling
- **USR-FR3**: Gemini API Integration for Answer Generation
- **USR-FR4**: Document Upload and Storage
- **USR-FR5**: Embedding Creation via ChromaDB
- **USR-FR6**: Retrieval-Augmented Generation (RAG) Question Answering
- **USR-FR7**: Voice Avatar Output with Mute Option
- **USR-FR8**: Chat History Storage and Retrieval

5.3.4 Approach

Testing will follow these methods:

- **Unit Testing**: Verifying individual components such as API endpoints, database operations, and logic for input processing.
- **Black Box Testing**: Ensuring the chatbot, document upload, and output features behave as expected without knowing the internal code.
- **Integration Testing**: Validating communication between modules such as the frontend chat interface, Gemini API, ChromaDB embeddings, and TTS avatar system.
- **System Testing**: Testing the entire SolveXpert application as a whole to ensure it meets functional and non-functional requirements.
- **User Acceptance Testing (UAT)**: Gathering user feedback to confirm the system fulfills end-user needs and expectations.

Test Cases: Test cases are predefined scenarios developed to validate that the system behaves as expected under specific conditions. Each test case includes a set of inputs, execution steps, expected outputs, and pass/fail criteria. These test cases are designed to ensure that each functional and non-functional requirement of the SolveXpert system is properly verified.

Test cases will cover core functionalities such as:

- User authentication (registration & login)
- Document upload and OCR processing
- Query handling through Gemini API
- Embedding creation with ChromaDB
- Text-to-speech response via avatar
- Copyable text output and chat history tracking

Table 5-2 Verification of User Registration

Test ID	**TC_1**
Test name	Registration
Date of test	October 15, 2024
Name of application	AI SolveXpert
Description	User will enter valid registration details to create an account.
Input	Fill all the fields username, email, password.
Expected output	Verification email should be sent to user entered email account and after verification user will be successfully

	registered.
Actual output	Verification email sent to user entered email account and after verification user is successfully registered
Test Role (Actor)	User
Test verified by	XYZ

Table 5-3 Verification of seller login

Test ID	**TC_2**
Test name	Login
Date of test	October 15, 2024
Name of application	SolveXpert
Description	User will enter the username and password to login into the "SolveXpert"
Input	Correct email and password
Expected output	Successful login
Actual output	Successful login
Test Role (Actor)	User
Test verified by	XYZ

Table 5-4 Verification of Document Upload

Test ID	**TC_3**
Test name	Document Upload
Date of test	October 15, 2024
Name of application	SolveXpert
Description	User uploads a document (PDF, DOCX, TXT) for question-answer generation.
Input	Valid document file
Expected	Document uploaded successfully and ready for

output	embedding and question-answering
Actual output	Document uploaded and processed successfully
Test Role (Actor)	User
Test verified by	XYZ

Table 5-5 Verification of QA via Chatbot

Test ID	**TC_4**
Test name	Question Answering via Chatbot
Date of test	October 15, 2024
Name of application	SolveXpert
Description	User asks a question related to the uploaded document and receives an answer from the chatbot.
Input	User's question (e.g., "What is the main topic of the document?")
Expected output	Relevant answer from the document processed by the chatbot.
Actual output	Chatbot responds with a correct and relevant answer based on the document content.
Test Role (Actor)	User
Test verified by	XYZ

Table 5-6 Verification of Avatar Response

Test ID	**TC_5**
Test name	Avatar Response
Date of test	October 15, 2024
Name of application	SolveXpert
Description	Avatar reads out the answer to the user's question and provides an option to mute the voice.
Input	User's question (e.g., "Explain the solution to this problem.")
Expected output	Avatar reads the answer aloud, and the user is given the option to mute the avatar if desired.
Actual output	Avatar correctly reads the response aloud, and the mute option works as expected.
Test Role (Actor)	User
Test verified by	XYZ

Table 5-7 Verification of Chatbot

Test ID	**TC_7**
Test name	Chatbot
Date of test	October 15, 2024
Name of application	AI SolveXpert
Description	Verify chatbot assistance functionality.
Input	User interacts with chatbot and initiate chat
Expected output	Chatbot provides relevant assistance or information.
Actual output	Assistance provided.
Test Role (Actor)	User
Test verified by	XYZ

6 USER MANUAL

This chapter serves as a guide for users interacting with the SolveXpert web application. It provides step-by-step instructions, along with relevant screenshots, to help users navigate and utilize all the platform's features effectively. The manual outlines how to perform core functions such as registration, login, document upload, question asking, and receiving AI-generated answers with voice responses. It also describes system requirements, supported file formats, and usage tips to enhance the user experience. The goal is to ensure users can efficiently use the application without external assistance.

6.1 Introduction

This document is designed to help users understand the functionalities and operations of the SolveXpert web application. It serves as a comprehensive guide containing step-by-step instructions, usage standards, and helpful tips. The purpose is to ensure that users can easily interact with the platform, perform tasks like uploading documents, receiving AI-generated answers, and using voice-based features, all while maximizing the system's capabilities efficiently.

6.2 Hardware/Software Requirements for the System

6.2.1 Hardware Requirements

- Desktop or laptop PC
- 5th generation Intel Core i5 processor or equivalent (or higher)
- Minimum 8 GB RAM for smooth processing
- Minimum 1 GB free disk space

6.2.2 Software Requirements

- Operating System: Windows 8 or above (Windows 10 recommended)
- Web Browser: Latest version of Google Chrome (preferred), Firefox, or Microsoft Edge

6.2.3 Other requirements

- Stable high-speed internet connection (for AI model responses and avatar output)
- Sufficient storage space to download exported answers or documents
- Audio output device (speaker/headphones) to hear the avatar's voice response

6.3 Installation guide for Application

To access and use **SolveXpert**, follow these steps:

1. **Visit the Website**: Open a web browser (Google Chrome is recommended) and go to the official **SolveXpert** website.
2. **Create an Account**: Click on the **Signup** button. Enter your username, email address, and password. Submit the form to create your account.
3. **Email Verification**: A verification email will be sent to your registered email address. Click the verification link to activate your SolveXpert account.
4. **Login to SolveXpert**: Once your account is verified, return to the login page and sign in using your email and password.
5. **Start Using SolveXpert**: After logging in, you can:

    - Upload documents for OCR and Q&A
    - Interact with the chatbot powered by Gemini API
    - View and manage chat history
    - Receive audio responses via avatar
    - Copy answers or export content as needed

6. Note: No software installation is required. SolveXpert is a browser-based platform.

6.4 Operating Manual

This section provides a complete guide on how to use the **SolveXpert** system effectively. The platform includes various functionalities aimed at solving academic queries using document-based Q&A, AI interaction, and audio-visual response systems. Below are the main features and how to use them:

6.4.1 User Registration & Login

New users must register using their email and password. After verifying the email, users can log in to access the dashboard.

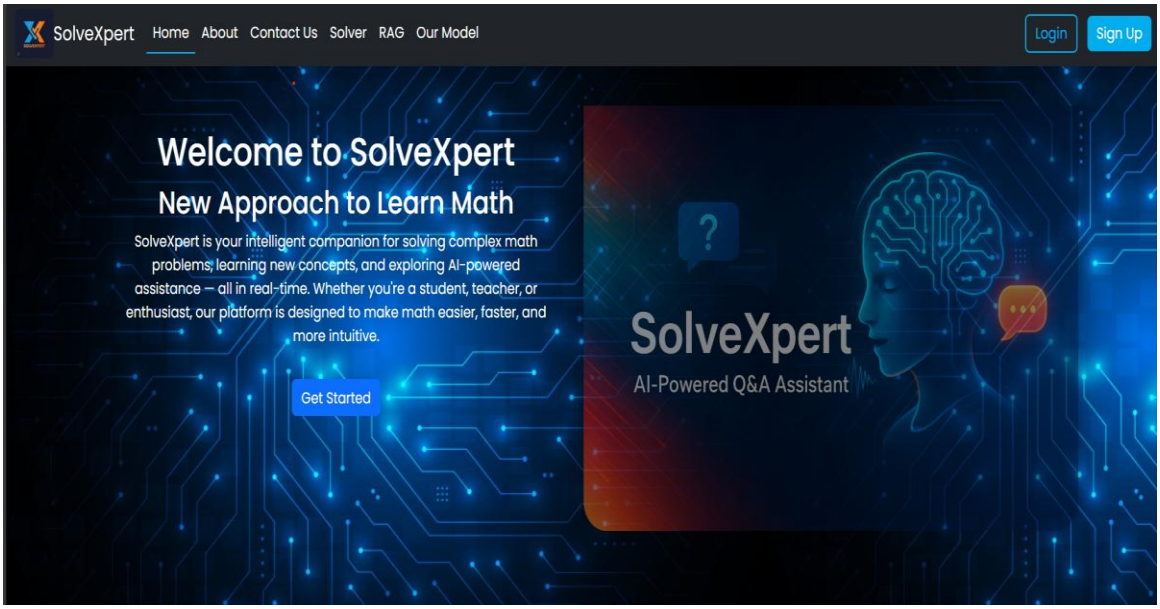

Figure 6.4-1 Home page

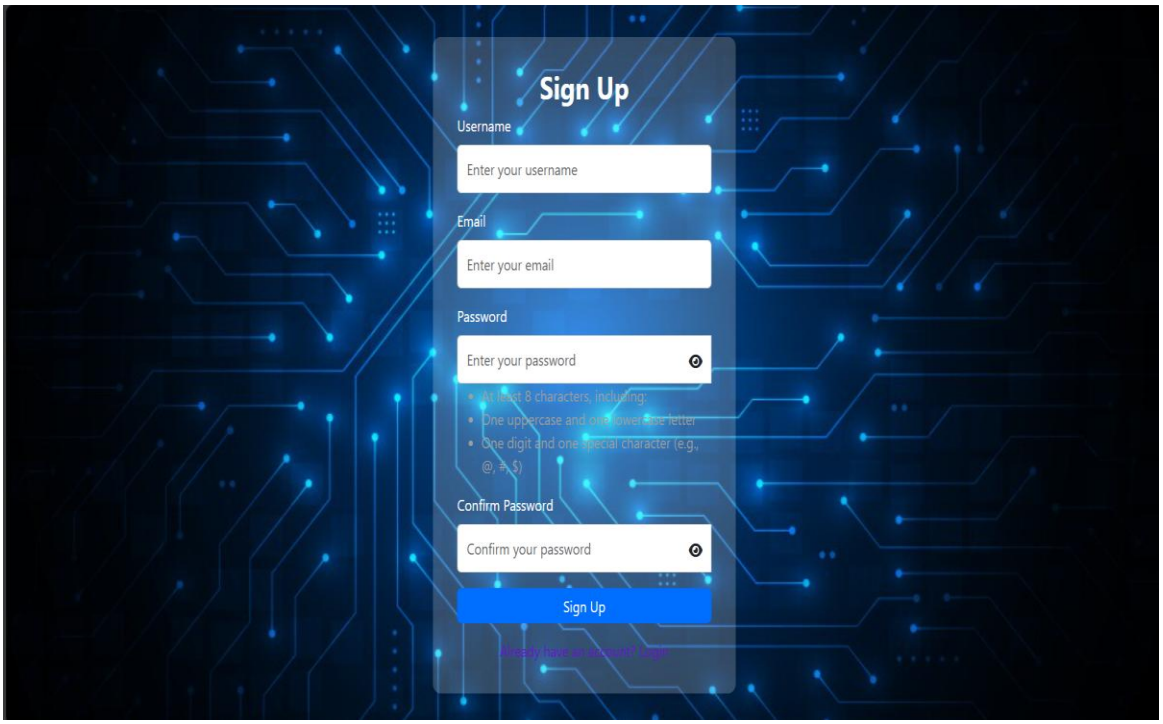

Figure 6.4-2 Signup Page

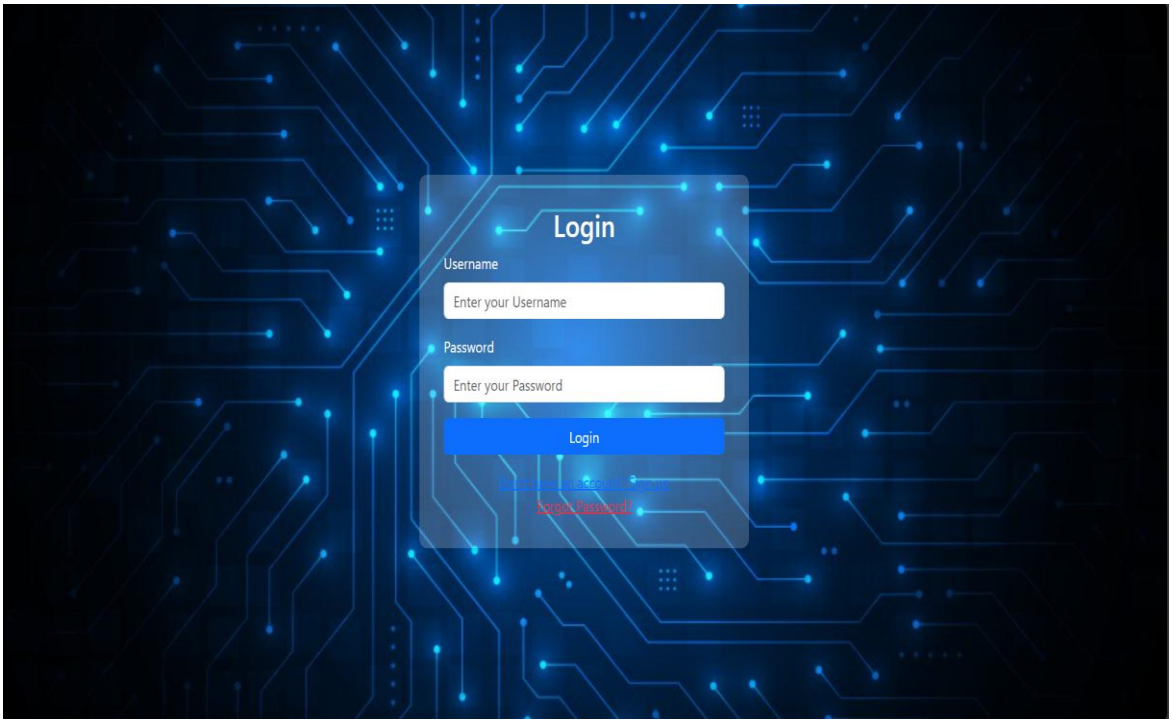

Figure 6.4-3 Login page

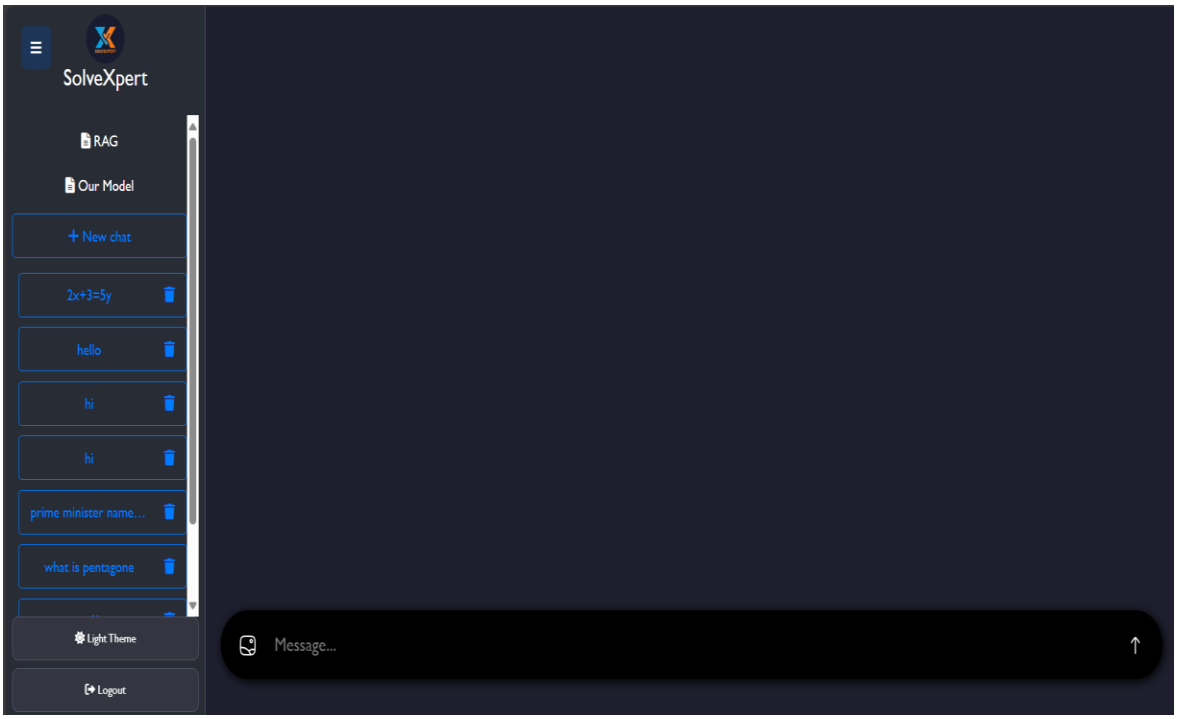

Figure 6.4-4 Redirected to Solver page

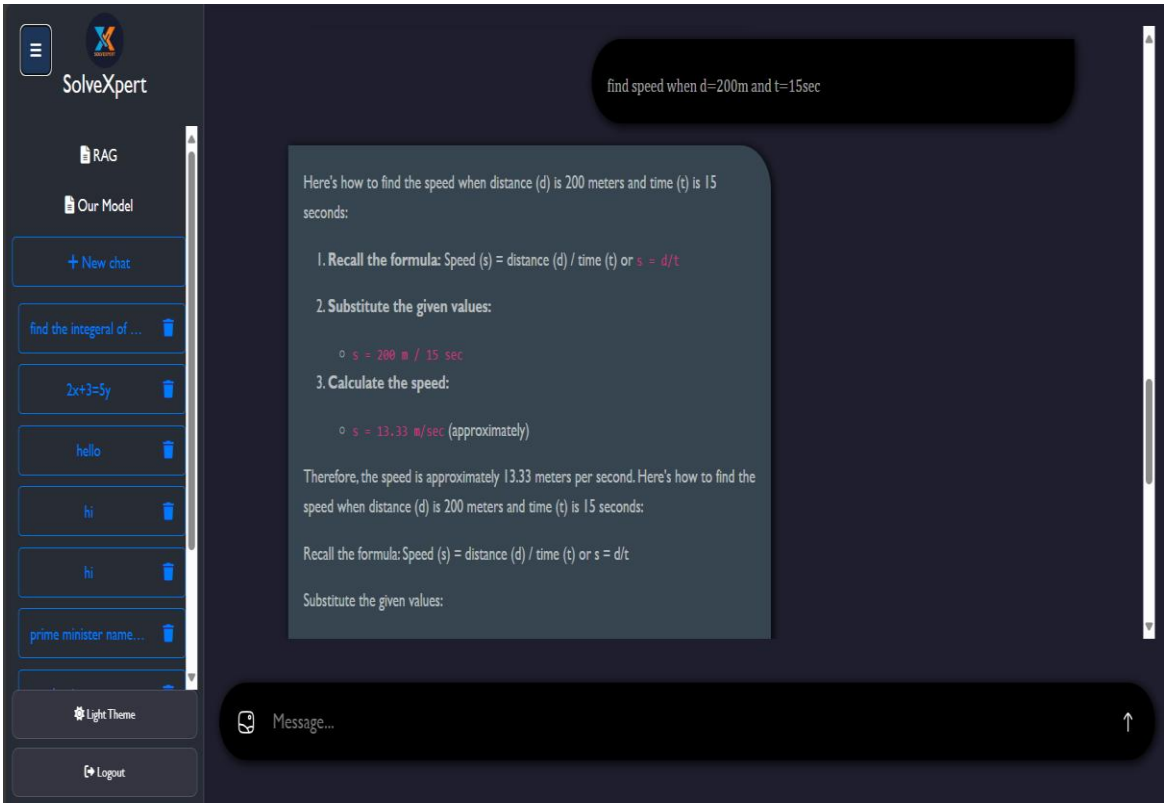

Figure 6.4-5 Response from Solver

6.4.2 RAG Page

The image below is the RAG page.

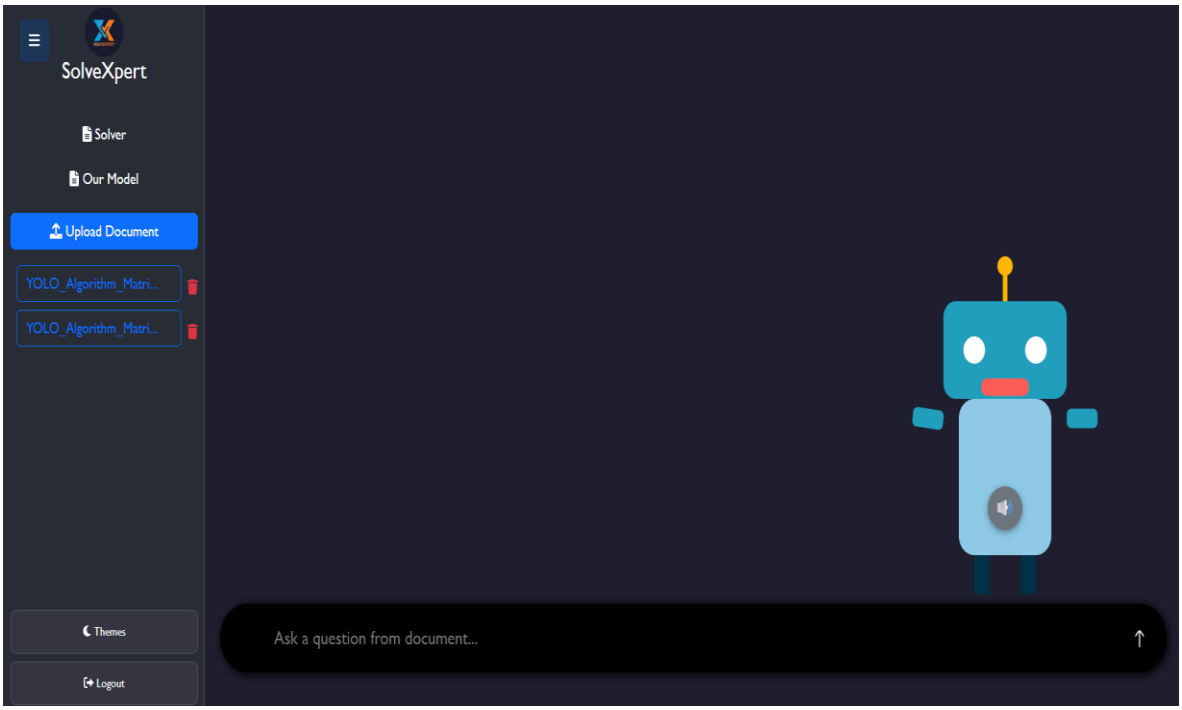

Figure 6.4-6 RAG Page

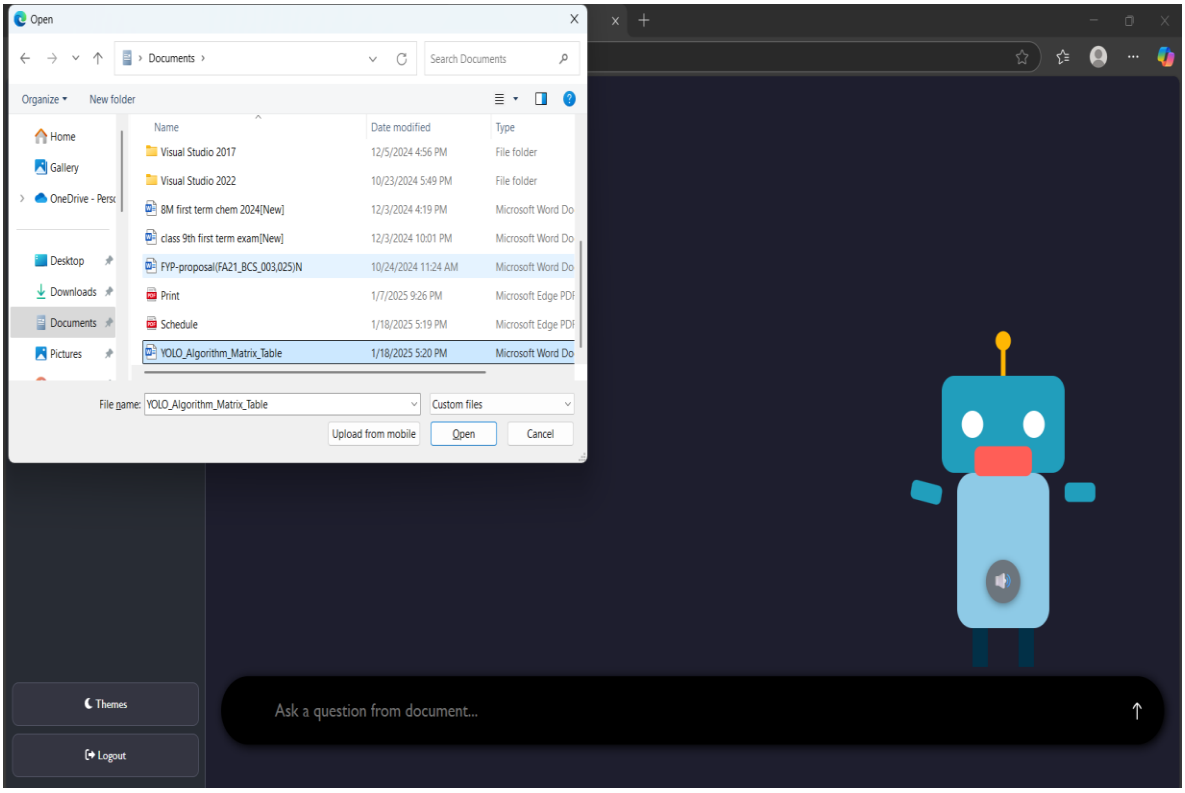

Figure 6.4-7 Upload Document

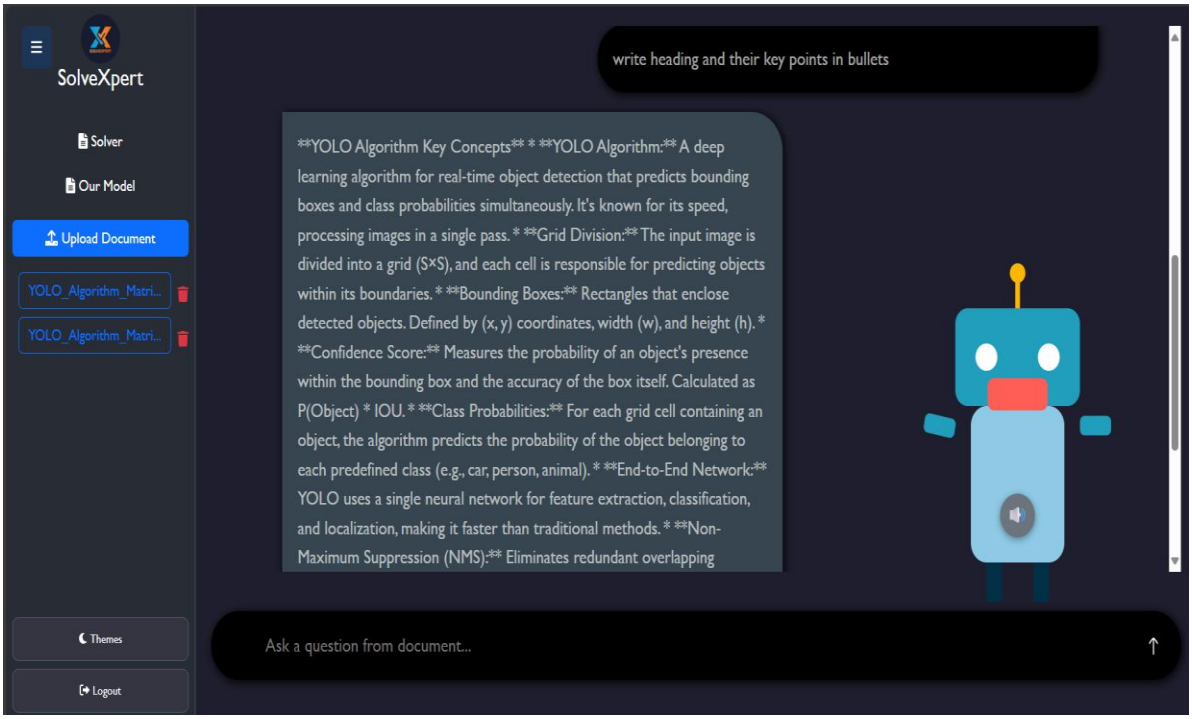

Figure 6.4-8 Response from RAG

7 GLOSSARY

Table 7-1 Glossary of Terms and Abbreviations

Abbreviation	Stands For
FR	Functional Requirement
TC	Test Case
USR-FR	User Functional Requirement
RTM	Requirement Traceability Matrix
UI	User Interface
TTS	Text-to-Speech
OCR	Optical Character Recognition
API	Application Programming Interface
DB	Database
QA	Quality Assurance
ERD	Entity Relationship Diagram
OTP	One-Time Password