

SHRI SHAMBHUBHAI V. PATEL COLLEGE OF
COMPUTER SCIENCE & BUSINESS MANAGEMENT
Near I. C. Gandhi School, Sumul Dairy Road, Surat. (Gujarat) India.



CERTIFICATE

This is to certify that the minor project entitled _____

AI-Chatbot

has been carried out by Mr./Ms. Yash Dayani P.

_____ in Bachelor of Computer Application, Semester V

Exam No. 3925

towards partial fulfillment of the course, for the Academic Year 2024-25.

Mentor/Guide
Date:



I/C. Principal
Date:

MINOR PROJECT OF B.C.A.

Academic Year: 2024 - 25

Approved by: _____

Examiners



INDEX

SR.NO	PARTICULAR	PAGE NO
1.	Introduction	1
	1.1 Project Profile	-
	1.2 Objectives	-
	1.3 Scope and Limitations	3
2.	System Environment	4
	2.1 Hardware Requirements	-
	2.2 Software Requirements	-
	2.3 Development Tools	5
3.	Problem Specification	7
	3.1 Introduction	-
	3.2 Problem Statement	-
	3.3 Objectives and Purpose	-
	3.4 Functional and Non-Functional Requirements	8
4.	Feasibility Study	11
	4.1 Technical Feasibility	-
	4.2 Operational Feasibility	-
	4.3 Economic Feasibility	12
	4.4 Legal and Ethical Considerations	13
5.	Risk Identification and Management	14
	5.1 Risk Factors	15
	5.2 Risk Assessment Techniques	-
	5.3 Risk Mitigation Strategies	16
	5.4 Security and Compliance Considerations	-
6.	System Planning and Development	18
	6.1 Requirement Gathering and Analysis	-
	6.2 Timeline Chart	-

7.	System Analysis	20
	7.1 Database Design	-
	7.2 ER Diagram	21
	7.3 Data Flow Diagram	22
	7.4 Use Case Diagram	24
	7.5 Data Dictionary	-
8.	Designing	26
	8.1 Design Principles	-
	8.2 User Interface	27
9.	Testing	31
	9.1 Software Testing	-
	9.2 Unit Testing	-
	9.3 System Testing	-
10.	References	32

1. Introduction

1.1 Project Profile

Project Title: AI Chat Bot

Project Overview:

This project is a web-based platform that allows users to submit and save personalized queries and responses through a chat interface. It manages user authentication and tracks search history, featuring user registration and login functionalities. Built with modern web technologies, the platform adheres to best practices for security and performance optimization.

Key Features:

- **User Authentication:** Secure registration and login processes with hashed passwords and JWT-based token management.
- **Search History Management:** Allows users to save, view, and delete their search history.
- **Responsive Design:** A user-friendly interface that adapts to different devices and screen sizes.
- **Dark/Light Theme Toggle:** Enhances user experience by allowing them to switch between dark and light modes.

Technologies Used:

- **Frontend:** React.js for building interactive user interfaces, with state management and routing handled within the application.
- **Backend:** Node.js with Express for server-side logic and API management.
- **Database:** MongoDB for data storage, utilizing Mongoose for object data modeling.
- **Authentication:** JWT (JSON Web Tokens) for secure token-based authentication.

1.2 Objectives

The primary objective of this project is to build an AI-powered chatbot that can interact with users effectively and provide relevant responses using machine learning models and APIs. The specific goals include:

- To create a conversational AI system that can process natural language inputs and generate meaningful responses.
- To develop a secure and robust user management system, allowing users to register, authenticate, and manage their chat history.
- To implement a responsive, easy-to-use interface that improves the overall user experience across various devices.

- To leverage modern technologies such as React, Node.js, and MongoDB for creating a scalable and performant application.
- To ensure data privacy and security by implementing industry-standard authentication and data protection measures.

1.3 Scope and Limitations

Scope:

- The project will include the development of both frontend and backend components to support user interaction with the chatbot.
- The chatbot will process user inputs and provide responses using the Gemini API, which offers AI-generated replies.
- The system will store and manage user-specific data, including authentication details and search history.
- Basic data analytics, such as tracking frequently asked questions, will be integrated into the platform for further analysis.

Limitations:

- The chatbot's responses are limited to the capabilities of the Gemini API and cannot answer questions outside the scope of its dataset.
- API limitations such as rate-limiting may affect the system's ability to provide continuous responses if usage quotas are exceeded.
- Since the system relies on third-party APIs, downtime or performance issues from the external API provider could impact the functionality of the chatbot.
- Multi-language support is not included in the initial version of the project but may be considered in future updates.
- Advanced natural language understanding (NLP) features like sentiment analysis, deep context understanding, and follow-up conversations are not implemented.

2. System Environment

This section outlines the hardware, software, and tools needed for the successful development and deployment of the AI chatbot system.

2.1 Hardware Requirements

The system requires the following hardware components for both development and production environments:

- Development Environment:
 - Processor: Minimum Intel i5 (8th generation or higher) or equivalent AMD processor.
 - RAM: 8 GB (16 GB recommended for optimal performance during development).
 - Storage: 100 GB of free disk space (SSD recommended for faster build times).
 - Network: Stable internet connection (minimum 10 Mbps) for API integration and testing.
 - Graphics: Integrated or dedicated GPU (optional) for testing graphics and UI performance.
- Production/Deployment Environment:
 - Processor: Multi-core server-grade processor (e.g., Intel Xeon or AMD EPYC).
 - RAM: 16 GB or more (scalable based on traffic).
 - Storage: Minimum of 200 GB (scalable storage, preferably SSD).
 - Network: High-speed internet connection with low latency (minimum 100 Mbps).
 - Server Type: Cloud server (AWS, Azure, Google Cloud) or physical server with containerization (e.g., Docker).

2.2 Software Requirements

The following software components are essential for the development and deployment of the AI chatbot system:

- Frontend:
 - React.js: JavaScript library for building user interfaces.
 - HTML5/CSS3: For structuring and styling the user interface.

- Node.js (v14 or higher): Required to run JavaScript outside the browser for development and building the React app.
 - Browser Compatibility: Chrome, Firefox, Safari, and Edge (latest versions) to ensure cross-browser compatibility.
- Backend:
 - Node.js (v14 or higher): Server-side platform to handle requests, API calls, and business logic.
 - Express.js: Web framework for building APIs and handling HTTP requests/responses.
 - MongoDB (v4.4 or higher): NoSQL database for storing user credentials and chat history.
 - JWT (JSON Web Tokens): For user authentication and session management.
- External Services:
 - Gemini API: AI service for generating chatbot responses.
 - MongoDB Atlas: for storing and managing large database
- Security:
 - HTTPS/SSL: Secure data transmission between clients and servers.
 - OAuth 2.0 / OpenID Connect: For user authentication, if integrating third-party services.
 - Firewall/Access Control: For securing the production environment from unauthorized access.

2.3 Development Tools

The development of the AI chatbot system relies on the following tools for coding, testing, and deployment:

- IDE/Code Editors:
 - Visual Studio Code: Popular lightweight editor with extensions for JavaScript, Node.js, and React.
 - IDX: Project IDX gets you into your dev workflow in no time, backed by the security and scalability of Google Cloud.
- Version Control:
 - Git: Distributed version control system for tracking changes and collaboration.

- GitHub/GitLab: Repository hosting platforms for managing source code and continuous integration.
- Testing Tools:
 - Google AI Studio: Gemini Api testing for better response.
 - Postman: API development and testing tool for testing backend endpoints.
- Package Management:
 - npm (Node Package Manager): Used to manage project dependencies for both frontend (React.js) and backend (Node.js).
- Build and Deployment:
 - Vite: Vite uses native ES modules in the browser to serve code during development, which allows for extremely fast server start times.
- Database Management:
 - MongoDB Atlas: Cloud-based MongoDB service for database hosting and scaling.

3. Problem Specification

This section outlines the problem the AI chatbot system is designed to solve, its objectives, and the system's functional and non-functional requirements.

3.1 Introduction

With the increasing demand for automated systems that can provide customer support, answer user queries, and improve engagement, AI chatbots have become an essential tool in various industries. However, creating an effective chatbot requires a system that can handle natural language queries, generate accurate responses, and manage user interaction data. This project aims to design and develop an AI-powered chatbot that can efficiently interact with users, store their queries, and provide meaningful responses using AI technologies.

3.2 Problem Statement

Modern businesses and applications often require round-the-clock customer interaction, but maintaining human operators for such purposes is expensive and inefficient. Additionally, the available customer support chatbots are often rigid, lack context awareness, and do not adequately handle complex queries. There is a need for an AI chatbot that can:

- Provide accurate, context-aware responses.
- Offer a user-friendly interface.
- Securely manage user authentication and store historical interactions for future reference.
- Scale with user demands and API requests.

Key issues:

- Existing chatbots lack natural language understanding capabilities.
- Poor user experience due to limited interactivity and personalization.
- Limited management of historical conversations for future queries and interactions.
- Concerns regarding security and data privacy for sensitive user information.

3.3 Objectives and Purpose

The primary objective of this project is to develop an AI-powered chatbot that offers seamless, interactive experiences to users by answering their queries through AI-driven responses. The chatbot should be capable of learning from past interactions to improve accuracy and relevance.

Objectives:

- Develop a conversational AI system: Build a chatbot capable of processing and responding to natural language queries using the Gemini API.
- Implement secure user authentication: Ensure that user data is protected through secure registration and login mechanisms using JWT (JSON Web Tokens).
- Provide history management features: Allow users to view, manage, and delete their chat history, offering personalized experiences based on previous interactions.
- Enhance user experience with a responsive design: The chatbot must be responsive and functional across various devices and screen sizes.
- Ensure scalability and performance: The system should be able to handle a growing number of users and API requests without sacrificing performance.

Purpose:

- To provide an AI chatbot that can act as a virtual assistant, answering user queries, assisting customers, and automating various interactions.
- To improve efficiency by reducing the need for human intervention in routine queries.
- To offer a secure, scalable solution for businesses and applications that need interactive AI-driven customer support.

3.4 Functional and Non-Functional Requirements

The system's requirements are divided into functional and non-functional categories to outline the expected behavior and performance of the chatbot.

Functional Requirements:

1. User Registration and Login:
 - The system must allow users to register by providing a username, email, and password.
 - It must allow users to log in with their credentials and generate a secure token (JWT).
 - Passwords must be stored securely using hashing algorithms.
2. Chatbot Interaction:
 - The system must allow users to interact with the chatbot by submitting text queries.
 - It must provide real-time responses based on the input by leveraging the Gemini API.
 - The system must store the query and response in the user's search history for future reference.
3. Search History Management:

- The system must allow users to view, delete, and manage their previous chat interactions.
- The history must be user-specific, and only accessible by authenticated users.

4. AI-Generated Responses:

- The chatbot must generate responses using the Gemini API, providing accurate, AI-driven replies to user queries.
- The system must handle API requests efficiently and show meaningful error messages if the API is unavailable.

5. Dark/Light Mode Toggle:

- The system must allow users to switch between dark and light themes for better user experience across different environments.

Non-Functional Requirements:

1. Security:

- The system must implement HTTPS for all communications to protect data in transit.
- JWT tokens must be used for user authentication and must expire after a set period for added security.
- The system must prevent unauthorized access to user data, including search history and personal information.

2. Performance:

- The system must be able to handle at least 1000 concurrent users without significant degradation in performance.
- Response times for chatbot queries must be less than 2 seconds under normal load.
- The system must scale horizontally to accommodate a growing user base by deploying on cloud infrastructure.

3. Reliability:

- The system must have an uptime of 99.9% to ensure continuous availability for users.
- Error handling mechanisms must be implemented to gracefully handle API timeouts, user input errors, or database failures.

4. Usability:

- The system must have an intuitive, user-friendly interface that requires minimal training or onboarding for new users.
- It should be accessible from both desktop and mobile devices, with all functionality available on both platforms.

5. Scalability:

- The system must be designed to handle a growing number of users and increased interaction frequency without major architectural changes.
- The system should support easy integration with other AI services and external APIs as required in future developments.

6. Maintainability:

- The codebase must follow best practices, with modular design and proper documentation to allow easy maintenance and updates.
- Automated testing should be incorporated to ensure the system remains stable during development and post-deployment changes.

4. Feasibility Study

The feasibility study evaluates whether the AI chatbot system can be successfully implemented, considering technical, operational, economic, legal, and ethical factors.

4.1 Technical Feasibility

Objective: To determine if the technology required to develop and deploy the AI chatbot is available, reliable, and scalable.

Assessment:

- **Technological Maturity:** The technologies planned for the AI chatbot (React.js, Node.js, MongoDB, and Gemini API) are well-established, widely supported, and have large developer communities. These tools have been successfully used in a variety of similar web-based applications, ensuring a robust technical foundation.
- **Infrastructure:** The system is designed to be deployed on cloud platforms such as AWS, Google Cloud, or Azure, all of which offer scalable infrastructure capable of handling high traffic and multiple concurrent users.
- **APIs:** The Gemini API, used for generating AI responses, provides extensive capabilities in natural language processing (NLP) and machine learning. However, API rate limits and response times will need to be carefully managed to ensure system performance and reliability.
- **Scalability:** The use of cloud-based infrastructure with containerization (e.g., Docker) ensures the system can scale horizontally to accommodate an increasing number of users without significant architectural changes.
- **Development Tools:** The development tools (Git, Visual Studio Code, Postman) are widely used and provide support for agile, rapid development cycles. Integration with CI/CD pipelines ensures continuous testing, builds, and deployment, reducing risks during development.

Conclusion: The project is technically feasible with the available tools, technologies, and infrastructure, ensuring smooth development and deployment.

4.2 Operational Feasibility

Objective: To evaluate whether the system can operate successfully in a real-world environment and whether users and stakeholders will be able to use it effectively.

Assessment:

- **Ease of Use:** The system is designed with a user-friendly interface that is intuitive and easy to navigate. The use of responsive design ensures that users on both desktop and mobile devices can access and use the chatbot without issues.

- **Training and Support:** Minimal training is required for end-users to interact with the chatbot. Most interactions are self-explanatory, and any issues that arise can be addressed through in-app guidance or a help section. Support teams may need to handle user registration issues or API-related problems.
- **User Engagement:** The AI chatbot is expected to enhance user engagement by providing quick, accurate, and relevant responses to user queries. The history management feature allows users to track previous conversations, making it easier to manage ongoing interactions.
- **Maintenance:** The system will require regular monitoring and maintenance, including updates to the underlying machine learning models (Gemini API), security patches, and performance optimization. An automated CI/CD pipeline ensures that updates can be deployed efficiently with minimal downtime.

Conclusion: The system is operationally feasible, requiring minimal user training and offering significant benefits to users by improving engagement and reducing manual intervention.

4.3 Economic Feasibility

Objective: To assess whether the project is economically viable and can deliver value that justifies the cost of development, deployment, and maintenance.

Assessment:

- **Development Costs:** The development phase will require costs related to developer salaries, tools, and cloud infrastructure. The technologies used (React.js, Node.js, MongoDB) are open-source, minimizing licensing costs. Hosting the development environment on cloud services (AWS, Google Cloud, or Azure) provides flexibility and scalability, but also introduces variable costs based on usage.
- **API Costs:** The Gemini API will likely have a cost associated with usage, especially if the project scales to a large number of users. Proper API management strategies (e.g., caching responses, limiting API calls per user) will be required to control costs.
- **Deployment and Maintenance Costs:** Deploying the system on cloud platforms will incur operational costs, which include storage, bandwidth, and server management. These costs can be managed by scaling resources up or down based on demand. Regular maintenance, security patches, and feature updates will require ongoing developer involvement, which must be factored into long-term operational costs.
- **Return on Investment (ROI):** The system will provide value by automating responses, reducing the need for human intervention, and enhancing user engagement, which can lead to cost savings for businesses. By reducing customer service overhead and improving user satisfaction, the chatbot can provide a high return on investment over time.

Conclusion: The project is economically feasible, with manageable development and operational costs, especially considering the use of open-source technologies. The expected benefits and ROI outweigh the costs, especially in the long term.

4.4 Legal and Ethical Considerations

Objective: To ensure the system complies with relevant legal regulations and addresses ethical concerns related to AI usage, data privacy, and security.

Assessment:

- **Data Privacy:** The system will collect and store user data, including login credentials and chat history. It is crucial to comply with privacy regulations such as GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act). Users must be informed about how their data is collected, used, and stored. The system must allow users to request data deletion or modification.
- **Security:** The system must ensure that all data transmitted between the user and the server is encrypted using HTTPS. Sensitive information, such as passwords, must be hashed and stored securely. Additionally, JWT tokens should be properly implemented to manage user sessions and prevent unauthorized access.
- **AI Transparency:** The chatbot must be transparent about the fact that users are interacting with an AI system. Ethical concerns may arise if users are misled into believing they are speaking with a human. Proper disclaimers must be provided to avoid confusion.
- **Bias in AI:** The Gemini API, or any AI system used, may inherit biases from the data it is trained on. To minimize bias in chatbot responses, regular audits and updates to the AI model must be conducted. Ethical use of AI includes ensuring that the chatbot does not perpetuate harmful stereotypes or misinformation.
- **Intellectual Property:** All third-party APIs and open-source tools used in the project must comply with licensing agreements. Proper attribution and usage rights should be respected to avoid intellectual property violations.

Conclusion: The project is legally and ethically feasible, provided that it adheres to data privacy regulations, implements strong security measures, ensures AI transparency, and addresses potential biases in AI-generated responses.

5. Risk Identification and Management

Risk management is a critical aspect of ensuring the success of the AI chatbot project. This section identifies potential risks, assesses their impact, and provides strategies to mitigate them.

5.1 Risk Factors

The following are the key risk factors associated with the development and deployment of the AI chatbot system:

1. API Limitations:

- The chatbot relies on third-party APIs (e.g., Gemini API) for generating AI-driven responses. Limitations such as rate-limiting, downtime, or poor response times can negatively impact user experience.

2. Security Vulnerabilities:

- As the system handles sensitive user data (authentication details, search history), it is vulnerable to data breaches, unauthorized access, or other security threats, especially if proper security measures are not implemented.

3. Performance Issues:

- Scalability and response time are critical for chatbot systems. High traffic volumes or inefficient API usage may result in slow response times or system downtime, which can lead to user dissatisfaction.

4. Data Privacy Concerns:

- Storing user chat history and credentials introduces risks related to data privacy, especially in jurisdictions with stringent privacy regulations like GDPR or CCPA. Failure to comply with these regulations could result in legal consequences.

5. AI Bias and Ethics:

- The AI used for generating responses may be biased based on the training data. If left unchecked, it could lead to the generation of biased, inappropriate, or unethical responses, which can harm user trust and the system's reputation.

6. Technical Debt:

- Rapid development cycles could lead to insufficient documentation, incomplete testing, or rushed implementations, introducing technical debt. This can make future updates and maintenance difficult, reducing system stability over time.

5.2 Risk Assessment Techniques

The following techniques are used to assess the likelihood and impact of the identified risks:

1. Risk Probability and Impact Matrix:

- Each risk is evaluated based on its likelihood of occurring and the potential impact it could have on the project. Risks are classified into high, medium, or low categories based on their likelihood and severity.
 - High Risk: Issues that are likely to occur and would have a significant impact (e.g., security breaches).
 - Medium Risk: Risks that are less likely to occur but could still have considerable effects (e.g., API rate-limiting).
 - Low Risk: Unlikely events that would have a minor impact (e.g., minor performance delays under high load).

2. SWOT Analysis (Strengths, Weaknesses, Opportunities, Threats):

- This method analyzes the internal strengths and weaknesses of the system and identifies external opportunities and threats that may impact the chatbot's development and operation.

3. Failure Mode and Effects Analysis (FMEA):

- This method identifies possible failures in each component of the system (e.g., API, database, authentication) and assesses the likelihood of failure and its consequences. FMEA assigns a Risk Priority Number (RPN) to help prioritize mitigation efforts.

4. Risk Breakdown Structure (RBS):

- The RBS categorizes risks into groups (technical, operational, financial, etc.) to organize and evaluate risks systematically. This allows for a structured approach to managing risks.

5.3 Risk Mitigation Strategies

Each risk identified in section 5.1 has corresponding mitigation strategies to reduce its likelihood or impact:

1. API Limitations:

- Mitigation Strategy: Implement API rate-limiting mechanisms and cache frequent responses to minimize redundant API calls. Establish fallback responses or a local AI model to handle situations when the external API is unavailable.

2. Security Vulnerabilities:

- Mitigation Strategy: Use HTTPS for secure communication, hash passwords with modern algorithms (e.g., bcrypt), and implement JWT-based authentication for token management. Regularly update dependencies and conduct security audits to patch vulnerabilities.

3. Performance Issues:

- Mitigation Strategy: Implement horizontal scaling on cloud infrastructure (e.g., AWS, Google Cloud) to handle increased traffic. Use load balancing, caching mechanisms (e.g., Redis), and optimize database queries to improve system performance.

4. Data Privacy Concerns:

- Mitigation Strategy: Ensure compliance with GDPR and CCPA by providing clear user consent forms, implementing "Right to Erasure" (allowing users to delete their data), and securing stored user data using encryption. Conduct regular audits to ensure data privacy standards are upheld.

5. AI Bias and Ethics:

- Mitigation Strategy: Periodically audit the chatbot's responses for bias and inappropriate content. Retrain the AI model with diverse datasets to ensure unbiased responses. Implement a feedback mechanism for users to report inappropriate or biased responses.

6. Technical Debt:

- Mitigation Strategy: Adhere to coding best practices, maintain comprehensive documentation, and implement automated testing frameworks (e.g., Jest, Cypress) to ensure quality. Allocate time in the development schedule for refactoring and technical debt reduction.

5.4 Security and Compliance Considerations

Given that the AI chatbot system will handle sensitive user data, security and compliance are critical. The following considerations ensure that the system is secure and compliant with relevant regulations:

1. Data Encryption:

- All data transmitted between users and the server must be encrypted using SSL/TLS to prevent data breaches during transmission. Additionally, sensitive user data (e.g., passwords) should be stored in the database in a hashed and salted format.

2. Authentication and Authorization:

- The system must implement secure authentication protocols, such as JWT (JSON Web Tokens), for handling user sessions. This ensures that only

authenticated users can access specific features, such as viewing or deleting their chat history.

- Role-based access control (RBAC) can be introduced for administrative features, limiting sensitive operations to authorized personnel only.

3. Compliance with Data Privacy Laws:

- The system must comply with Data Privacy Laws to protect user data. Key compliance requirements include:
 - User Consent: Users must be informed about data collection practices and provide explicit consent.
 - Right to Erasure: Users must have the ability to delete their data upon request.
 - Data Breach Notification: In case of a data breach, users and authorities must be informed promptly.

4. Regular Security Audits:

- Periodic security audits should be conducted to identify vulnerabilities in the system, including weak points in authentication, data storage, and API integration. Penetration testing can also be performed to simulate cyberattacks and assess the system's resilience.

5. Access Control and Monitoring:

- The system must implement strict access controls to prevent unauthorized access to sensitive data. Multi-factor authentication (MFA) can be added to strengthen security for administrative users.
- Real-time monitoring and logging should be in place to track suspicious activity or unauthorized access attempts.

6. System Planning and Development

This section details the process of gathering requirements, planning the development timeline, selecting the appropriate development methodology, and following a structured development lifecycle for the AI chatbot project.

6.1 Requirement Gathering and Analysis

Objective: The primary objective of the requirement gathering phase is to ensure that the needs of the end-users, stakeholders, and business goals are fully understood and documented before development begins.

Process:

- **Functional Requirements:** Define the system's functional requirements, including:
 - User authentication and authorization.
 - Chat interface for interacting with the AI chatbot.
 - Integration with Gemini API for AI-generated responses.
 - Management of user chat history.
- **Non-Functional Requirements:** Document the non-functional requirements such as:
 - **Performance:** System should respond within 2 seconds for most queries.
 - **Scalability:** The system must handle at least 1,000 concurrent users.
 - **Security:** Data must be encrypted, and user sessions should be secured with JWT.
- **Feasibility Study:** Conduct a detailed feasibility study (technical, operational, and financial) to assess the project's viability.

Outcome: A comprehensive requirements document outlining both functional and non-functional requirements that will guide the design and development process.

6.2 Timeline Chart

Once the requirements are gathered and analyzed, a detailed project timeline is created. This timeline outlines key milestones and deliverables, helping to ensure the project stays on schedule.

Key Milestones:

1. Requirement Gathering and Analysis: 1 week
2. System Design: 1 weeks
 - UI/UX design
 - Database schema and architecture design

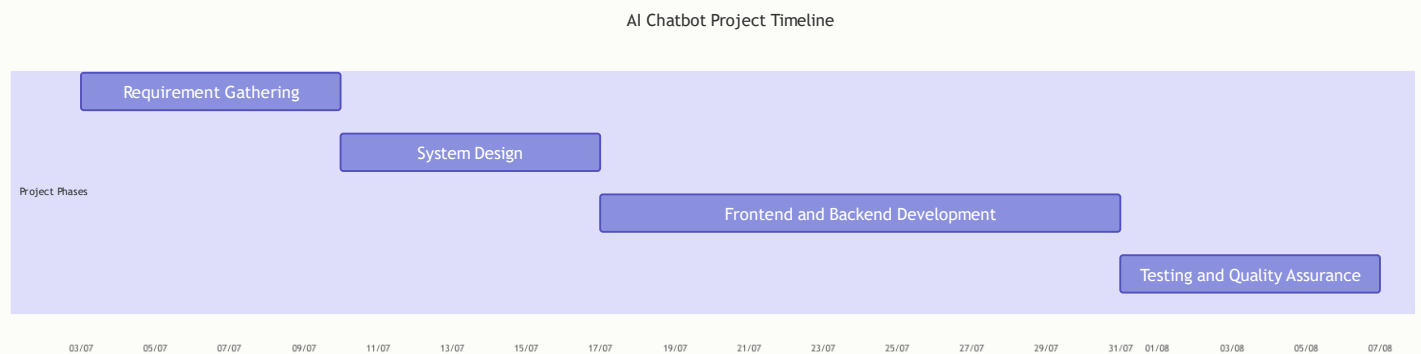
3. Frontend and Backend Development: 2 weeks

- React.js frontend development
- Node.js backend API development
- MongoDB integration

4. Testing and Quality Assurance: 1 weeks

- Unit testing
- Integration testing
- End-to-end testing

Chart



7. System Analysis

7.1 Database Design

Database Overview:

- Database Management System (DBMS): MongoDB
- Purpose: To store and manage data for user authentication and search history tracking.

User	
ObjectId	_id
string	user name
string	email
string	password

SearchHistory	
ObjectId	_id
ObjectId	user
string	prompt
string	response
date	createdAt

Schema Design:

1. User Collection:

- Schema:

```
users.db

const UserSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});
```

- Description: Stores user information including username, email, and hashed password.

2. SearchHistory Collection:

- Schema:

```
searchhistories.db

const SearchHistorySchema = new mongoose.Schema({
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  prompt: { type: String, required: true },
  response: { type: String, required: true },
  createdAt: { type: Date, default: Date.now }
});
```

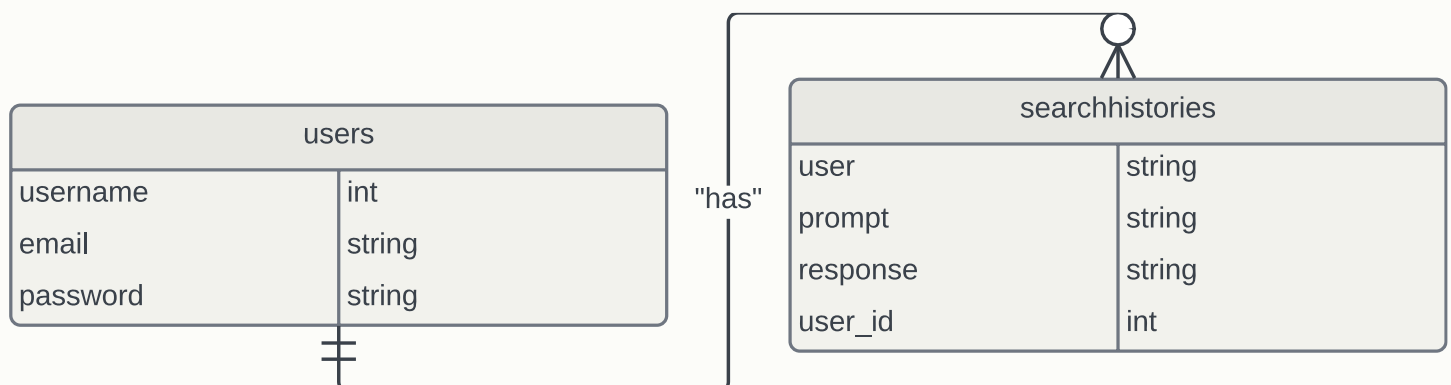
- Description: Records search prompts and responses associated with a user.

Relationships:

- User ↔ SearchHistory: One-to-many relationship where one user can have multiple search history records.

7.2 ER Diagram

Entity-Relationship Diagram:



Entities:

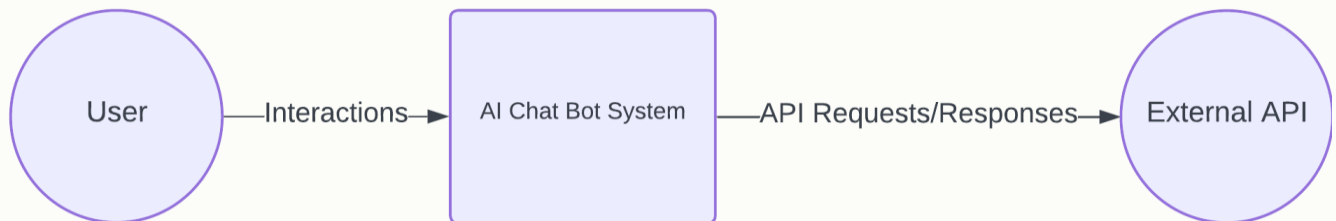
- User:
 - Attributes: username, email, password
- SearchHistory:
 - Attributes: user, prompt, response, createdAt

Relationships:

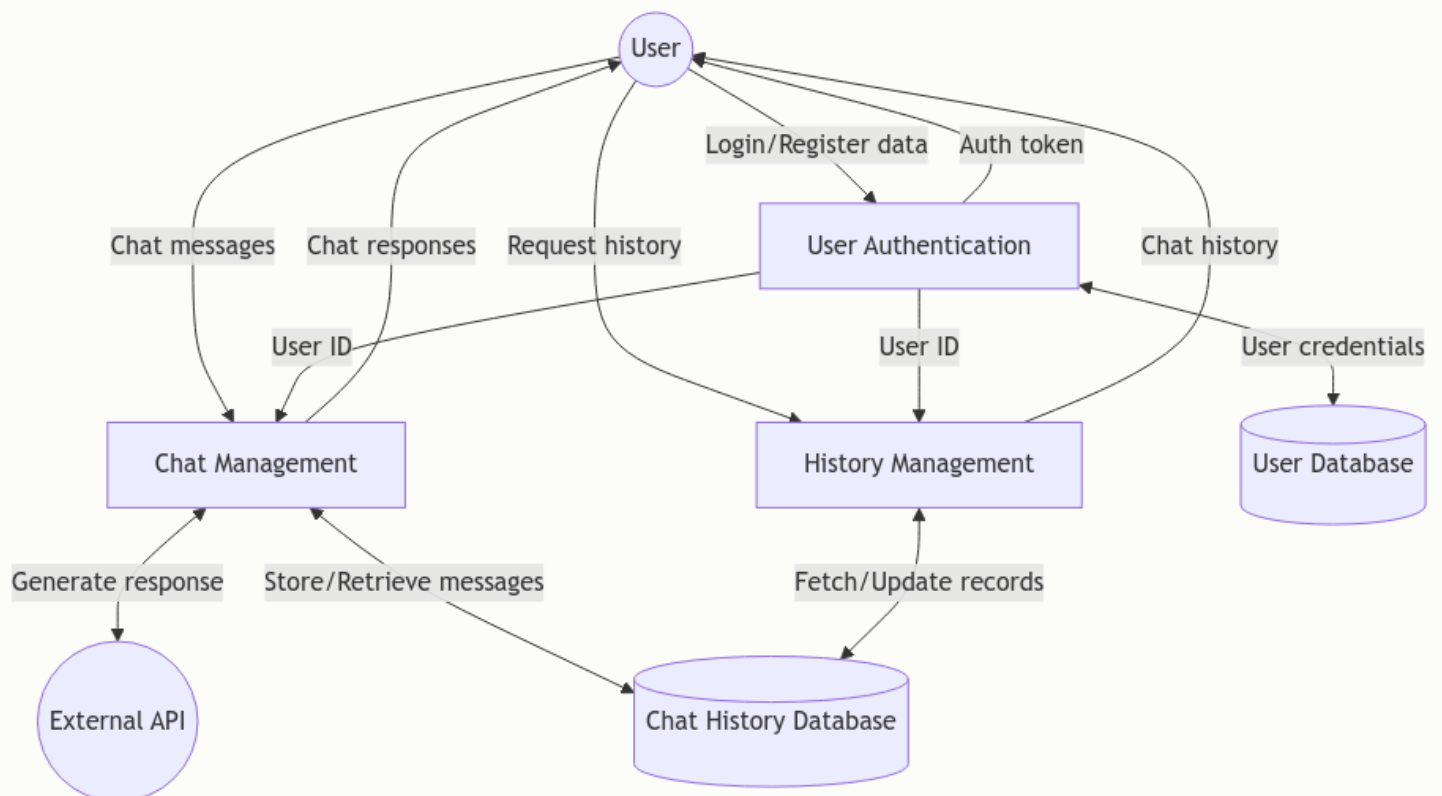
- User (1) → (M) SearchHistory
 - A user can have multiple search history records.

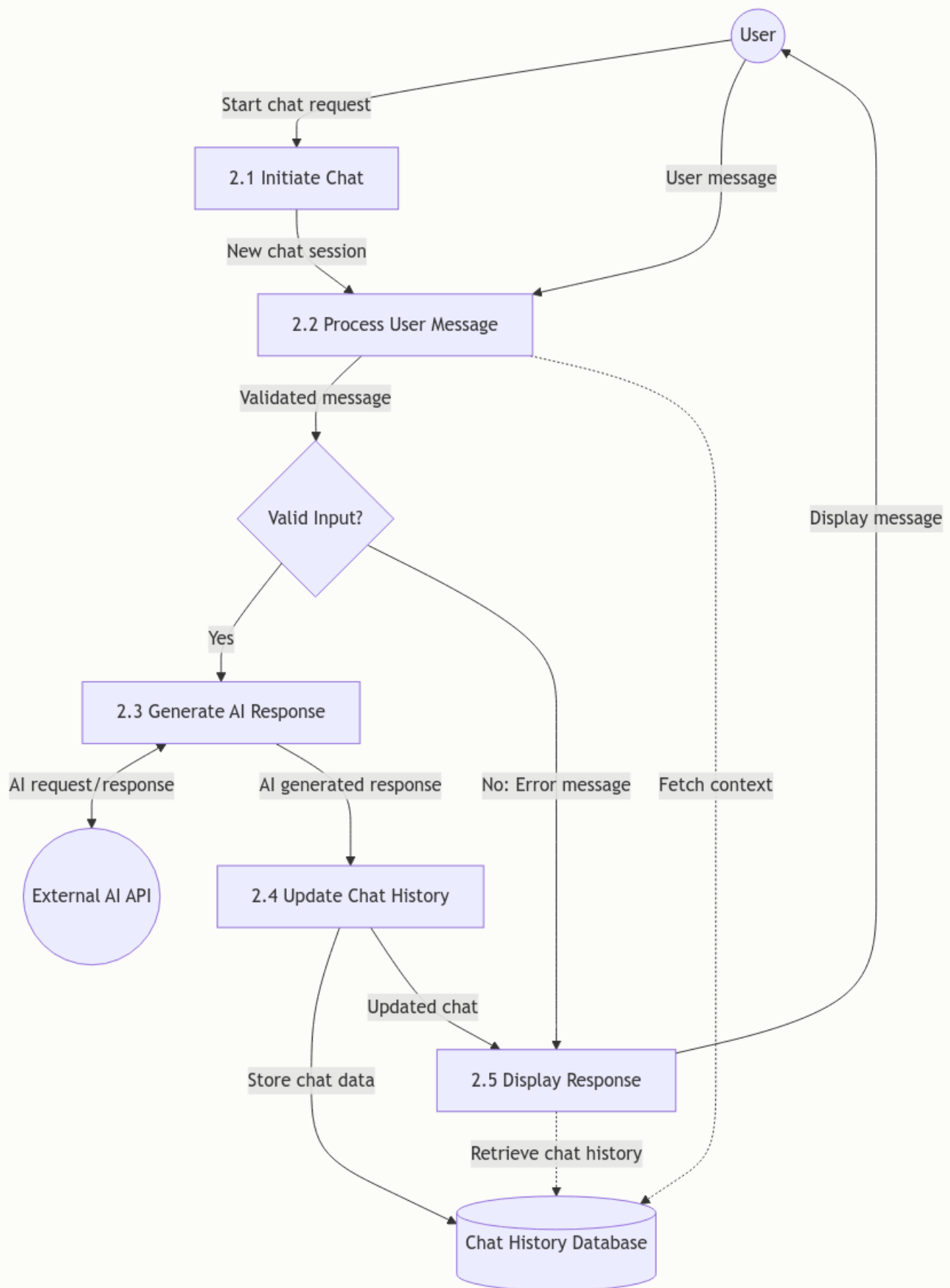
7.3 Data Flow Diagram

Level: 0

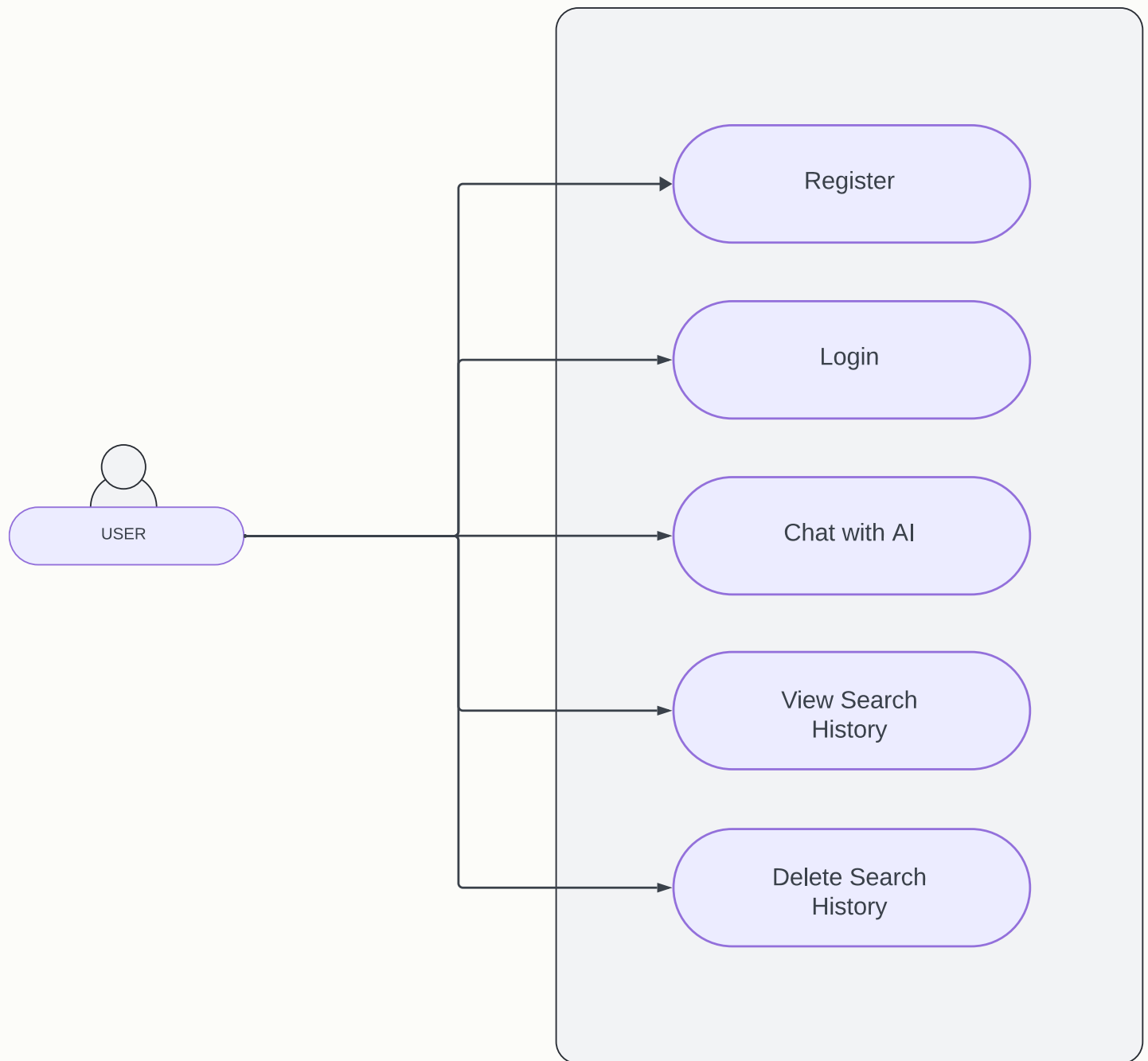


Level: 1





7.4 Use Case Diagram



7.5 Data Dictionary

User Collection:

- username:
 - Type: String
 - Description: The unique username of the user.
 - Constraints: Required, Unique
- email:
 - Type: String

- Description: The email address of the user.
 - Constraints: Required, Unique, Valid Email Format
- password:
 - Type: String
 - Description: The hashed password of the user.
 - Constraints: Required

SearchHistory Collection:

- user:
 - Type: ObjectId
 - Description: References the user who created the search history entry.
 - Constraints: Required
- prompt:
 - Type: String
 - Description: The search prompt entered by the user.
 - Constraints: Required
- response:
 - Type: String
 - Description: The response returned to the user based on the search prompt.
 - Constraints: Required
- createdAt:
 - Type: Date
 - Description: The timestamp when the search history entry was created.
 - Constraints: Default to the current date/time

8. Designing

8.1 Design Principles

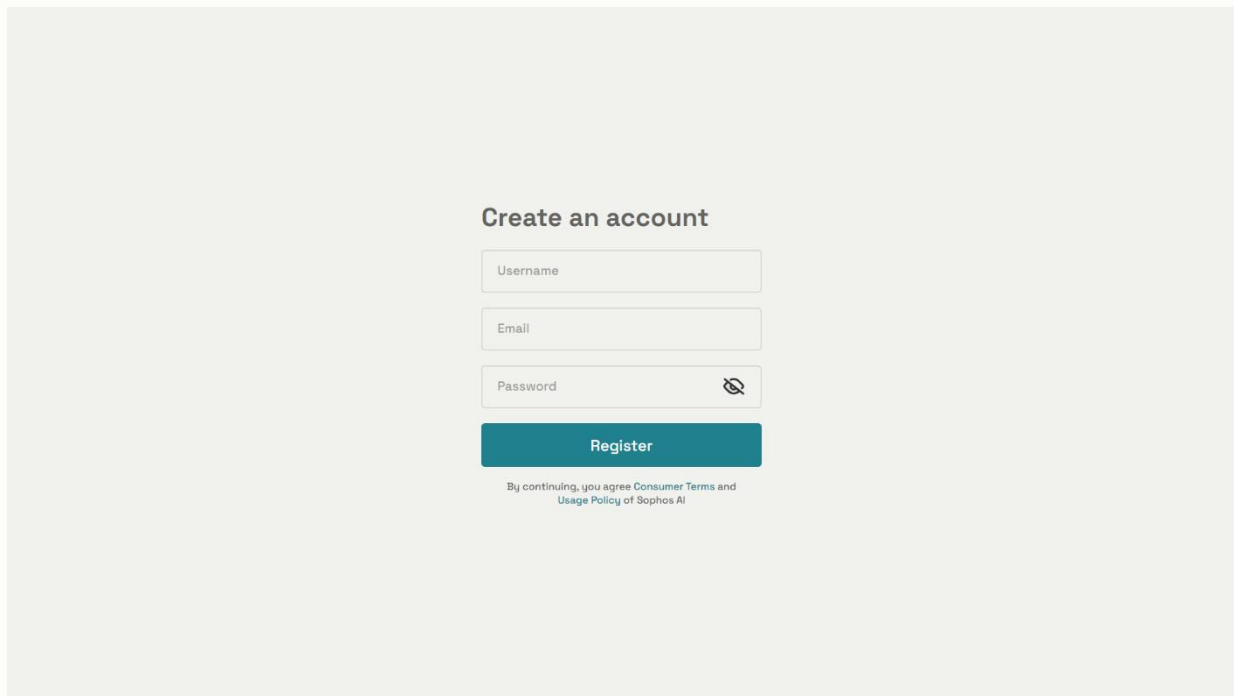
The design of the AI chatbot follows several key UI/UX design principles to ensure a seamless user experience:

1. Simplicity:
 - The interface must be intuitive and easy to use, focusing on core functionalities without unnecessary clutter. The chatbot window should present clear input areas and simple controls for interaction.
2. Consistency:
 - Consistent use of color schemes, typography, and iconography across the application. This includes consistent placement of buttons (e.g., "Send" button for chat) and menus, making navigation predictable.
3. Feedback:
 - The system must provide clear feedback after every user interaction. For example, when a user submits a query, the system should display a loading animation until the response is received, ensuring users are informed of progress.
4. Accessibility:
 - The design must follow accessibility guidelines to ensure that users with disabilities can easily interact with the system. This includes using appropriate contrast ratios, providing alt text for images, and enabling keyboard navigation.
5. Aesthetic Minimalism:
 - The interface design should maintain a clean, professional look without overwhelming users with unnecessary elements. White space, readable fonts, and minimal distractions help to keep the user focused on their interaction with the chatbot.
6. User-Centric Design:
 - The design process must prioritize the user's needs. The interface should be flexible enough to accommodate a variety of use cases, whether the user is seeking simple information or engaging in complex queries.
7. Error Prevention and Handling:
 - Users should be guided to avoid errors. If errors occur (e.g., invalid input), they should be handled gracefully with clear, actionable error messages that help users correct the issue.

8.2 User Interface

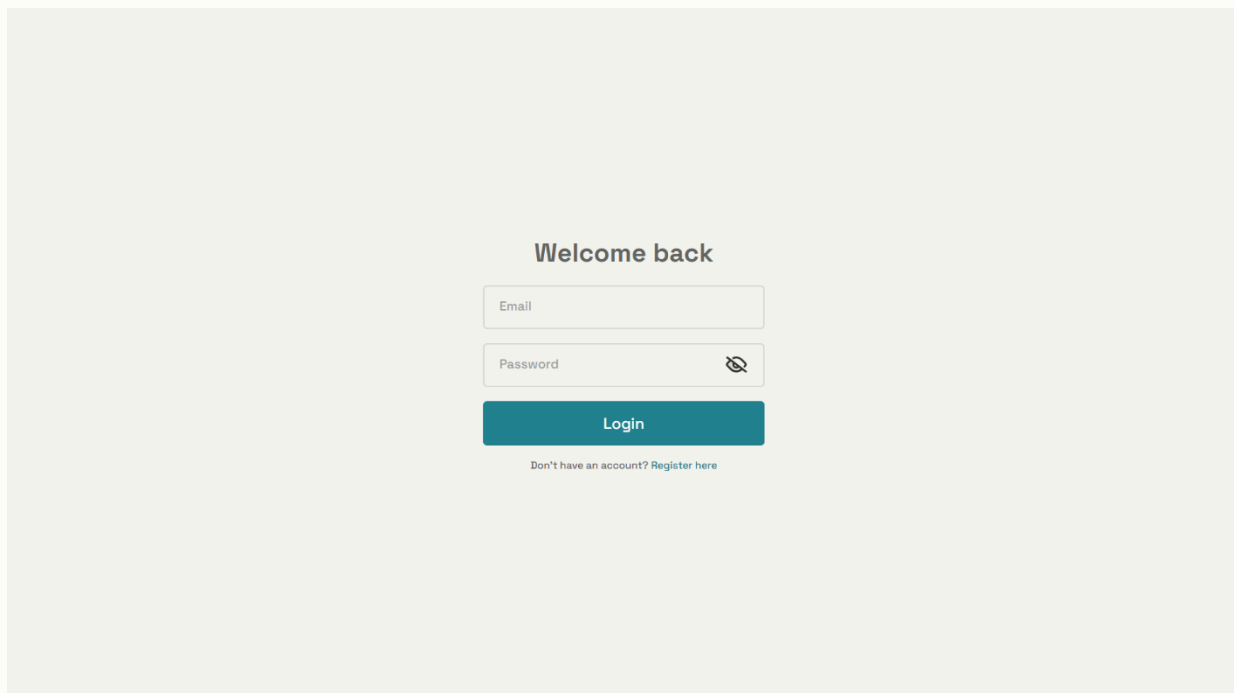
The user interface includes:

- Registration: For user to create an account



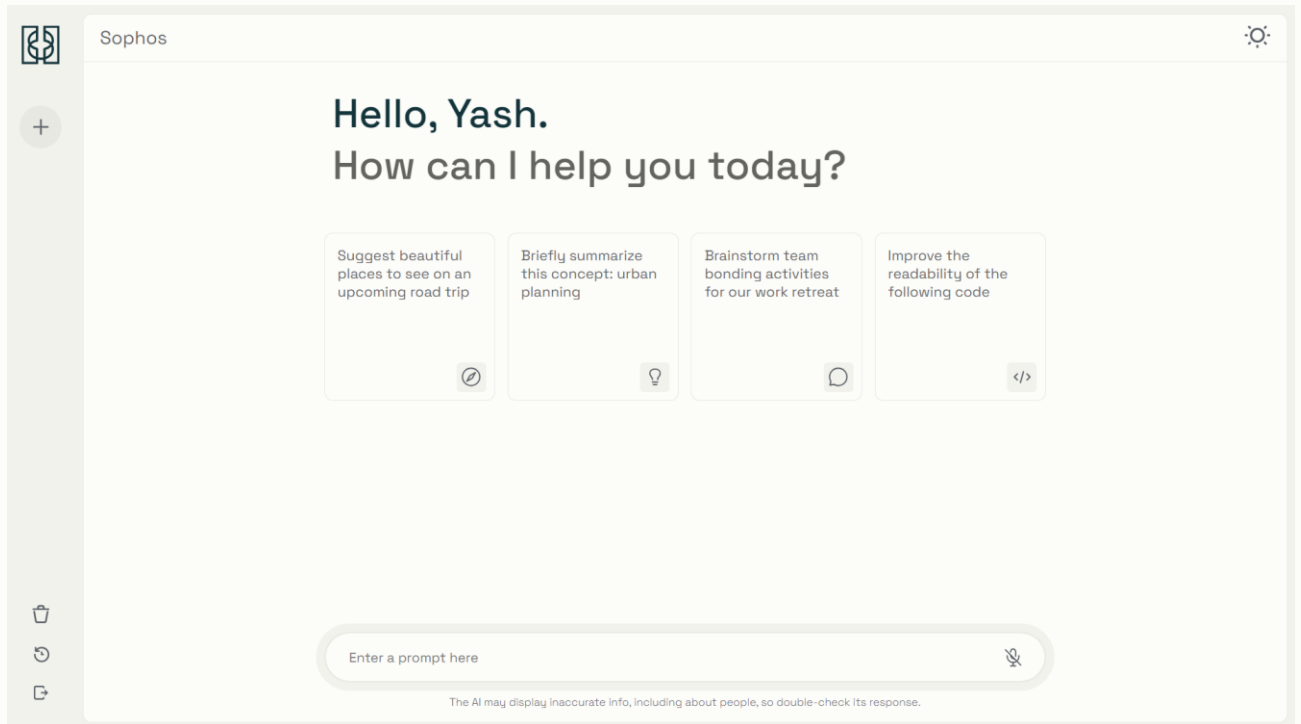
A registration form titled "Create an account" centered on a light gray background. The form consists of three input fields: "Username", "Email", and "Password". The "Password" field includes a toggle icon (an eye with a slash) to the right of the input box. Below the input fields is a teal button labeled "Register". Underneath the button, there is a line of small text: "By continuing, you agree Consumer Terms and Usage Policy of Sophos AI".

- Login: For users to access their account

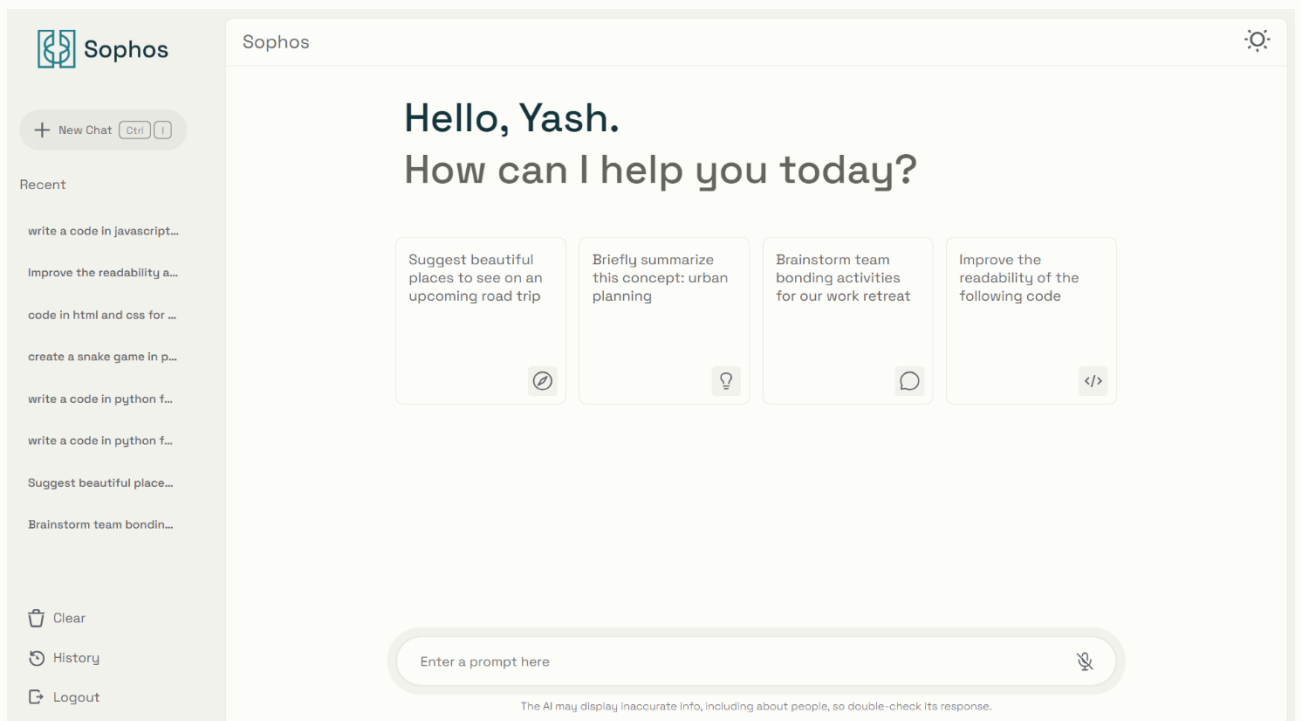


A login form titled "Welcome back" centered on a light gray background. The form consists of two input fields: "Email" and "Password". The "Password" field includes a toggle icon (an eye with a slash) to the right of the input box. Below the input fields is a teal button labeled "Login". Underneath the button, there is a line of small text: "Don't have an account? Register here".

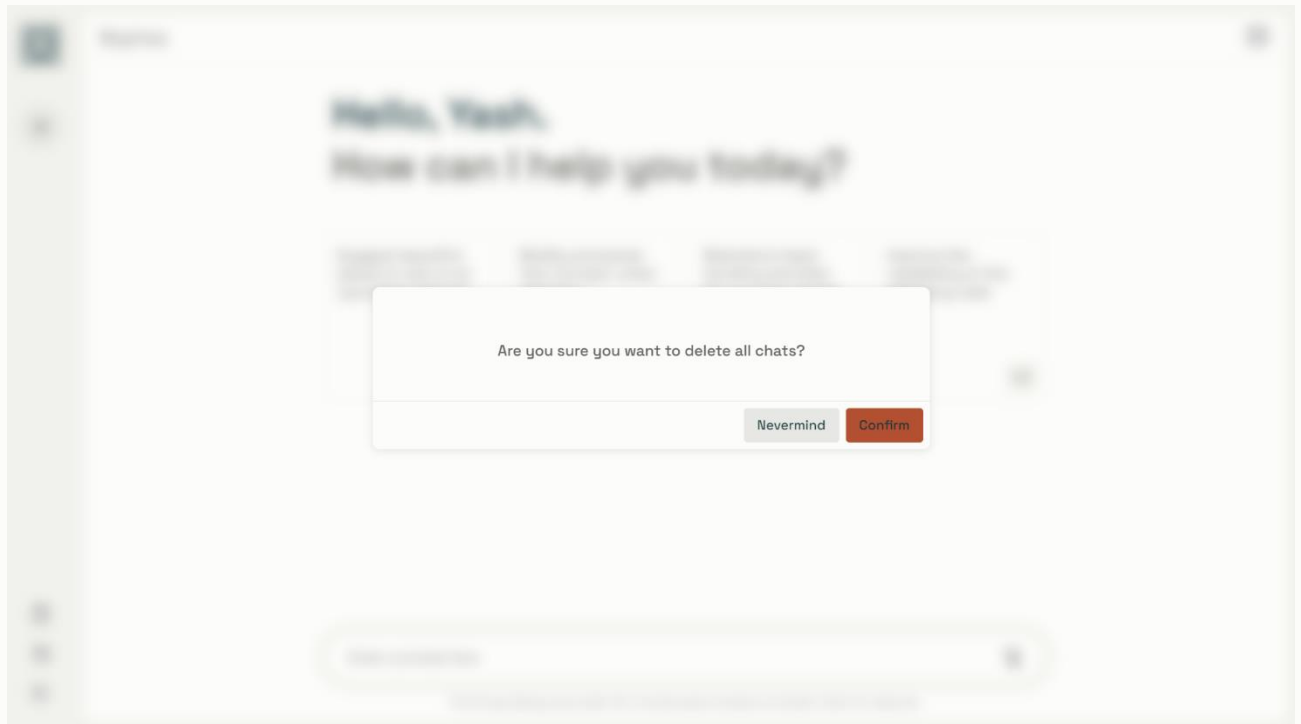
- Chat Window: For users to interact with the chatbot.



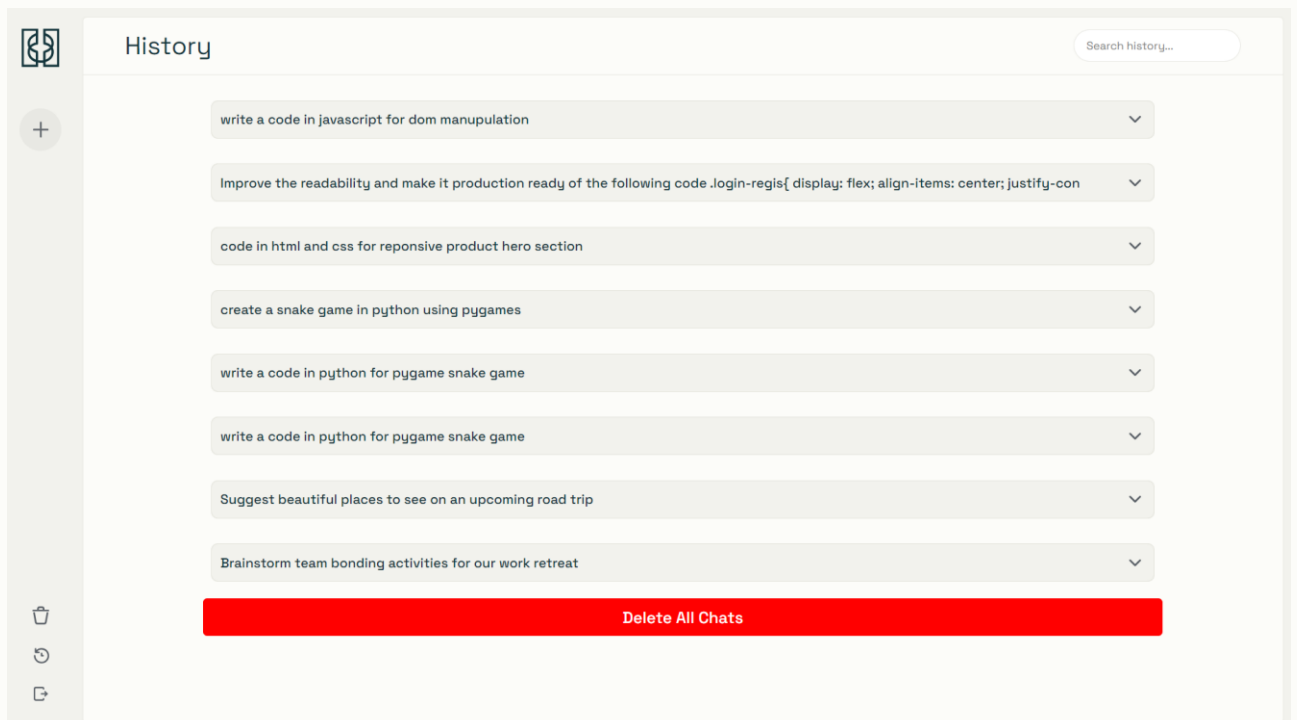
- Sidebar: For managing chat history and user settings.



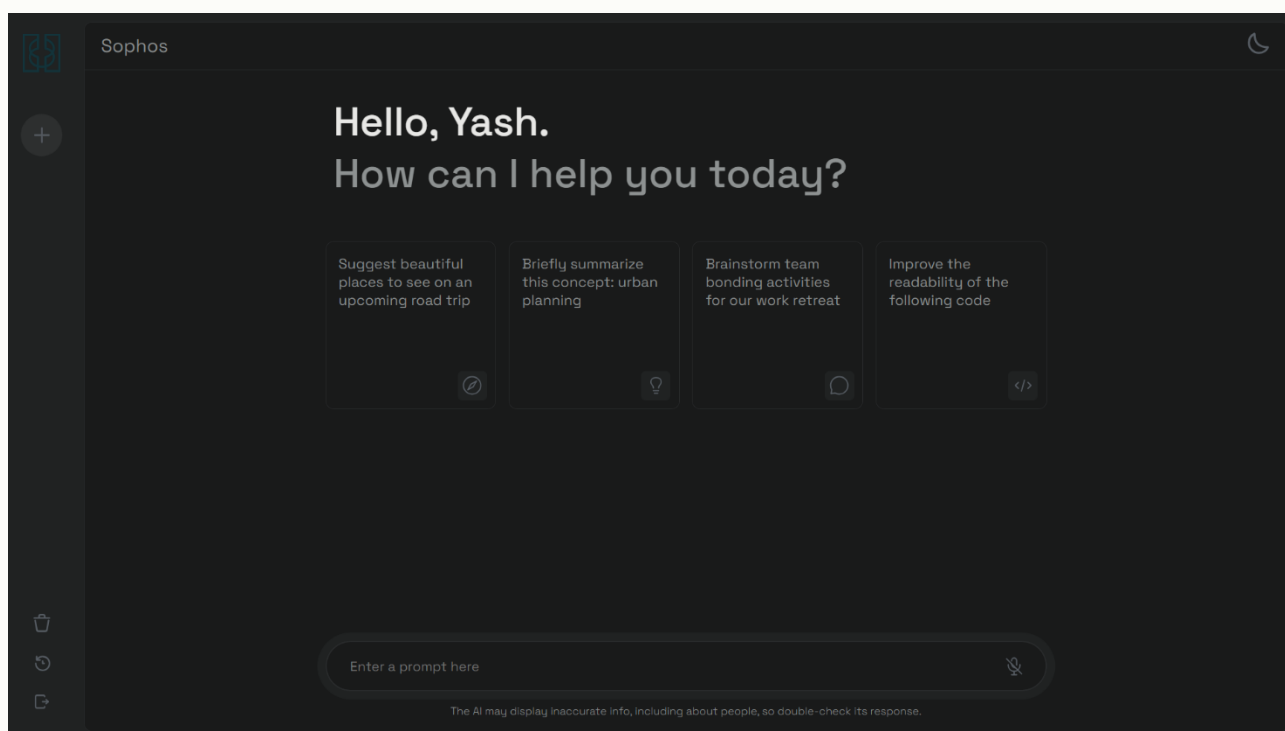
- Popup: For confirming actions that are irreversible.



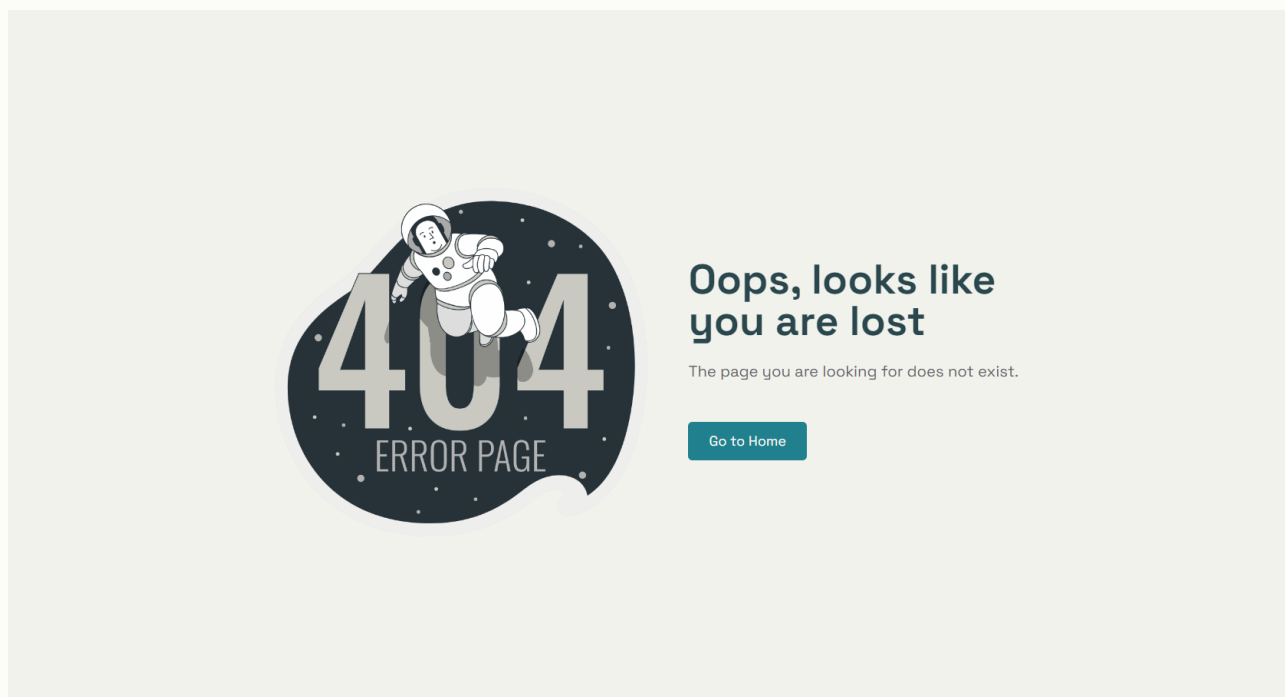
- History: To access and view past chat history.



- Dark Mode: Reduces eye strain while browsing during night-time.



- 404 Page: Help user navigate to home if the URL entered is invalid.



9. Testing

9.1 Software Testing

- Functional Testing: Verify that all features work as expected.
- Integration Testing: Ensure that different components of the system interact correctly.

9.2 Unit Testing

- Frontend: Test React components and UI elements.
- Backend: Test API endpoints and server logic.

9.3 System Testing

- End-to-End Testing: Simulate user interactions and verify end-to-end functionality.

10. References

1. React.js Documentation

- URL: <https://react.dev/learn>
- Description: Official documentation for the React JavaScript library, which was used for developing the frontend of the AI chatbot system.

2. Node.js Documentation

- URL: <https://nodejs.org/docs/latest/api>
- Description: Comprehensive reference for Node.js, the backend JavaScript runtime environment used for the server-side logic of the project.

3. Express.js Documentation

- URL: <https://expressjs.com/en/guide/routing.html>
- Description: The official documentation for Express.js, the web framework used for building the backend REST API and routing system.

4. MongoDB Documentation

- URL: <https://www.mongodb.com/docs/manual>
- Description: Reference documentation for MongoDB, the NoSQL database used to store user credentials and chat history for the AI chatbot.

5. Gemini API Documentation

- URL: <https://ai.google.dev/gemini-api/docs>
- Description: API documentation for the Gemini AI service, which provides the AI-driven responses for the chatbot's user queries.

6. JWT (JSON Web Tokens) Documentation

- URL: <https://jwt.io/introduction>
- Description: Documentation for JSON Web Tokens, which are used for secure authentication and authorization within the system.