**Synopsis On**

# ****AskAnything: An AI-Powered Chatbot Inspired by Gemini AI****

Submitted to the **Uttaranchal University** in partial fulfilment of the requirements for the award of the Degree of

**MASTER OF COMPUTER APPLIC ATIONS**

Submitted by

**Abhishek Lathiyan**

**(Learner ID: 2226010113)**

Under the Guidance of

**Guide Name with Designation (Faculty Guide)**

# UTTARANCHAL UNIVERSITY, DEHRADUN

**Table of Content**

|  |  |  |
| --- | --- | --- |
| S.no | Topic | Page No |
| 1 | Introduction and Objectives of the Project. | 3-4 |
| 2 | Tools/Platform, Hardware and Software Requirement specifications. | 4-6 |
| 3 | Problem Definition, Requirement Specifications (Detailed functional Requirements and Technical Specifications), Project Planning and Scheduling (Gantt chart and PERT chart). | 6-8 |
| 4 | Analysis (Data Models like 0, 1 and 2 level DFDs, Complete ER Diagrams with cardinality, Class Diagrams etc. as per the project requirements). | 9-9 |
| 5 | A complete structure which includes: | 10-11 |
| a. Number of modules and their description to provide estimation of the | 10-11 |
| b. student’s effort on the project. Along with process logic of each Module. | 10-11 |
| c. Data Structures as per the project requirements for all the modules. | 10-11 |
| d. Process Logic of each module. | 10-11 |
| e. List of reports that are likely to be generated. | 10-11 |
| 6 | Proposed security mechanisms at various levels | 11-11 |
| 7 | Future scope and further enhancement of the project. | 12-12 |
| 8 | Bibliography | 12-12 |

**Introduction and Objectives of the Project**

**Introduction**

With the advancement of artificial intelligence, AI-powered chatbots have revolutionized how people interact with technology. These chatbots bridge the gap between human needs and machine capabilities, providing quick and intelligent responses in real time. One such project is "**AskAnything**", an AI chatbot inspired by Google’s Gemini AI. The core purpose of this chatbot is to deliver real-time, AI-generated responses to user queries by leveraging the Google Generative AI API. The system is designed to be efficient, user-friendly, and responsive, incorporating modern web development technologies.

The **frontend** of "AskAnything" is built using **React** and **Vite**. React is a powerful JavaScript library for building user interfaces, known for its component-based architecture and efficient rendering process. React helps in breaking down the UI into reusable components, which makes the development process smoother and the code more maintainable. To enhance performance, Vite is used as a build tool. Vite provides an optimized development environment by offering faster server startup and hot module replacement, making it ideal for modern web applications.

In addition to React, the project uses React Hooks such as **useState** and **useContext** to manage component state and context. React Hooks simplify state management by allowing developers to use state and other React features without writing class components. For example, useState manages the chatbot's input state and recent chat history, while useContext helps in sharing state between components efficiently. This results in a cleaner, more readable code structure.

The chatbot’s user interface design is handled by **Tailwind CSS**, a utility-first CSS framework. Tailwind CSS enables rapid development of responsive and visually appealing designs by providing low-level utility classes for styling elements. Unlike traditional CSS frameworks, Tailwind allows developers to customize designs directly in the markup, eliminating the need for writing custom CSS. This approach ensures consistency in design and significantly reduces the time required for styling. Tailwind's responsive design utilities make the chatbot adaptable to various devices, enhancing the user experience.

To add interactive and visually appealing icons, **React-Icons** is used. This library offers a wide range of customizable icons that can be easily integrated into React components. React-Icons provides icons from popular sets like Font Awesome and Material Icons, ensuring flexibility in design and enhancing the chatbot's visual appeal.

The **backend** of "AskAnything" relies on **Node.js** and **JavaScript**. Node.js is a runtime environment that allows JavaScript to be executed on the server side. It provides a non-blocking, event-driven architecture, making it well-suited for handling API requests and responses asynchronously. Node.js, in combination with **npm** (Node Package Manager), allows developers to install and manage dependencies easily. This ensures that the chatbot can effectively communicate with the Google Generative AI API, send user queries, and process the responses in real time.

The project does not rely on a database; instead, it uses React state and session cookies to maintain recent chat history. This ensures that the recent chats are available during the session but are erased once the page is refreshed, maintaining simplicity and efficiency.

In summary, "**AskAnything**" leverages a combination of cutting-edge technologies to deliver a seamless and responsive chatbot experience. The frontend is powered by **React, Vite, React Hooks, Tailwind CSS,** and **React-Icons**, while the backend **uses Node.js, JavaScript**, and **npm** to handle API communication. Together, these technologies create a robust and modern AI chatbot that provides users with real-time responses and a dynamic interface, making AI-driven interactions smoother and more accessible.

**Objectives**

1. The primary goal of this project is to **develop an intuitive AI chatbot** that effectively processes user queries and returns AI-generated responses in real time. By leveraging modern web technologies and AI capabilities, the chatbot provides a smooth and interactive user experience, ensuring quick and relevant answers. The project aims to create a system where users can seamlessly engage with the AI to ask various questions and receive contextually accurate responses.
2. To achieve this, a **dynamic user interface** is built using React and Tailwind CSS. React’s component-based architecture allows the interface to be modular, easy to maintain, and highly responsive. Tailwind CSS, with its utility-first approach, enables rapid development of a clean and visually appealing design, ensuring that the chatbot is both functional and aesthetically pleasing. This combination helps create an interface that is responsive and adaptable to different screen sizes, providing a seamless experience on desktops, tablets, and mobile devices.
3. The integration of the **Google Generative AI API** plays a crucial role in generating accurate and context-aware responses. The chatbot sends user queries to the API, processes the data received, and displays the response in real time. This integration ensures that users receive high-quality AI-generated answers, enhancing the chatbot's reliability and effectiveness.
4. A key feature of the chatbot is the "**Recent Chat**" functionality, which uses session cookies and React state to store chat history temporarily. This means users can view their recent interactions during a session, but the data is cleared when the page is refreshed. This approach simplifies data management while maintaining a useful feature for short-term recall.
5. To enhance **user experience** (UX), the chatbot incorporates dynamic elements like showing the submit button only when a prompt is entered. This feature prevents accidental submissions and provides a cleaner interface.
6. Throughout the development process, challenges related to **API integration** were addressed, particularly issues with handling and displaying API data correctly. Debugging efforts ensured that responses are accurate and free of errors. The chatbot is designed to be responsive, offering consistent performance and usability across different devices and screen sizes, making it accessible to a wider audience.

**Tools/Platform, Hardware, and Software Requirement Specifications**

**Tools and Platforms**

Front-End Technologies:

* **React:** JavaScript library for building dynamic user interfaces.
* **Vite:** A fast build tool for modern web projects.
* **Tailwind CSS:** Utility-first CSS framework for responsive design.
* **React Hooks:** For managing component state (useState) and context (createContext).
* **React-Icons:** Library for integrating customizable icons.

Back-End Technologies:

* **Node.js**: JavaScript runtime for handling server-side operations.

API Integration:

* **Google Generative AI API:** Provides AI-generated responses for user queries.

Development Tools:

* **Visual Studio Code (VS Code):** IDE for code development.
* **Npm (Node Package Manager):** For managing dependencies.
* **Git and GitHub:** Version control and code repository hosting.

Deployment Platforms:

* **Vercel:** For front-end deployment and backend-server

**Hardware Requirements**

|  |  |
| --- | --- |
| Component | Specification |
| Processor | Intel Core i3 or higher |
| RAM | 4 GB minimum |
| Storage | 128 GB SSD or higher |
| Internet Connection | High-speed internet for API calls |

**Software Requirements**

|  |  |
| --- | --- |
| Software | Version |
| Operating System | Windows 10/Linux/macOS |
| Node.js | Version 16 or higher |
| React | Latest Stable Version |
| Tailwind CSS | Latest Stable Version |
| Browser | Chrome/Firefox/Edge |

**Problem Definition**

**Problem Statement**

The objective of "**AskAnything**" is to build an AI chatbot similar to Google’s Gemini AI. Users often seek immediate, accurate answers from AI chatbots, but implementing this effectively can be challenging. The main issues addressed by this project include integrating the Google Generative AI API, ensuring a seamless user interface, and maintaining recent chat history temporarily within a session.

**Key Challenges**

API Integration:

* Properly fetch and display accurate responses from the Google Generative AI API.
* Handle issues where the API returns incorrect data formats (e.g., regex patterns).

State Management:

* Manage recent chat history without using a database.
* Implement session cookies and React state effectively to preserve chat data during a session.

Dynamic UI Elements:

* Ensure the submit button only appears when a valid prompt is entered.

Debugging:

* Resolve errors related to data fetching and displaying responses accurately.

**Requirement Specifications**

**Functional Requirements**

User Interface Requirements:

* A clean, responsive chat interface.
* The submit button should only appear when a prompt is entered.
* Display AI-generated responses immediately after submission.

Chat Functionality:

* Users can type and submit queries.
* Display the user's prompt and the AI’s response.
* Maintain recent chat history within the session.

API Integration Requirements:

* Integrate the Google Generative AI API for fetching responses.
* Handle API errors and display appropriate messages.

Session Management:

* Use React state and session cookies to manage recent chat history.

**Technical Requirements**

Front-End Specifications:

* **Framework:** React (with Vite for faster builds).
* **Styling:** Tailwind CSS for responsive design.
* **State Management:** React Hooks (useState and createContext).

Back-End Specifications:

* Server: Node.js

API Integration:

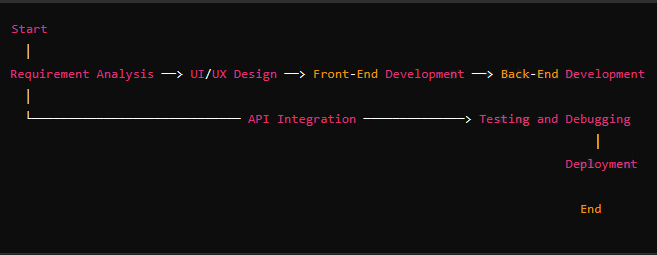
* Google Generative AI API: For real-time AI responses.

**Project Planning and Scheduling**

**Gantt Chart:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Week 1 | Week 2 | Week 3 | Week 4 | Week 5 | Week 6 |
| Requirement Analysis | ✔ |  |  |  |  |  |
| UI/UX Design |  | ✔ | ✔ |  |  |  |
| Front-End Development |  |  | ✔ | ✔ |  |  |
| Back-End Development |  |  |  | ✔ | ✔ |  |
| API Integration |  |  |  |  | ✔ |  |
| Testing and Debugging |  |  |  |  |  | ✔ |
| Deployment |  |  |  |  |  | ✔ |

**Pert Chart:**

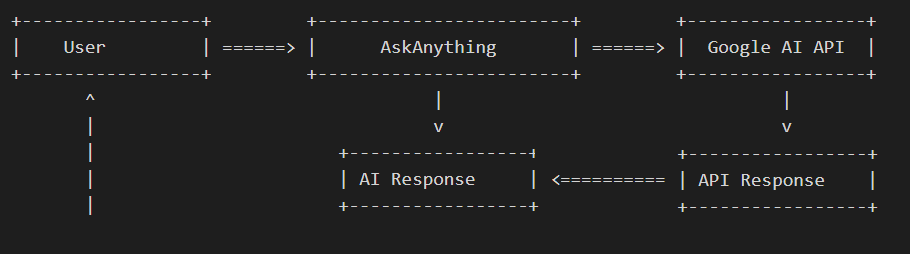


**Analysis**

**Data Flow Diagrams (DFDs)**

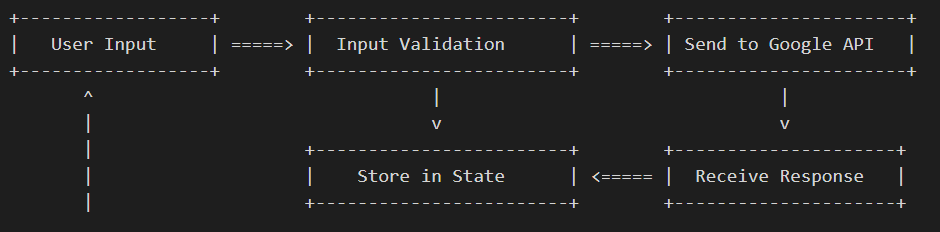
Data Flow Diagrams help in understanding the flow of data within the system. Since the project does not use a database, the DFDs focus on how user input flows through the system and interacts with the Google Generative AI API.

**Level 0 DFD (Context Diagram)**



* User inputs a prompt into the system.
* AskAnything processes the prompt and sends it to the Google AI API.
* The API returns a response to AskAnything, which then displays it to the User.

**Level 1 DFD**



**ER Diagram**

Since the project does not use a database, a traditional ER (Entity-Relationship) Diagram is not applicable. However, entities can be represented conceptually:

* **User:** Represents the person interacting with the system.
* **Prompt:** Represents the input query.
* **Response:** Represents the AI-generated response.

**Project Structure**

**Number of Modules and Their Description**

The "AskAnything" project consists of the following key modules:

User Input Module

* **Description:** Manages the user’s input prompt.
* **Functionality:** Captures the input, validates it, and enables the submit button dynamically.

AI Response Module

* **Description:** Manages the user’s input prompt.
* **Functionality:** Captures the input, validates it, and enables the submit button dynamically.

Recent Chat Module

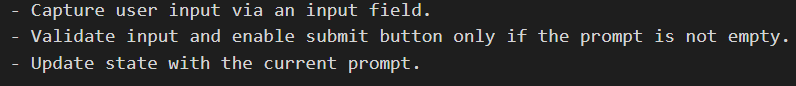
* **Description:** Manages the recent chat history within the session.
* **Functionality:** Stores recent chats in React state and displays them temporarily.

UI Module

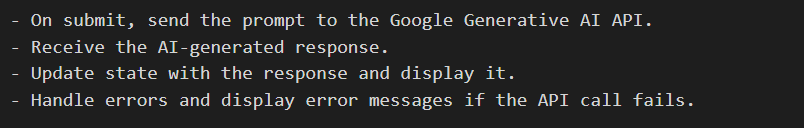
* **Description:** Provides the interface layout and styling using Tailwind CSS.
* **Functionality:** Ensures a responsive, dynamic user interface.

**Process Logic of Each Module**

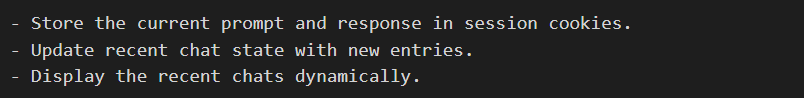
User Input Module



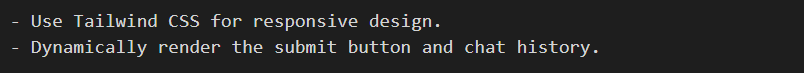
AI Response Module



Recent Chat Module



UI Module



**Data Structures**

Since no persistent data storage is used, the primary data structures are React state variables:

* **Prompt State:** useState("") – Stores the current user prompt.
* **Recent Chat State:** useState([]) – Stores the list of recent chats temporarily.

**Proposed Security Mechanisms**

**Front-End Security**

Input Validation:

* Ensure that user inputs are sanitized to prevent malicious entries.

Session Security:

* Use session cookies securely to avoid tampering.

API Key Protection:

* Store the API key securely (e.g., in environment variables).

**Network Security**

HTTPS:

* Ensure that the application communicates via HTTPS for data encryption.

Rate Limiting:

* Implement rate limiting to protect the API from abuse.

**Future Scope and Further Enhancements**

**Future Scope**

Persistent Chat History:

* Integrate a database (e.g., MongoDB) to store chat history permanently

User Authentication:

* Implement login functionality for personalized user experiences.

Voice Input and Output:

* Add speech-to-text and text-to-speech capabilities for accessibility.

Multilingual Support:

* Enable responses in multiple languages.

Enhanced UI/UX:

* Improve design and add themes for customization.

**Further Enhancements**

API Alternatives:

* Integrate additional AI models to provide varied responses.

Analytics Dashboard:

* Track and visualize user interactions.

Chat Export Feature:

* Allow users to export chat history as PDFs or text files.

**Bibliography**

* React Documentation – <https://reactjs.org/docs>
* Tailwind CSS Documentation – <https://tailwindcss.com/docs>
* Node.js Documentation – <https://nodejs.org/en/docs>
* Google Generative AI API – <https://developers.google.com/ai>
* Vite Documentation – <https://vitejs.dev/guide>