**Telegram AI Chatbot**

**Introduction**

The Telegram AI Chatbot is a single-page React application designed to provide a user-friendly interface for interacting with a chatbot, registering user details, and uploading files for analysis. The core functionality includes user registration, chat interaction, file analysis, and web search capabilities.

**Project Overview**

The Telegram AI Chatbot is a comprehensive application that allows users to:

* Register their details and contact information.
* Interact with a simulated AI chatbot.
* Upload files for analysis.
* Perform web searches and view results.

**Key Components**

1. **User Registration and Contact Request**
   * Collects user's first name, username, and phone number.
   * Displays confirmation messages after submission.
2. **Gemini-Powered Chat**
   * Allows users to send messages and interact with a simulated AI chatbot.
   * Displays chat history with timestamps and avatars.
   * Simulates bot responses based on user queries.
3. **Image/File Analysis**
   * Enables users to upload images or files.
   * Simulates file analysis and displays detailed results.
4. **Web Search**
   * Allows users to perform web searches by typing **/websearch** followed by a query.
   * Simulates search results and displays top web links.

**Features**

**User Registration and Contact Request**

* **Tasks**
  + Create a form for user registration.
  + Collect first name and username.
  + Display a confirmation message after submission.
* **Subtasks**
  + Implement state management for first name, username, and phone number.
  + Create input fields for first name, username, and phone number.
  + Create buttons to submit the contact request and phone number.
  + Display confirmation messages once the forms are submitted.

**Gemini-Powered Chat**

* **Tasks**
  + Create a chat interface.
  + Allow users to send messages.
  + Display chat history with timestamps.
  + Simulate bot responses.
* **Subtasks**
  + Implement state management for chat history.
  + Create an input field and send button for user messages.
  + Display messages with timestamps and sender information.
  + Create a function to generate bot responses based on user input.
  + Update chat history with bot responses.

**Image/File Analysis**

* **Tasks**
  + Create a file upload interface.
  + Display uploaded file details.
  + Simulate file analysis and display results.
* **Subtasks**
  + Implement file upload functionality.
  + Create a button to trigger file upload.
  + Display file name and size.
  + Create a function to simulate file analysis.
  + Display the simulated analysis result.

**Web Search**

* **Tasks**
  + Create a web search interface.
  + Allow users to perform web searches.
  + Display search results and top web links.
* **Subtasks**
  + Implement web search functionality.
  + Create an input field for search queries.
  + Display search results and top web links.
  + Simulate search results based on user queries.

**Setup Instructions**

**Prerequisites**

* Node.js and npm installed on your machine.

**Steps to Set Up the Project**

1. **Create a New React App**:
2. npx create-react-app telegram-chatbot
3. cd telegram-chatbot
4. **Install Required Dependencies**:
5. npm install @shadcn/ui lucide-react
6. **Replace src/App.tsx**:

Open **src/App.tsx** and replace its content with the following code:

import React from 'react'

import './App.css'

import { TelegramChatbot } from './telegram-chatbot'

function App() {

return (

<div className="App">

<TelegramChatbot />

</div>

)

}

* export default App

1. **Create the Component File**:
   * Create a new file named **telegram-chatbot.tsx** in the **src** directory and paste the provided code into it.
2. **Run the Application**:
3. npm start
4. **View the Application**:
   * Open your web browser and go to **http://localhost:3000** to see the Telegram AI Chatbot in full screen.

**Implementation Details**

**User Registration and Contact Request**

* **State Management**:
  + **firstName**, **username**, **phoneNumber**, **contactSubmitted**, **fileMetadataSaved**.
* **Components**:
  + **Input** for first name, username, and phone number.
  + **Button** to submit the contact request and phone number.
  + **Label** for input fields.
* **Simulated Actions**:
  + Simulate saving user data to MongoDB by logging to the console.

**Gemini-Powered Chat**

* **State Management**:
  + **userQuery**, **chatHistory**.
* **Components**:
  + **Input** for user messages.
  + **Button** to send messages.
  + **Card** to display chat history.
  + **Avatar** for user and bot avatars.
* **Simulated Actions**:
  + Simulate bot responses using predefined messages.
  + Simulate saving chat history to MongoDB by logging to the console.

**Image/File Analysis**

* **State Management**:
  + **file**, **fileAnalysis**, **fileMetadataSaved**.
* **Components**:
  + **Input** for file upload.
  + **Button** to trigger file upload.
  + **Card** to display file details and analysis results.
* **Simulated Actions**:
  + Simulate file analysis using detailed descriptions.
  + Simulate saving file metadata to MongoDB by logging to the console.

**Web Search**

* **State Management**:
  + **webSearchQuery**, **webSearchResults**.
* **Components**:
  + **Input** for search queries.
  + **Button** to perform search.
  + **Card** to display search results and top web links.
* **Simulated Actions**:
  + Simulate search results using predefined summaries and links.
  + Simulate saving search queries and results to MongoDB by logging to the console.

**Design and User Experience**

* **Responsive Design**: The app is designed to be responsive and works well on both mobile and desktop devices.
* **Tailwind CSS**: The app uses Tailwind CSS for styling, ensuring a clean and intuitive design.
* **Shadcn UI Components**: The app uses Shadcn UI components for consistent styling and functionality.
* **Lucide Icons**: The app uses Lucide icons for visual elements, such as the upload icon and phone icon.

**Simulated Interactions**

**Gemini API Simulation**

* **Bot Responses**: The **getBotResponse** function provides predefined responses based on user queries to simulate the Gemini API.
* **File Analysis**: The **simulateFileAnalysis** function simulates file analysis using the Gemini API with detailed descriptions of the contents and objects in the image.

**MongoDB Simulation**

* **User Registration Data**: Simulated saving to MongoDB by logging user data to the console.
* **Phone Number**: Simulated saving to MongoDB by logging phone number to the console.
* **Chat History**: Simulated saving to MongoDB by logging chat history to the console.
* **File Metadata**: Simulated saving to MongoDB by logging file metadata to the console.
* **Web Search Queries and Results**: Simulated saving to MongoDB by logging search queries and results to the console.

**Conclusion**

The Telegram AI Chatbot provides a functional and engaging experience for users, allowing them to register, chat with a bot, upload files for analysis, and perform web searches. The app is designed to be user-friendly and responsive, with simulated interactions to mimic real-world functionality.