INSIGHTBOT

INSIGHTBOT

PROJECT REPORT

Submitted by

Himanshu Kushwaha(201500296)

Raj Sinha(201500552)

Subodh Srivastava(201500710)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

Computer Engineering and Application

GLA University, Mathura

GLA UNIVERSITY MATHURA

APRIL 2023

BONAFIDE CERTIFICATE

Certified that this project report **"INSIGHTBOT"** is the bonafide work of "**HIMANSHU KUSHWAHA , RAJ SINHA and SUBODH SRIVATSATA"** who carried out the project work under my/our supervision.

SIGNATURE

Mr. ROHIT AGRAWAL

(HOD)

SIGNATURE

Ms. Ruchi Talwar

(SUPERVISOR)

TABLE OF CONTENTS

INTRODUCTION

1.1. Project Overview

Our project, InsightBot, is a cutting-edge AI chatbot designed to provide fast and accurate support to users seeking information or assistance. With the rapid pace of modern life, users need access to information quickly, and our chatbot is the perfect solution for this demand. The InsightBot AI uses advanced artificial intelligence technology to understand user input and respond with relevant information in real-time. The chatbot also includes a suggestion feature that predicts the user's next question as they type, ensuring that users receive the most accurate and helpful information possible. One of the unique features of InsightBot is its suggestion feature. The chatbot can predict what users will ask next as they type, making it faster and easier to get the information they need. This feature also reduces the likelihood of errors, ensuring that users receive the most accurate information possible. Overall, InsightBot is an essential tool for any modern customer service or support system. Its ability to provide fast and accurate information, along with its advanced suggestion feature, makes it a valuable asset for anyone in need of quick assistance. With InsightBot, users can get the help they need quickly and efficiently, without the need to wait for a human representative to become available.

1.2. Project Description

The InsightBot project is a cutting-edge AI chatbot that offers users quick and accurate assistance in retrieving information. The project utilizes the OpenAI API, a leading artificial intelligence technology, to enable the chatbot to understand user input and provide relevant responses. This technology is a significant step forward in improving the speed and accuracy of information retrieval, making the chatbot a highly effective and efficient solution.

One of the project's innovative features is the suggestion feature, which predicts the user's intended question as they type. This helps users save time and reduces errors in information retrieval, resulting in an improved user experience and a more efficient chatbot.

The InsightBot project offers numerous benefits for businesses and organizations looking to provide efficient and cost-effective customer support. The chatbot can handle a large volume of queries simultaneously, making it a scalable solution for businesses that need to provide assistance to a large number of users. Additionally, the chatbot is available 24/7, making it an ideal solution for businesses that operate around the clock.

Overall, the InsightBot project is a powerful tool that showcases the capabilities of artificial intelligence technology and its potential to transform the way users obtain information. By leveraging the OpenAI API and including the suggestion feature, the project aims to improve user satisfaction, streamline the process of obtaining accurate information, reduce the need for human resources, and demonstrate the true potential of AI technology.

2.1 Primary Reason to Choose This Project

The primary reason to choose this AI project is its ability to provide efficient and accurate support to users. With its advanced artificial intelligence technology and suggestion feature, the chatbot can quickly and accurately respond to user queries, saving time and minimizing errors.

Another significant advantage of the chatbot is its scalability. It can handle a large volume of queries simultaneously, making it an ideal solution for businesses or organizations that need to provide assistance to a large number of users.

The project is also cost-effective as it can automate the process of answering user queries, reducing the costs associated with human resources. This makes it a highly attractive option for businesses or organizations looking to optimize their resources.

Finally, the chatbot's 24/7 availability is a critical advantage. Users can access support at any time of the day or night, making it an ideal solution for businesses or organizations

that operate around the clock. Overall, the combination of efficiency, scalability, cost-effectiveness, and 24/7 availability make this AI chatbot project an attractive.

2.2. Goals/Objectives

Our InsightBot project aims to create an AI-powered chatbot that provides quick and accurate assistance to users seeking information. Leveraging advanced artificial intelligence technology, we strive to improve the chatbot's efficiency and effectiveness. One notable feature is the suggestion feature that predicts the user's intended question, reducing the need for users to type out their entire query. This feature saves time and reduces the likelihood of errors in the user's question, resulting in the most accurate response possible.

This project has the potential to lower operational costs for businesses and organizations by reducing the need for human resources. Moreover, it demonstrates the impressive capabilities of artificial intelligence technology in transforming the way users obtain information and seek assistance.

Our objectives for this project are to enhance user satisfaction, streamline information retrieval, and demonstrate the potential of AI technology. The inclusion of the suggestion feature is a significant step forward in achieving these goals, and we are excited to witness the impact of this project on the industry.

Technologies Used

• **React Front-End** - The React library can be used to build a responsive and dynamic user interface that enhances the user experience. React allows for the creation of reusable UI components, making it easier to maintain and update the project's user interface.

React's popularity today has eclipsed that of all other front-end development frameworks. Here is why:

- Easy creation of dynamic applications: React makes it easier to create dynamic web applications because it requires less coding and offers more functionality, as opposed to JavaScript, where coding often gets complex very quickly.
- Improved performance: React uses Virtual DOM, thereby creating web applications faster. Virtual DOM compares the components' previous states and updates only the items in the Real DOM that were changed, instead of updating all of the components again, as conventional web applications do.
- Reusable components: Components are the building blocks of any React application, and a single app usually consists of multiple components. These components have their logic and controls, and they can be reused throughout the application, which in turn dramatically reduces the application's development time.
- Unidirectional data flow: React follows a unidirectional data flow. This means that when designing a React app, developers often nest child components within parent components. Since the data flows in a single direction, it becomes easier to debug errors and know where a problem occurs in an application at the moment in question.

- Small learning curve: React is easy to learn, as it mostly combines basic HTML and JavaScript concepts with some beneficial additions. Still, as is the case with other tools and frameworks, you have to spend some time to get a proper understanding of React's library.
- It can be used for the development of both web and mobile apps: We already know that React is used for the development of web application , but that's not all it can do. There is a framework called React Native, derived from React itself, that is hugely popular and is used for creating beautiful mobile applications. So, in reality, React can be used for making both web and mobile applications.
- Dedicated tools for easy debugging: Facebook has released a Chrome extension that can be used to debug React applications. This makes the process of debugging React web applications faster and easier.

The above reasons more than justify the popularity of the React library and why it is being adopted by a large number of organizations and businesses. Now let's familiarize ourselves with React's features.

ReactJS Advantages

1. React.js builds a customized virtual DOM. Because the JavaScript virtual DOM is quicker than the conventional DOM, this will enhance the performance of apps.
2. ReactJS makes an amazing UI possible.
3. Search - engine friendly ReactJS.
4. Modules and valid data make larger apps easier to manage by increasing readability.

5. React integrates various architectures.
6. React makes the entire scripting environment process simpler.
7. It makes advanced maintenance easier and boosts output.
8. Guarantees quicker rendering
9. The availability of a script for developing mobile apps is the best feature of React.
10. ReactJS is supported by a large community.

• **MongoDB Database** - MongoDB is a popular NoSQL database that can be used to store and manage data in a flexible and scalable manner. Your project may use MongoDB to store user data, chat logs, or other relevant information.

In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.

The manufacturing company 10gen has defined MongoDB as:

"MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen

MongoDB was designed to work with commodity servers. Now it is used by the company of all sizes, across all industry.

History of MongoDB

The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to window azure.

MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. It was initially developed as a PAAS (Platform as a Service). Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.

The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010.

The primary purpose of building MongoDB is:

- Scalability
- Performance
- High Availability
- Scaling from single server deployments to large, complex multi-site architectures.
- Key points of MongoDB
- Develop Faster
- Deploy Easier
- Scale Bigger

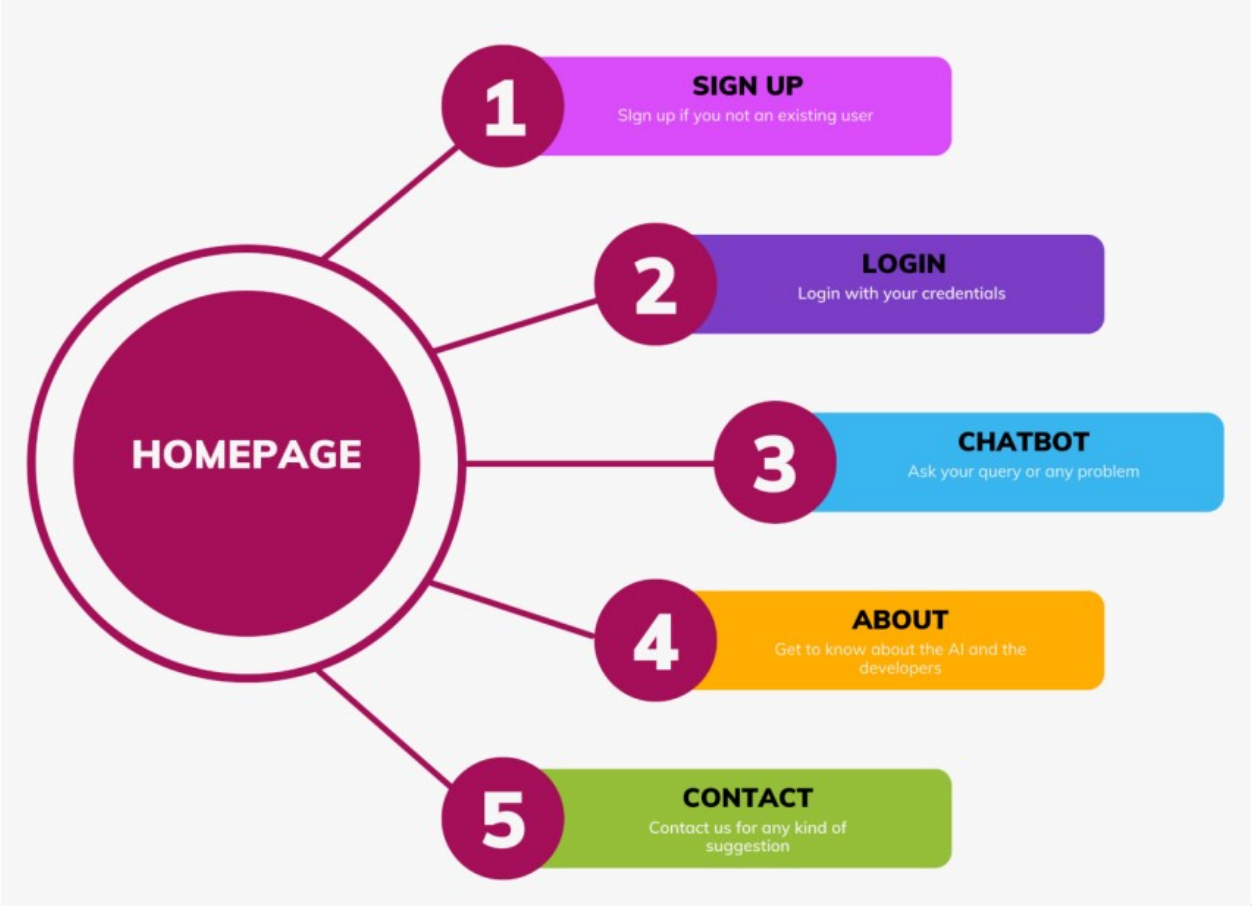
- **Node.js Backend** - Node.js is a server-side platform based on the JavaScript Engine in Google Chrome. It was created by Ryan Dahl in 2009, and the most recent version is v0.10.36. This is a cross-platform runtime environment for developing server-side and networking applications that are open source. Node.js programs are written in JavaScript and run on the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also comes with a big library of JavaScript modules, which makes developing Node.js web applications much easier.

The Node js program runs in a single process rather than establishing a new thread for each request. Blocking behavior is the exception rather than the rule in Node.js, because the standard library offers a set of asynchronous I/O primitives that prevent JavaScript code from blocking, and libraries in Node.js are frequently written using non-blocking

paradigms. The popularity of Node.js is skyrocketing right now. Netflix, Uber, PayPal, Twitter, and more well-known companies are presently using Node.js. According to StackOverflow's 2021 Developer Survey, Node.js is the 6th most popular technology among programmers, with nearly one-third of professional developers putting it as their first preference.

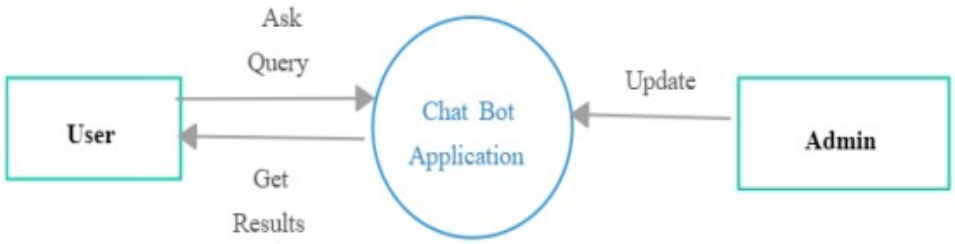
- **Third-Party API Integration** – OpenAI api is used to provide the ai chatbot experience and handle the query of the use

DESIGN FLOW/PROCESS

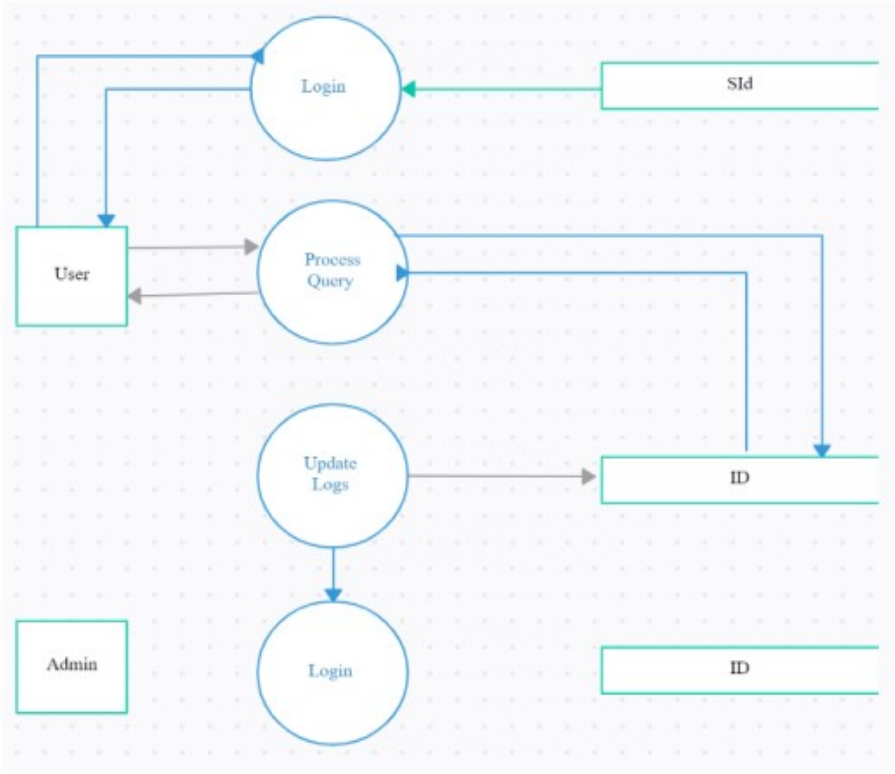


Data Flow Diagram

Level 0



Level 1



Implementation plan/methodology

An AI bot powered by OpenAI API works by leveraging state-of-the-art natural language processing and machine learning algorithms to accomplish its tasks. Here are the general steps that an AI bot using OpenAI API may take to complete its tasks:

Query Processing: The AI bot processes user queries using natural language processing algorithms and sends the queries to the OpenAI API for further processing.

Response Generation: OpenAI API generates responses based on the processed query and sends them back to the AI bot.

Deployment: Once the AI bot receives a response from OpenAI API, it can be deployed to perform its specific tasks, such as answering questions, making recommendations, or providing customer service.

Continuous Improvement: As the AI bot interacts with users and receives feedback, it can continue to learn and improve its performance over time. This is achieved through ongoing training and updates to the OpenAI API algorithms.

Overall, the working methodology of an AI bot using OpenAI API is to leverage advanced natural language processing and machine learning algorithms to perform tasks that would typically require human intelligence, such as understanding natural language and generating responses based on complex information..

SOURCE CODE

App.js

```jsx
import React from "react";
import { Route, Routes } from "react-router-dom";
import Home from "./Componenets/SubComponents/Home";
import About from "./Componenets/SubComponents/About";
import Contact from "./Componenets/SubComponents/Contact";
import Signup from "./Componenets/SubComponents/Signup";
import Login from "./Componenets/SubComponents/Login";
import FrontPage from "./Componenets/SubComponents/FrontPage";

const App = () => {
  return (
    <>
      <Routes>
        <Route path="/" element={<FrontPage/>} />
        <Route path="/about" element={<About />} />
        <Route path="/contact" element={<Contact />} />
        <Route path="/Signup" element={<Signup />} />
        <Route path="/Login" element={<Login />} />
        <Route path="/home" element={<Home/>}/>
      </Routes>
    </>
  );
};

export default App;
```

Login.js

```
  const handleSubmit = (event) => {
    event.preventDefault();

    if (validateForm()) {
      // Here you can make an API call to check the validity of the username and password
      // If they are valid, set isLoggedIn to true
      setIsLoggedIn(true);
    }
  };

  return (
    <div className="loginbox">
      <h1>Login</h1>
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label htmlFor="username">Username:</label>
          <input
            type="text"
            id="username"
            name="username"
            value={username}
            onChange={handleUsernameChange}
          />
          {errors.username && <span className="error">{errors.username}</span>}
        </div>
        <div className="form-group">
          <label htmlFor="password">Password:</label>
          <input
            type="password"
            id="password"
            name="password"
            value={password}
            onChange={handlePasswordChange}
          />
          {errors.password && <span className="error">{errors.password}</span>}
        </div>
        <button type="submit">Log In</button>
      </form>
```

```jsx
import React, { useState } from "react";
import "../../CSS/Login.css";

function LoginPage() {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [errors, setErrors] = useState({});
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const validateForm = () => {
    let errors = {};
    let isValid = true;

    if (!username.trim()) {
      errors.username = "Username is required";
      isValid = false;
    }

    if (!password.trim()) {
      errors.password = "Password is required";
      isValid = false;
    }

    setErrors(errors);

    return isValid;
  };

  const handleUsernameChange = (event) => {
    setUsername(event.target.value);
  };

  const handlePasswordChange = (event) => {
    setPassword(event.target.value);
  };
```

SignUp.js

```jsx
import React, { useState } from "react";
import validator from "validator";
import zxcvbn from "zxcvbn";
import "../../CSS/Signup.css";
import axios from "axios";

function SignupPage() {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [errors, setErrors] = useState({});

  const handleNameChange = (event) => {
    setName(event.target.value);
  };

  const handleEmailChange = (event) => {
    setEmail(event.target.value);
  };

  const handlePasswordChange = (event) => {
    setPassword(event.target.value);
  };

  const handleConfirmPasswordChange = (event) => {
    setConfirmPassword(event.target.value);
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    const errors = {};

    // Validate name
    if (!name) {
      errors.name = "Name is required";
    }

    // Validate email
    if (!email) {
      errors.email = "Email is required";
    } else if (!validator.isEmail(email)) {
      errors.email = "Invalid email address";
    }

    // Validate password
    if (!password) {
      errors.password = "Password is required";
    } else if (zxcvbn(password).score < 3) {
      errors.password = "Password is not strong enough";
    }

    // Validate confirm password
    if (!confirmPassword) {
      errors.confirmPassword = "Confirm password is required";
    } else if (password !== confirmPassword) {
      errors.confirmPassword = "Passwords do not match";
    }
```

```
    if (Object.keys(errors).length === 0) {
      // Here you can make an API call to submit the form data
      axios
        .post("/api/user", {
          name,
          email,
          password,
        })
        .then((response) => {
          console.log("User data saved:", response.data);
          // Clear form after successful submission
          setName("");
          setEmail("");
          setPassword("");
          setConfirmPassword("");
          setErrors({});
        })
        .catch((error) => {
          console.error("Error saving user data:", error);
        });
    } else {
      setErrors(errors);
    }
  };

  return (
    <div className="signupbox">
      <h1>Register</h1>
      <form onSubmit={handleSubmit}>
        <label>
          Name:
          <input
            type="text"
            value={name}
            onChange={handleNameChange}
            required
```

```
        {errors.email && <span className="error">{errors.email}</span>}
      </label>
      <br />
      <label>
        Password:
        <input
          type="password"
          value={password}
          onChange={handlePasswordChange}
          required
        />
        {errors.password && <span className="error">{errors.password}</span>}
      </label>
      <br />
      <label>
        Confirm Password:
        <input
          type="password"
          value={confirmPassword}
          onChange={handleConfirmPasswordChange}
          required
        />
        {errors.confirmPassword && (
          <span className="error">{errors.confirmPassword}</span>
        )}
      </label>
      <br />
      <button type="submit" disabled={Object.keys(errors).length > 0}>
        Sign Up
      </button>
    </form>
  </div>
 );
}
```

Homepage.js

```jsx
import React from 'react'
import Header from '../Header'
import Footer from '../Footer'
import ChatBox from './ChatBox'

function Home() {
  return (
    <>
        <Header/>
        <div className='chatcont'>
        <ChatBox />
        </div>
        <Footer/>

    </>
  )
}

export default Home
```

Chatbot.js

```jsx
import { useRef, useState } from "react";
import "../../CSS/ChatBox.css";
import axios from "axios";
var data = require("../../Assest/ques.json");

const YOU = "you";
const AI = "ai";

function ChatBox() {
  const [value, setValue] = useState("");

  const onChange = (event) => {
    setValue(event.target.value);
  };

  const onSearch = (searchTerm) => {
    setValue(searchTerm);
    // our api to fetch the search result
    console.log("search ", searchTerm);
  };

  const inputRef = useRef();
  const [qna, setQna] = useState([]);
  const [loading, setLoading] = useState(false);

  const updateQNA = (from, value) => {
    setQna((qna) => [...qna, { from, value }]);
  };
```

```
  const handleSend = () => {
    const question = inputRef.current.value;
    updateQNA(YOU, question);

    setLoading(true);
    axios
      .post("http://localhost:4000/chat", {
        question,
      })
      .then((response) => {
        updateQNA(AI, response.data.answer);
      })
      .finally(() => {
        setLoading(false);
      });
  };

  const renderContent = (qna) => {
    const value = qna.value;

    if (Array.isArray(value)) {
      return value.map((v) => <p className="message-text">{v}</p>);
    }

    return <p className="message-text">{value}</p>;
  };
```

```
  return (
    <main class="container">
      <div class="chats">
        {qna.map((qna) => {
          if (qna.from === YOU) {
            return (
              <div class="send chat">
                <p>{renderContent(qna)}</p>
                <img
                  src="https://cdn-icons-png.flaticon.com/512/924/924874.png"
                  alt=""
                  class="sendavtar"
                />
              </div>
            );
          }
          return (
            <div class="recieve chat">
              <img
                src="https://cdn-icons-png.flaticon.com/512/6819/6819755.png"
                alt=""
                class="reciveavtar"
              />
              <p>{renderContent(qna)}</p>
            </div>
          );
        })}

        {loading && (
          <div class="recieve chat">
            <img
              src="https://cdn-icons-png.flaticon.com/512/6819/6819755.png"
              alt=""
              class="reciveavtar"
            />
            <p>Typing...</p>
          </div>
        )}
```

FrontPage.js

```jsx
import React from 'react'
import Header from '../Header'
import Footer from '../Footer'
import "../../CSS/FrontPage.css"
import IntroImg from "../../Assest/intro.jpg"
import { Link } from 'react-router-dom'
export default function FrontPage() {
  return (
    <>
    <div className='navbarforhome'>
    <Header/>
    <div className='contents'>
    <h1 className='headingforhome'>Welcome to Advanced Era of AI</h1>
    <h1 className='subheadingforhome'>Why to burn yourself Out!</h1>
    <h1 className='subheadingforhome'>Get Answer of all your doubts using HelpyChat</h1>
    <div className='outerbox'>
    <Link className='linktohome' to={'/home'}>HelpyChat</Link>
    </div>
    </div>
    </div>

    {/* <div className="mask">
        <img className="into-img" src={IntroImg} alt="IntroImg" />
      </div> */}

    </>
  )
}
```

Backend Part

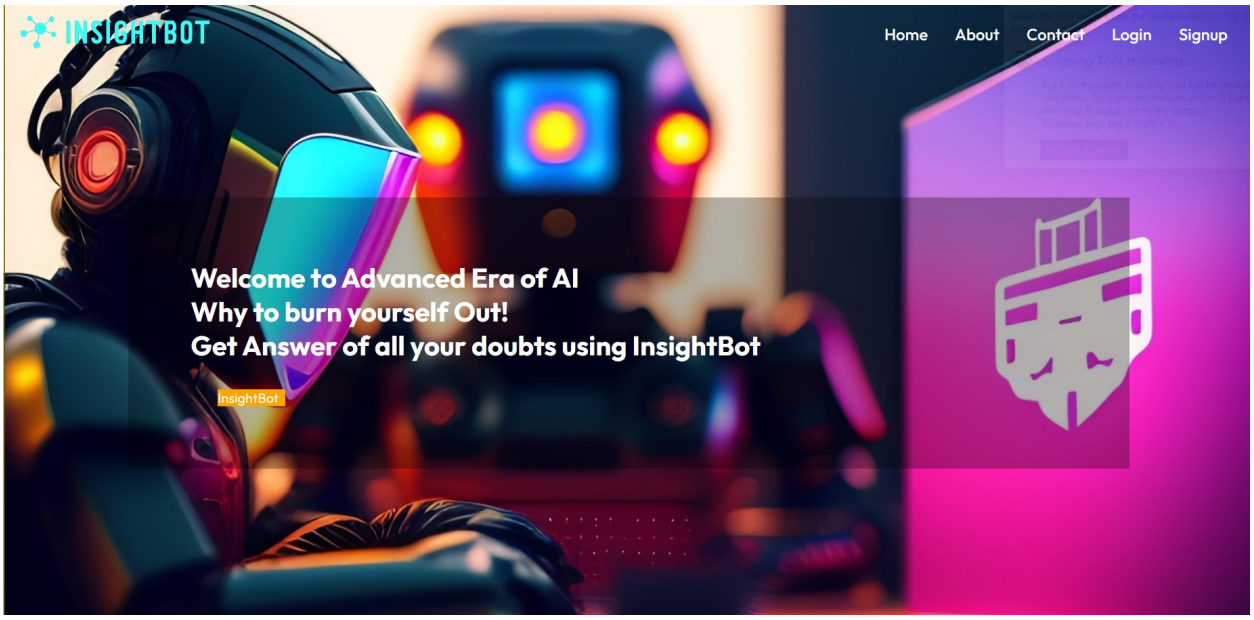
Server.js

```jsx
const OPENAI_API_KEY = "";  //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
//
//
//
const bcrypt = require("bcrypt");
const compare = bcrypt.compare;

const express = require("express");
const mongoose = require("mongoose");
const { Configuration, OpenAIApi } = require("openai");
const cors = require("cors");
const configuration = new Configuration({
  apiKey: OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);

const app = express();
app.use(express.json());
app.use(cors());

// connection to database
const mongoUrl = "mongodb://0.0.0.0:27017/HelpyChat";
mongoose
  .connect(mongoUrl, {
    useNewUrlParser: true,
  })
  .then(() => {
    console.log("Connected to database");
  })
  .catch((e) => console.log(e));

require("./userDetail");
const User = mongoose.model("UserInfo");
```

```
app.post("/Signup", async (req, res) => {
  const { name, email, password } = req.body;

  try {
    const oldUser = await User.findOne({ email });

    if (oldUser) {
      return res.json({ error: "User Exists" });
    }
    await User.create({
      name,
      email,
      password,
    });
    res.send({ status: "ok" });
  } catch (error) {
    res.send({ status: "error" });
  }
});

//User Login

app.post("/Login", async (req, res) => {
  const { email, password } = req.body;

  const user = await User.findOne({ email });
  if (user) {
    if (user.password === password) {
      return res.json("Login Ok");
    } else {
      return res.json({ error: "Password doesn't match!" });
    }
  } else {
    return res.json({ error: "User doesn't exist" });
  }
});
```

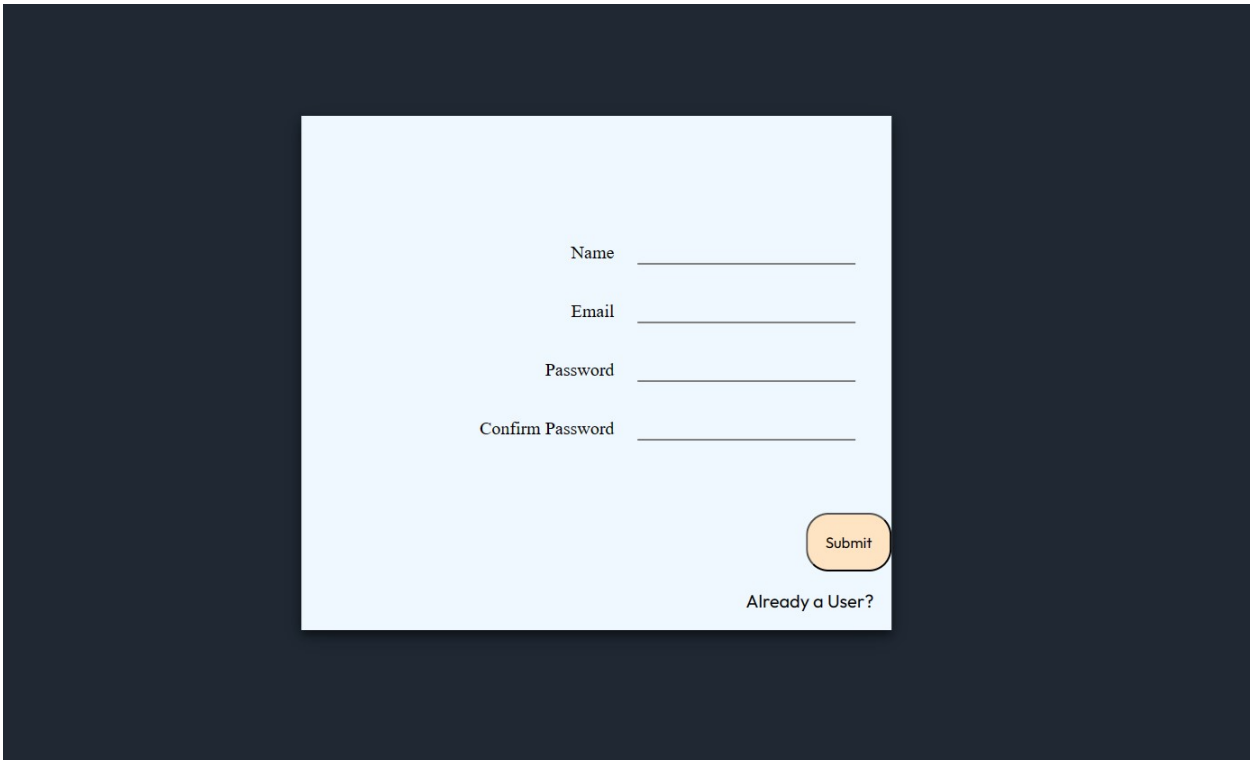
PREVIEWS

Home Page

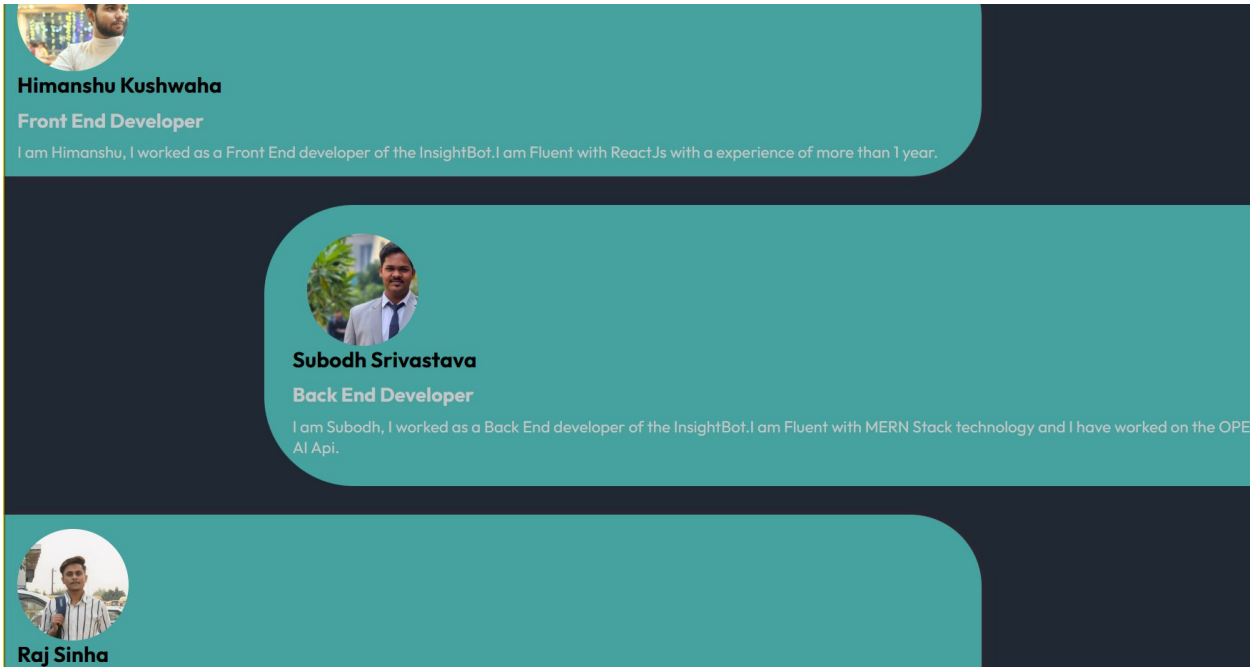


Login Page

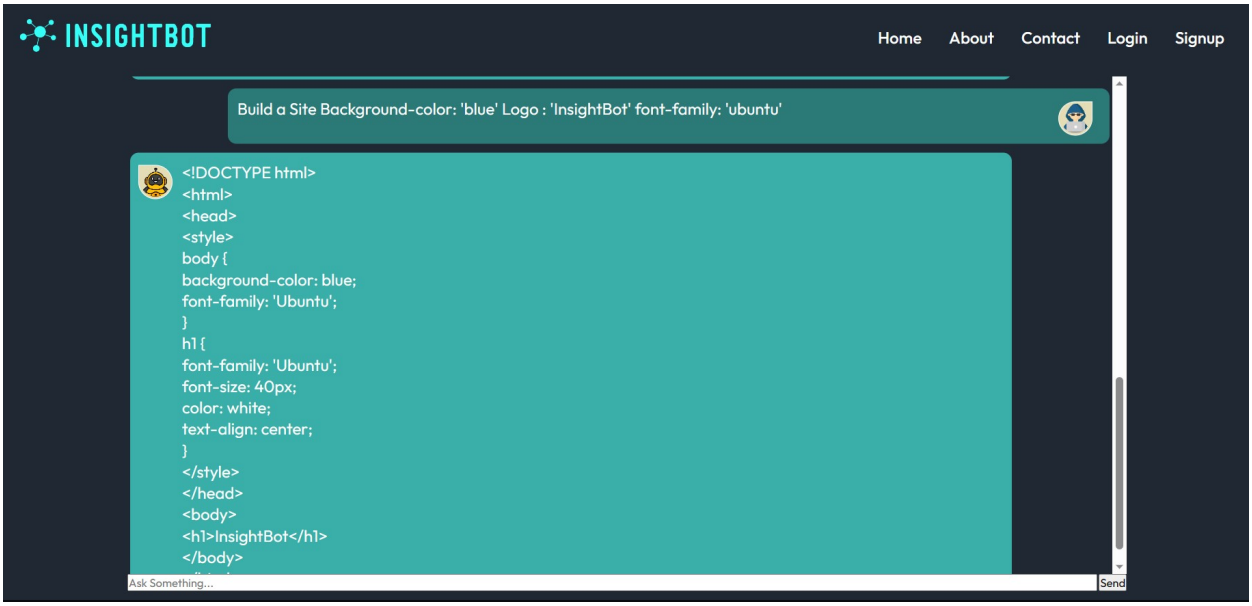
Signup Page

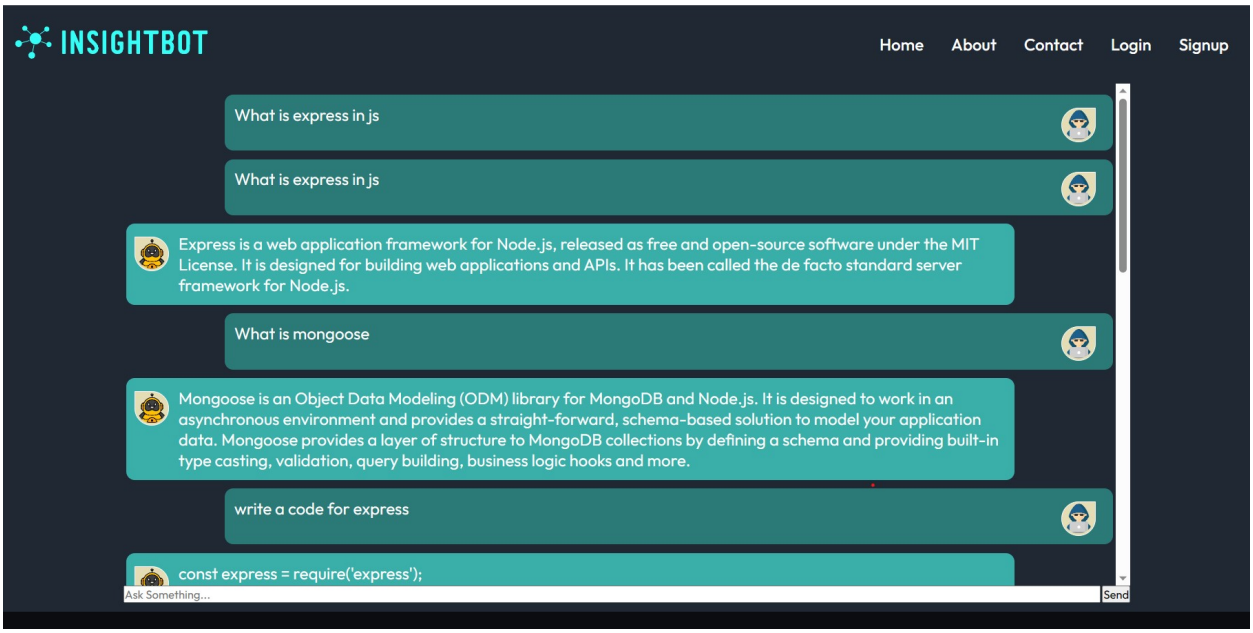


About Page



Conclusion


INSIGHTBOT
Home
About
Contact
Login
Signup
Build a Site Background-color: 'blue' Logo : 'InsightBot' font-family: 'ubuntu'
<!DOCTYPE html>
<html>
<head>
<style>
body {
background-color: blue;
font-family: 'Ubuntu';
}
h1 {
font-family: 'Ubuntu';
font-size: 40px;
color: white;
text-align: center;
}
</style>
</head>
<body>
<h1>InsightBot</h1>
</body>
Ask Something...
Send


INSIGHTBOT
Home
About
Contact
Login
Signup
What is express in js
What is express in js
Express is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.
What is mongoose
Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It is designed to work in an asynchronous environment and provides a straight-forward, schema-based solution to model your application data. Mongoose provides a layer of structure to MongoDB collections by defining a schema and providing built-in type casting, validation, query building, business logic hooks and more.
write a code for express
const express = require('express');
Ask Something...
Send

References

Websites: https://www.w3schools.com/

https://www.geeksforgeeks.org/