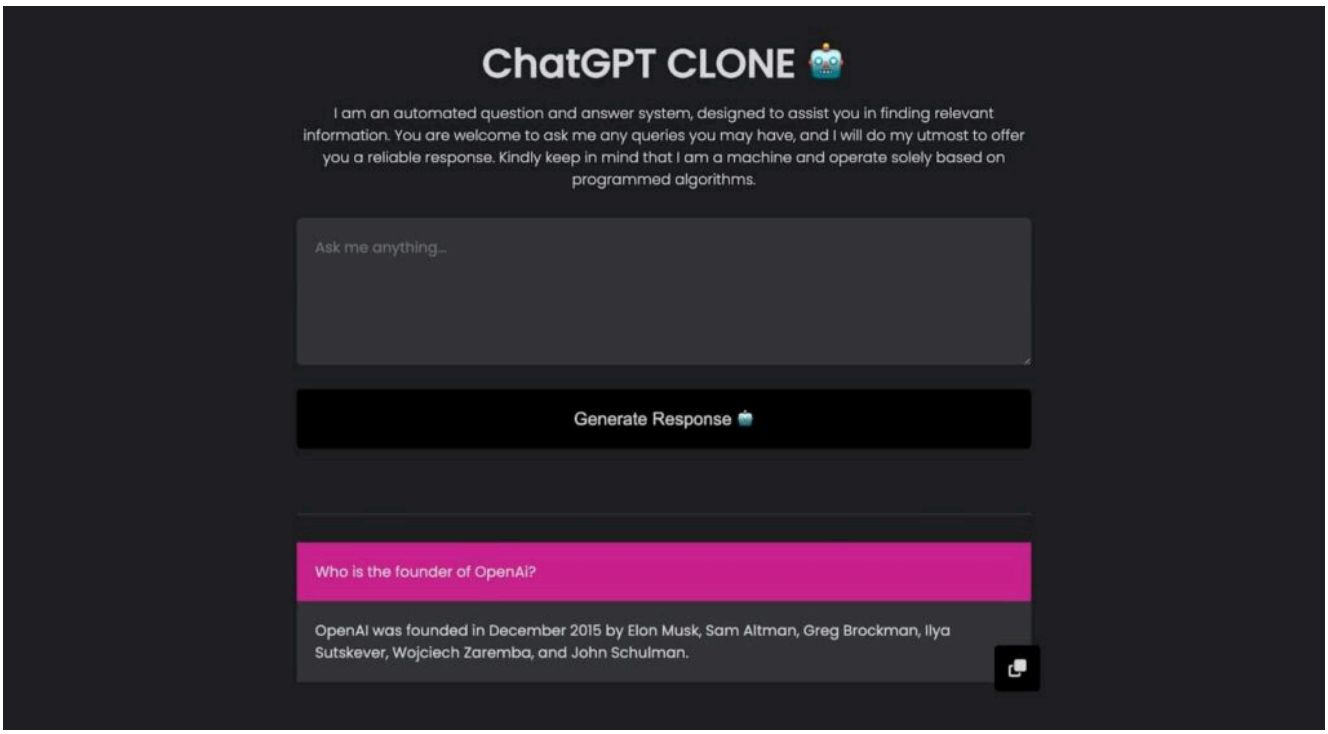
Как создать и развернуть приложение-клон ChatGPT с помощью React и OpenAI API

11.05.2023

Поскольку использование чат-ботов и виртуальных помощников продолжает расти, многие компании и разработчики ищут способы создания собственных чат-ботов с поддержкой ИИ. ChatGPT – один из таких чат-ботов, созданный компанией OpenAI, который способен вести беседу в человеческом стиле и отвечать на широкий спектр вопросов.

Что вы будете создавать

В этом руководстве вы узнаете, как создать приложение-клон ChatGPT с помощью React и API OpenAI. Если вы хотите попробовать свои силы в интересном и увлекательном проекте на выходных, это отличная возможность погрузиться в React и OpenAI.



Если вы хотите более детально изучить этот проект, вы можете

получить доступ к его репозиторию **на GitHub**. В качестве альтернативы, используя этот шаблон стартового проекта, вы можете выбрать Use this template > Create a new repository – это скопирует код стартового проекта в новый репозиторий. Этот стартовый проект содержит такие фундаментальные элементы, как стили, ссылку на Font Awesome CDN, пакет OpenAi и базовую структуру, которые помогут вам начать работу.

Информация
Бесплатный кредит на использование API OpenAI ограничен $18. Если вы тестируете это демо-приложение и оно перестает работать, это означает, что кредит мог закончиться. Чтобы продолжать пользоваться API OpenAI, вам необходимо обновить тарифный план.

Требования/Пререквизиты

Этот учебник разработан для того, чтобы вы могли “следовать за ним”. Поэтому рекомендуется, чтобы вы обладали следующими знаниями и умениями, чтобы с легкостью писать код рядом:

- Фундаментальное понимание HTML, CSS и JavaScript
- Некоторое знакомство с React
- Node.js и npm (Node Package Manager) или yarn, установленные на вашем компьютере

Что такое API OpenAI?

OpenAI API – это облачная платформа, предоставляющая разработчикам доступ к языковым моделям OpenAI, таким как GPT-3, через API. Это позволяет разработчикам добавлять функции обработки естественного языка, такие как завершение текста, анализ настроения, обобщение и перевод, в свои приложения без разработки и обучения своих моделей. Чтобы использовать API OpenAI, разработчики должны создать учетную запись на сайте OpenAI и получить ключ API. Ключ API

используется для аутентификации запросов API и отслеживания использования. После получения ключа API разработчики могут использовать API для отправки текста в языковую модель и получения ответов.

Почему именно React?

React — это популярная библиотека JavaScript для построения пользовательских интерфейсов. По данным опроса разработчиков Stack Overflow за 2022 год, это вторая по распространенности веб-технология, занимающая 42,62% рынка. React позволяет разработчикам создавать декларативные компоненты, представляющие различные части пользовательского интерфейса. Эти компоненты определяются с помощью синтаксиса, называемого JSX, который представляет собой комбинацию JavaScript и HTML. Благодаря большой экосистеме библиотек и наборов компонентов, разработчики могут легко работать и интегрировать API, такие как OpenAI API, для создания сложных интерфейсов чата, что делает его отличным выбором для создания клона приложения ChatGPT.

Как настроить среду разработки React

Лучший способ установить React или создать проект React — это установить его с помощью create-react-app. Одним из предварительных условий является наличие установленного Node.js на вашей машине. Чтобы убедиться, что у вас установлен Node, выполните следующую команду в терминале.

```
node -v
```

Если он выводит версию, значит, она существует. Чтобы использовать npx, вам нужно убедиться, что версия вашего Node не ниже v14.0.0, а версия NPM — не ниже v5.6; в противном случае вам придется обновить его, выполнив **npm update -g**. Разобравшись с npm, вы можете установить проект React, выполнив команду ниже:

```
npx create-react-app chatgpt-clone
```

Примечание: “chatgpt-clone” — это имя приложения, которое мы создаем, но вы можете изменить его на любое имя по вашему выбору. Процесс установки может занять несколько минут. После успешной установки вы можете перейти в каталог и установить пакет Node.js OpenAI, который обеспечивает удобный доступ к API OpenAI из Node.js с помощью команды ниже:

```
npm install openai
```

Теперь вы можете запустить **npm start** и увидеть, что ваше приложение работает на **localhost:3000**.

Когда проект React создается с помощью команды **create-react-app**, он автоматически создает структуру папок. Основной папкой, которая вас волнует, является папка src, в которой происходит разработка. По умолчанию эта папка содержит множество файлов, но вас должны волновать только файлы **App.js**, **index.js** и **index.css**.

- **App.js:** Файл App.js является основным компонентом в приложении React. Обычно он представляет собой компонент верхнего уровня, который отображает все остальные компоненты в приложении.
- **index.js:** Этот файл является точкой входа вашего React-приложения. Он является первым файлом, загружаемым при открытии приложения, и отвечает за отображение компонента App.js в браузере.
- **index.css:** Этот файл отвечает за определение общего стиля и макета вашего React-приложения.

Как создать клон ChatGPT с помощью React и OpenAI API

Клон приложения ChatGPT будет состоять из двух компонентов, чтобы приложение было проще понять и поддерживать. Этими двумя

компонентами являются:

1. **Секция формы:** Этот компонент включает в себя текстовое поле и кнопку для взаимодействия пользователей с чатботом.
2. **Секция ответов:** Вопросы и соответствующие ответы будут храниться в массиве и отображаться в этой секции. Вы будете просматривать массив в хронологическом порядке, показывая сначала последние ответы.

Настройка приложения-клона ChatGPT

В этом руководстве мы начнем с создания интерфейса приложения, а затем вы сможете реализовать функциональность, чтобы ваше приложение взаимодействовало с API OpenAI. Начните с создания двух компонентов, которые вы будете использовать в этом учебнике. Для правильной организации вы создадите папку components в папке src, где будут храниться все компоненты.

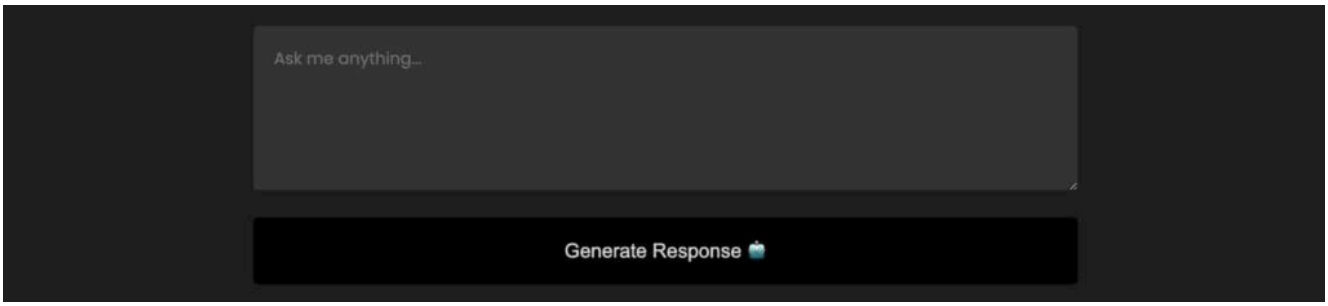
Компонент раздела формы

Это простая форма, состоящая из текстовой области и кнопки отправки.

```
// components/FormSection.jsx

const FormSection = () => {

    return (
        <div className="form-section">
            <textarea
                rows="5"
                className="form-control"
                placeholder="Ask me anything..."
            ></textarea>
            <button className="btn">
                Generate Response
            </button>
        </div>
```

```
    )
}

export default FormSection;
```

Вот как должна выглядеть форма, когда вы импортируете ее в свой файл App.js:

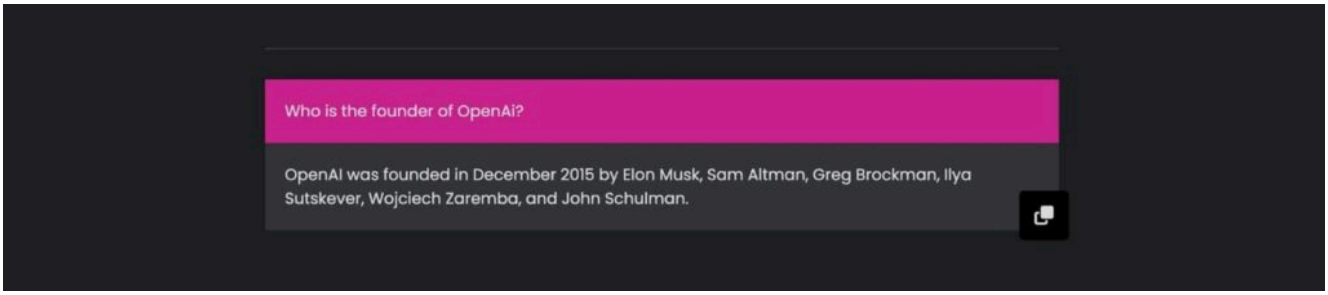


Информация

В этом руководстве основное внимание уделяется созданию и развертыванию вашего приложения. Поэтому вы можете скопировать стили из файла src/index.css в свой собственный проект, чтобы получить тот же результат/приложение.

Компонент раздела “Ответы”

В этом разделе отображаются все вопросы и ответы. Вот как будет выглядеть этот раздел, когда вы импортируете его в файл App.js.



Вы будете получать эти вопросы и ответы из массива и зацикливать их, чтобы облегчить чтение и сопровождение вашего кода.

```
// components/AnswerSection.jsx

const AnswerSection = () => {
    return (
```

```
        <>
            <hr className="hr-line" />
            <div className="answer-container">
                <div className="answer-section">
                    <p className="question">Who is the founder
of OpenAi?</p>
                     <p className="answer">OpenAI was founded
in December 2015 by Elon Musk, Sam Altman, Greg Brockman, Ilya
Sutskever, Wojciech Zaremba, and John Schulman.</p>
                    <div className="copy-icon">
                        <i className="fa-solid fa-copy"></i>
                    </div>
                </div>
            </div>
        </>
    )
}

export default AnswerSection;
```

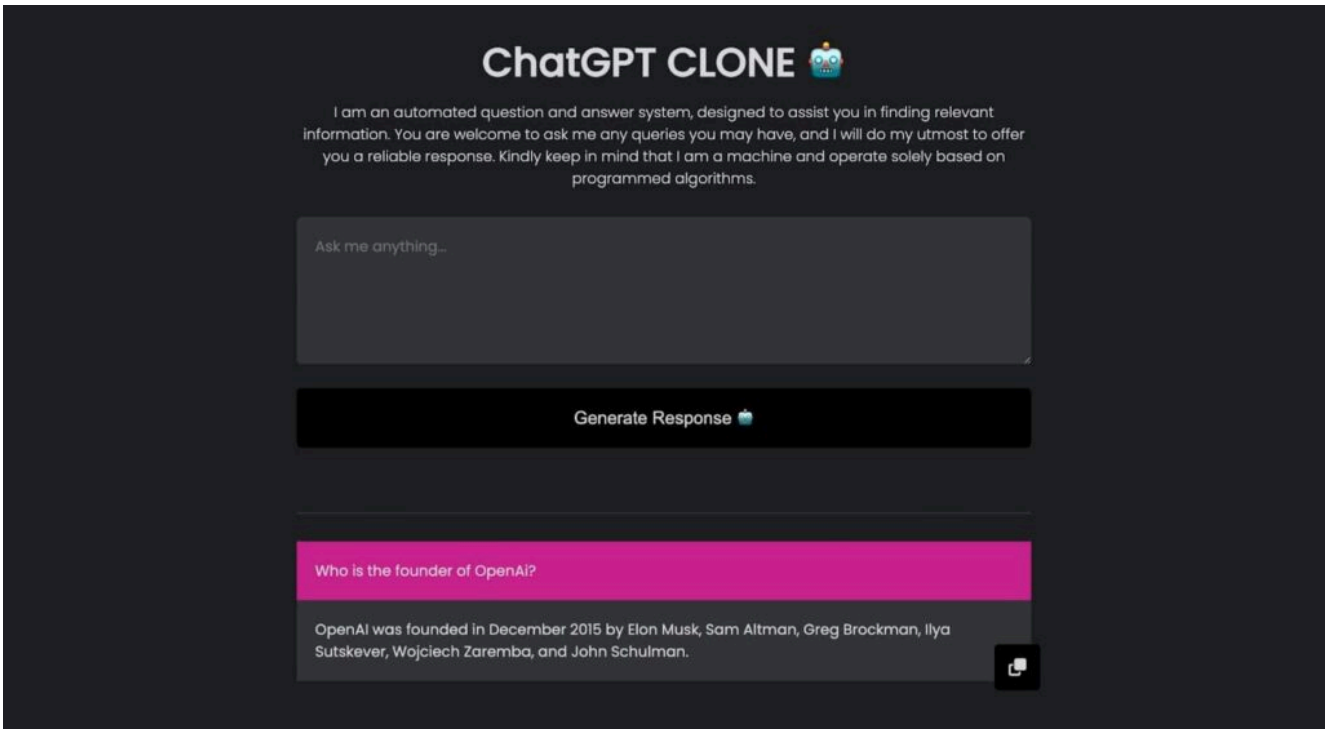
Главная страница

Теперь у вас созданы оба компонента, но при запуске приложения ничего не появится, потому что их нужно импортировать в файл App.js. В этом приложении вы не будете применять какую-либо форму маршрутизации, то есть файл App.js будет служить домашним компонентом/страницей вашего приложения. Вы можете добавить некоторое содержимое, например, название и описание вашего приложения, перед импортом компонентов.

```
// App.js

import FormSection from './components/FormSection';
import AnswerSection from './components/AnswerSection';

const App = () => {
    return (
        <div>
            <div className="header-section">
                <h1>ChatGPT CLONE </h1>
                <p>
```

```
                    I am an automated question and answer
system, designed to assist you
                    in finding relevant information. You are
welcome to ask me any queries
                    you may have, and I will do my utmost to
offer you a reliable
                    response. Kindly keep in mind that I am a
machine and operate solely
                    based on programmed algorithms.
                </p>
            </div>

            <FormSection />
            <AnswerSection />
        </div>
    );
};

export default App;
```

В приведенном выше коде два компонента импортируются и добавляются в приложение. Когда вы запустите свое приложение, вот как будет выглядеть ваше приложение:



Добавление функциональности и интеграция API OpenAI

Теперь у вас есть пользовательский интерфейс вашего приложения. Следующий шаг — сделать приложение функциональным, чтобы оно могло взаимодействовать с OpenAI API и получать ответы. Во-первых, вам нужно получить значение из формы при отправке, поскольку оно будет использоваться для запроса к OpenAI API.

Получение данных из формы

В React лучшим способом хранения и обновления данных является использование состояний. В функциональных компонентах для работы с состояниями используется хук useState(). Вы можете создать состояние, присвоить ему значение из вашей формы и обновлять его всякий раз, когда значение изменяется. Давайте начнем с импорта крючка useState() в компонент FormSection.jsx, а затем создадим состояние для хранения и обновления newQuestions.

```
// components/FormSection.jsx

import { useState } from 'react';

const FormSection = ({ generateResponse }) => {
    const [newQuestion, setNewQuestion] = useState('');

    return (
        // Form to submit a new question
    )
}

export default FormSection;
```

Далее вы можете присвоить значение поля textarea состоянию и создать событие onChange() для обновления состояния при каждом изменении значения ввода:

```
<textarea
```

```
    rows="5"
    className="form-control"
    placeholder="Ask me anything..."
    value={newQuestion}
    onChange={(e) => setNewQuestion(e.target.value)}
>
</textarea>
```

Наконец, можно создать событие onClick(), чтобы загружать функцию при нажатии на кнопку отправки. Этот метод будет создан в файле App.js и передан в качестве props в компонент FormSection.jsx со значениями newQuestion и setNewQuestion в качестве аргументов.

```
<button      className="btn"      onClick={()      =>
generateResponse(newQuestion, setNewQuestion)}>
    Generate Response </button>
```

Теперь вы создали состояние для хранения и обновления значения формы, добавили метод, который передается в качестве props из файла App.js, и обработали событие click. Вот как будет выглядеть окончательный вариант кода:

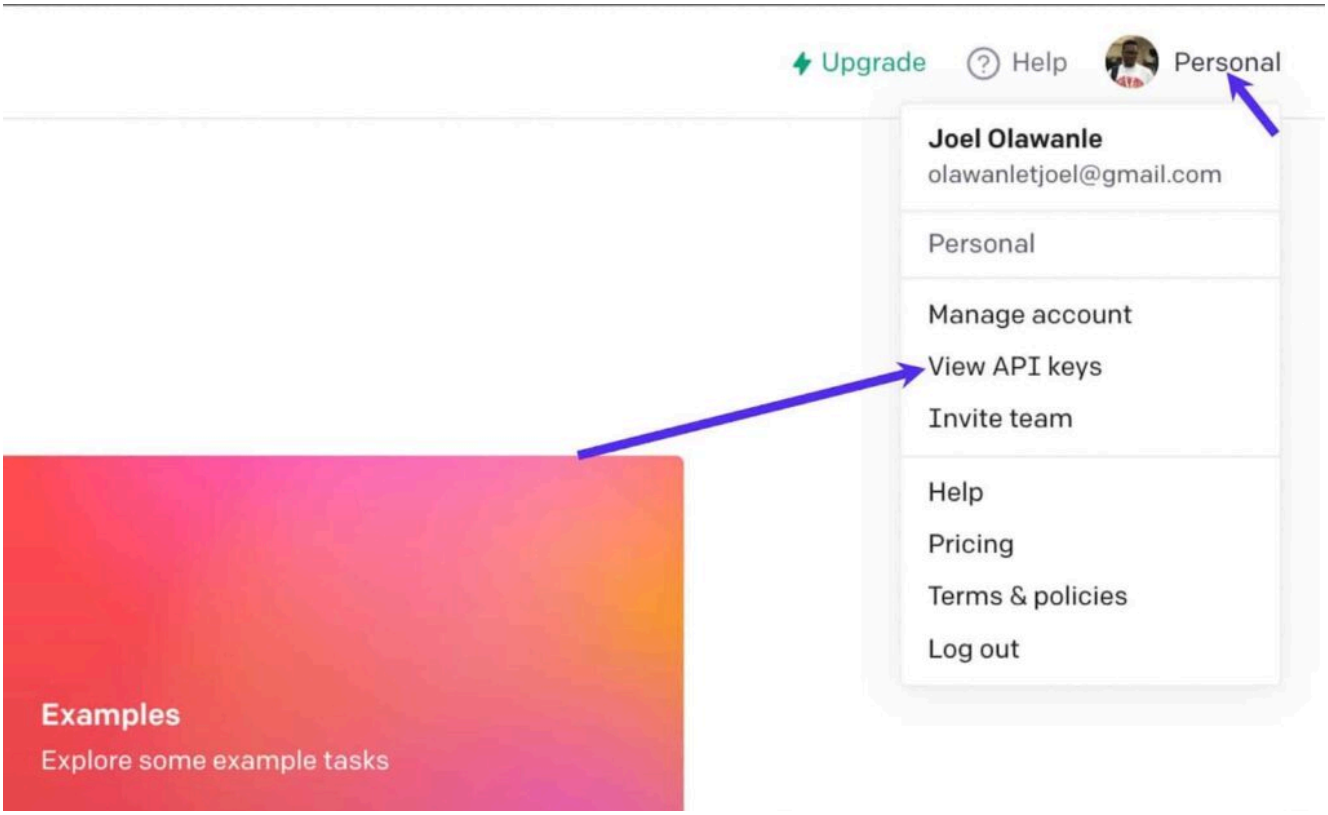
```
// components/FormSection.jsx

import { useState } from 'react';

const FormSection = ({ generateResponse }) => {
    const [newQuestion, setNewQuestion] = useState('');

    return (
        <div className="form-section">
            <textarea
                rows="5"
                className="form-control"
                placeholder="Ask me anything..."
                value={newQuestion}
                                    onChange={(e)  =>
setNewQuestion(e.target.value)}
            ></textarea>
                <button className="btn" onClick={() =>
generateResponse(newQuestion, setNewQuestion)}>
```

```
                Generate Response
            </button>
        </div>
    )
}

export default FormSection;
```

Следующим шагом будет создание метода в файле App.js для обработки всего процесса взаимодействия с API OpenAI.

Взаимодействие с OpenAI API

Чтобы взаимодействовать с OpenAI API и получать API-ключи в приложении React, необходимо создать учетную запись OpenAI API. Вы можете зарегистрировать учетную запись на сайте OpenAI, используя свой аккаунт google или электронную почту. Чтобы сгенерировать API-ключ, нажмите Personal в правом верхнем углу сайта; появится несколько опций; нажмите View API keys.



Нажмите кнопку Создать новый секретный ключ, скопируйте ключ куда-нибудь, так как вы будете использовать его в этом приложении для взаимодействия с OpenAI. Теперь вы можете

приступить к инициализации OpenAI, импортировав пакет openai (который вы уже установили) вместе с методом конфигурации. Затем создайте конфигурацию со сгенерированным ключом и используйте ее для инициализации OpenAI.

```
// src/App.js

import { Configuration, OpenAIApi } from 'openai';

import FormSection from './components/FormSection';
import AnswerSection from './components/AnswerSection';

const App = () => {
    const configuration = new Configuration({
        apiKey: process.env.REACT_APP_OPENAI_API_KEY,
    });

    const openai = new OpenAIApi(configuration);

    return (
        // Render FormSection and AnswerSection
    );
};

export default App;
```

В приведенном выше коде ключ API OpenAI хранится как переменная окружения в файле .env. Вы можете создать файл .env в корневой папке вашего приложения и хранить ключ в переменной REACT_APP_OPENAI_API_KEY.

```
// .env
REACT_APP_OPENAI_API_KEY = sk-xxxxxxxxxx…
```

Теперь вы можете перейти к созданию метода generateResponse в файле App.js и передать в него два параметра, ожидаемых от формы, которую вы уже создали для обработки запроса и получения ответа от API.

```
// src/App.js
```

```
import FormSection from './components/FormSection';
import AnswerSection from './components/AnswerSection';

const App = () => {
    const generateResponse = (newQuestion, setNewQuestion) =>
{
        // Set up OpenAI API and handle response
    };

    return (
        // Render FormSection and AnswerSection
    );
};

export default App;
```

Теперь вы можете отправить запрос в OpenAI API. API OpenAI может выполнять множество операций, таких как вопросы и ответы (Q&A), исправление грамматики, перевод и многое другое. Для каждой из этих операций существуют свои параметры. Например, значение движка для Q&A — text-davinci-00, а для SQL translate — code-davinci-002. Не стесняйтесь проверить документацию по примерам OpenAI для различных примеров и их опций.

В этом руководстве мы работаем только с Q&A, вот как выглядит опция:

```
{
  model: "text-davinci-003",
  prompt: "Who is Obama?",
  temperature: 0,
  max_tokens: 100,
  top_p: 1,
  frequency_penalty: 0.0,
  presence_penalty: 0.0,
  stop: [""],
}
```

Примечание: Я изменил значение подсказки.

Подсказка — это вопрос, который отправляется из формы. Это

означает, что вам нужно получить его из ввода формы, который вы передаете в метод generateResponse в качестве параметра. Для этого нужно определить опции, а затем с помощью оператора spread создать полную опцию, включающую подсказку:

```
// src/App.js

import { Configuration, OpenAIApi } from 'openai';
import FormSection from './components/FormSection';
import AnswerSection from './components/AnswerSection';

const App = () => {
    const configuration = new Configuration({
        apiKey: process.env.REACT_APP_OPENAI_API_KEY,
    });

    const openai = new OpenAIApi(configuration);

       const generateResponse = async (newQuestion, setNewQuestion) => {
        let options = {
            model: 'text-davinci-003',
            temperature: 0,
            max_tokens: 100,
            top_p: 1,
            frequency_penalty: 0.0,
            presence_penalty: 0.0,
            stop: ['/'],
        };

        let completeOptions = {
            ...options,
            prompt: newQuestion,
        };

    };

    return (
         // Render FormSection and AnswerSection
    );
};
```

```
export default App;
```

На этом этапе остается только отправить запрос через метод createCompletion в OpenAI, чтобы получить ответ.

```
// src/App.js

import { Configuration, OpenAIApi } from 'openai';
import FormSection from './components/FormSection';
import AnswerSection from './components/AnswerSection';

import { useState } from 'react';

const App = () => {
    const configuration = new Configuration({
        apiKey: process.env.REACT_APP_OPENAI_API_KEY,
    });

    const openai = new OpenAIApi(configuration);

    const [storedValues, setStoredValues] = useState([]);

      const generateResponse = async (newQuestion, setNewQuestion) => {
        let options = {
            model: 'text-davinci-003',
            temperature: 0,
            max_tokens: 100,
            top_p: 1,
            frequency_penalty: 0.0,
            presence_penalty: 0.0,
            stop: ['/'],
        };

        let completeOptions = {
            ...options,
            prompt: newQuestion,
        };

                        const response = await openai.createCompletion(completeOptions);
```

```
        console.log(response.data.choices[0].text);
    };

    return (
        // Render FormSection and AnswerSection
    );
};

export default App;
```

В приведенном выше коде текст ответа будет выведен на консоль. Не стесняйтесь протестировать свое приложение, задав любой вопрос. Последним шагом будет создание состояния, в котором будет храниться массив вопросов и ответов, а затем отправка этого массива в качестве prop в компонент AnswerSection. Вот как будет выглядеть окончательный код App.js:

```
// src/App.js
import { Configuration, OpenAIApi } from 'openai';

import FormSection from './components/FormSection';
import AnswerSection from './components/AnswerSection';

import { useState } from 'react';

const App = () => {
    const configuration = new Configuration({
        apiKey: process.env.REACT_APP_OPENAI_API_KEY,
    });

    const openai = new OpenAIApi(configuration);

    const [storedValues, setStoredValues] = useState([]);

       const generateResponse = async (newQuestion, setNewQuestion) => {
        let options = {
            model: 'text-davinci-003',
            temperature: 0,
            max_tokens: 100,
            top_p: 1,
```

```
            frequency_penalty: 0.0,
            presence_penalty: 0.0,
            stop: ['/'],
        };

        let completeOptions = {
            ...options,
            prompt: newQuestion,
        };

                            const   response   =   await
openai.createCompletion(completeOptions);

        if (response.data.choices) {
            setStoredValues([
                {
                    question: newQuestion,
                    answer: response.data.choices[0].text,
                },
                ...storedValues,
            ]);
            setNewQuestion('');
        }
    };

    return (
        <div>
            <div className="header-section">
                <h1>ChatGPT CLONE</h1>
                    <p>
                         I am an automated question and answer
system, designed to assist you
                          in finding relevant information. You
are welcome to ask me any
                        queries you may have, and I will do my
utmost to offer you a
                        reliable response. Kindly keep in mind
that I am a machine and
                         operate solely based on programmed
algorithms.
                    </p>
```

```
            </div>

             <FormSection generateResponse={generateResponse}
/>

            <AnswerSection storedValues={storedValues} />
        </div>
    );
};

export default App;
```

Теперь вы можете отредактировать компонент AnswerSection, чтобы он получал значение props из App.js и использовать метод JavaScript Map() для просмотра массива storedValues:

```
// components/AnswerSection.jsx

const AnswerSection = ({ storedValues }) => {
    return (
        <>
            <hr className="hr-line" />
            <div className="answer-container">
                {storedValues.map((value, index) => {
                    return (
                           <div className="answer-section"
key={index}>
                                                                <p
className="question">{value.question}</p>
                                                                <p
className="answer">{value.answer}</p>
                                <div className="copy-icon">
                                      <i className="fa-solid fa-
copy"></i>
                                </div>
                            </div>
                    );
                })}
            </div>
        </>
    )
}
```

```
export default AnswerSection;
```

Когда вы запустите свое приложение и протестируете его, задавая вопросы, ответ отобразится ниже. Но вы заметите, что кнопка копирования не работает. Вам нужно добавить событие onClick() для кнопки, чтобы оно вызывало метод для обработки функциональности. Для этого можно использовать метод navigator.clipboard.writeText(). Вот как теперь будет выглядеть компонент AnswerSection:

```
// components/AnswerSection.jsx

const AnswerSection = ({ storedValues }) => {
    const copyText = (text) => {
        navigator.clipboard.writeText(text);
    };

    return (
        <>
            <hr className="hr-line" />
            <div className="answer-container">
                {storedValues.map((value, index) => {
                    return (
                          <div className="answer-section"
key={index}>
                                                          <p
className="question">{value.question}</p>
                                                          <p
className="answer">{value.answer}</p>
                            <div
                                 className="copy-icon"
                                             onClick={() =>
copyText(value.answer)}
                            >
                                  <i className="fa-solid fa-
copy"></i>
                            </div>
                        </div>
                    );
                })}
            </div>
```

```
        </>
    )
}

export default AnswerSection;
```

Когда вы запустите свое приложение, ваше приложение-клон ChatGPT будет работать безупречно. Теперь вы можете развернуть свое приложение, чтобы получить доступ к нему в Интернете и поделиться им с друзьями.

Заключение

API OpenAI можно использовать для создания широкого спектра потенциальных приложений, от поддержки клиентов и персональных помощников до языкового перевода и создания контента. В этом руководстве вы узнали, как создать клон приложения ChatGPT с помощью React и OpenAI. Вы можете интегрировать это приложение/функцию в другие приложения, чтобы предоставить пользователям человекоподобный разговорный опыт. С помощью API OpenAI можно многое сделать и улучшить это приложение-клон. Например, можно реализовать локальное хранение, чтобы предыдущие вопросы и ответы не исчезали даже после обновления браузера.