AI GENERATED SMART CONTRACT

A PROJECT REPORT

Submitted by

SHAILENDER KUMAR MAURIYA (19030141CSE075)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the Supervision of

Dr. ANOOP KUMAR SRIVASTAVA
Professor
Department of CSE

ALLIANCE COLLEGE OF ENGINEERING AND DESIGN
Department Of Computer Science and Engineering

ALLIANCE UNIVERSITY

BANGALORE 562106

JUNE-2023

Department Of Computer Science & Engineering
Alliance College of Engineering & Design
Alliance University
Chikkahagede Cross Chandapura-Anekal Main Road
Bangalore-562106

CERTIFICATE

This is to certify that the project work entitled **"AI Generated Smart Contract"** is the bonafide work done by **Mr. SHAILENDER KUMAR MAURIYA (**19030141CSE075) submitted in partial fulfilment of the requirements for the award of the degree **Bachelor of Technology** in **Computer Science and Engineering** during the year 2022-2023.

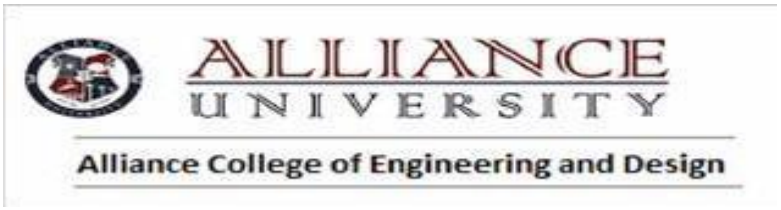
Dr. Anoop Kumar Srivastava
Supervisor

Dr. Abraham George
Head of the department

External Examiners:

Name: Signature:

Name: Signature:



Declaration

This is to declare that the report titled "**AI Generated Smart Contract**" has been made for the partial fulfilment of the Course Bachelor of Technology in Computer Science and Engineering, under the Supervision of **Dr. Anoop Kumar Srivastava.** We confirm that this report truly represents our work undertaken as a part of our project work. This work is not a replication of work done previously by any other person. We also confirm that the contents of the report and the views contained therein have been discussed and deliberated with the faculty guide.

NAME	REG. NO	SIGNATURE
SHAILENDER KUMAR MAURIYA	19030141CSE075	

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We are very thankful to our guide and Project Coordinator **Dr. Anoop Kumar Srivastava,** Department of Computer Science and Engineering for his sustained inspiring guidance and cooperation throughout the process of this project. His wise counsel and valuable suggestions are invaluable.

We would like to thank **Dr. Abraham George,** Head of the department and **Dr. Reeba Korah**, Dean fortheir encouragement and cooperation at various levels of Project.

We avail this opportunity to express my deep sense of gratitude and hearty thanks to the Managementof Alliance University, for providing world class infrastructure, congenial atmosphere and encouragement.

We express my deep sense of gratitude and thanks to the teaching and non-teaching staff at our department who stood with me during the project and helped me to make it a successful venture.

We place highest regards to my parents, my friends and well-wishers who helped a lot in making thereport of this project.

SHAILENDER KUMAR MAURIYA

Abstract

This section provides a high-level overview of an AI-generated smart contract, highlighting its key features and benefits. The AI-generated smart contract automates and streamlines contractual agreements by leveraging artificial intelligence and blockchain technology. It provides a dependable and efficient solution for securely executing complex business transactions by combining the power of machine learning and cryptography.

Traditional contractual agreements are being transformed by AI-generated smart contracts, which incorporate advanced technologies to improve efficiency, transparency, and trust. It is intended to automate and enforce agreement terms without the use of intermediaries such as lawyers or third-party entities.

The smart contract analyses and understands the context and requirements of the agreement using machine learning algorithms, ensuring accurate execution. It can interpret natural language inputs, identify potential risks or ambiguities, and recommend changes to improve the terms for all parties involved.

The smart contract ensures immutability, transparency, and security by integrating with blockchain technology. The terms and conditions of the agreement are stored on a distributed ledger, eliminating the possibility of manipulation or unauthorized changes. The blockchain's decentralized nature ensures that all participants have access to a synchronized version of the contract, reducing disputes and allowing for efficient resolution.

The AI-generated smart contract significantly reduces the costs, time, and potential human errors associated with traditional contract management by eliminating the need for intermediaries. It allows for faster agreement execution, simplifies auditing, and ensures adherence to predefined rules and regulations.

Overall, the AI-generated smart contract is a powerful tool for businesses and individuals, enabling secure and efficient collaboration while reducing the need for manual intervention. Its combination of artificial intelligence and blockchain technology opens up new avenues for streamlining contractual agreements, increasing trust, and driving innovation across multiple industries.

Table of Contents

List of abbreviations

AI – Artificial Intelligence

ML – Machine Learning

HTML – Hyper Text Markup Language

CSS – Cascade Style Sheets

RL – Reinforcement Learning

NLP – Natural Language Processing

VS code – Visual Studio Code

List of Figures

Chapter 1

1. Introduction

With the introduction of smart contracts created by artificial intelligence, the world of contract management is undergoing a seismic change. These cutting-edge digital contracts have the potential to completely transform how contracts are produced, carried out, and enforced across industries thanks to the integration of artificial intelligence (AI) and blockchain technology.

Smart contracts are computer programs that execute predefined actions when certain conditions are met. They rely heavily on blockchain technology to ensure transparency, immutability, and security. Traditional smart contracts, on the other hand, have limitations in terms of complexity and adaptability. This project investigates incorporating AI into smart contracts to address these limitations and open up new possibilities.

AI-generated smart contracts use cutting-edge algorithms and machine learning skills to automatically construct agreements that are enforceable in court. These intelligent algorithms can construct thorough and tailored contracts more quickly and accurately than conventional manual techniques by evaluating enormous amounts of data and deciphering difficult contractual needs.

The addition of blockchain technology expands the capabilities of smart contracts. These contracts give unrivalled transparency, security, and traceability by using the decentralized and immutable nature of blockchain. The removal of intermediaries and the automation of contract execution speed processes, lower costs, and increase confidence among parties involved.

The advantages of AI-generated smart contracts are numerous. They reduce human error, improve contract efficiency, and ensure legal and regulatory compliance. Businesses can save time and resources while reducing disputes by streamlining and automating contract processes.

They could transform supply chain management, expedite financial transactions, and ensure legal and regulatory compliance. AI-generated smart contracts offer a disruptive solution that can improve the way business is

conducted by eliminating time-consuming and error-prone manual methods.

AI-generated smart contracts represent a fundamental leap in contractual automation as firms try to optimize their operations. These contracts, with their ability to save time, cut costs, and enhance accuracy, are poised to change industries, open new opportunities, and set new standards for efficiency and openness in contract management.

This project's major findings include identifying key challenges and limitations of traditional smart contracts, exploring various AI techniques such as natural language processing and machine learning for smart contract automation, and developing a prototype AI-generated smart contract system. The project's findings highlight the potential of AI in improving the efficiency, reliability, and flexibility of smart contract systems. There are also recommendations for future research and development in this area.

Automation and decentralization are transforming the way we conduct transactions and establish trust in numerous industries in the digital age. Smart contracts, which are code-based self-executing agreements, have emerged as a useful tool for automating processes, eliminating intermediaries, and ensuring transparency and efficiency. However, manually developing smart contracts is time-consuming, error-prone, and necessitates specialist knowledge of both contract law and programming. This is where Artificial Intelligence (AI) comes in, with the potential to revolutionize smart contract production and deployment using AI-generated smart contracts.

1.1 Literature Review

This section provides an overview of existing smart contract platforms such as Ethereum, Hyperledger Fabric, and EOS. Each platform's strengths and limitations are examined, highlighting areas where AI techniques can improve functionality and performance. This shows the various AI techniques that can be used to automate smart contracts. It looks at how natural language processing (NLP) can be used to interpret and generate contracts, machine learning algorithms for contract optimization and adaptation, and reinforcement learning for contract negotiation and enforcement.

Smart contracts are self-executing agreements based on blockchain technology principles. They are intended to enforce predefined rules and conditions automatically, without the use of intermediaries. Transparency, immutability, and lower transaction costs are just a few of the benefits of smart contracts. Traditional smart contracts, on the other hand, face challenges due to their limited expressiveness, lack of privacy, and reliance on manual coding and verification. To address these issues, researchers and developers have investigated incorporating AI techniques into smart contracts. Smart contracts can become more dynamic, intelligent, and adaptable by leveraging AI. AI-powered smart contracts have the potential to improve contract logic expressiveness, enable natural language interpretation and generation, automate contract optimization, and ease contract negotiation and enforcement.

The limitations of existing smart contract platforms can be addressed by incorporating AI techniques. For example, by optimizing the execution of smart contracts with AI algorithms, Ethereum's gas fees and scalability issues can be mitigated. The privacy concerns raised by Hyperledger Fabric can be addressed by utilizing AI models for secure and private contract negotiation and enforcement.

There are some techniques which is used to generate the smart contract automatically, namely natural language processing (NLP), machine learning, and reinforcement learning. We can use all this technology for generating smart contracts using artificial intelligence.

Using NLP techniques to interpret and generate contracts in human-readable language to automate the contract generation process. Optimizing smart contract execution by using machine learning algorithms to learn from historical data, predict outcomes, and adapt contracts to changing conditions. Using RL techniques, smart contracts can make autonomous decisions and dynamically adjust terms and conditions.

Improving the efficiency and scalability of smart contracts through the use of AI algorithms to optimize contract execution while reducing gas fees and privacy concerns. By addressing these issues, the proposed AI-generated smart contract system seeks to revolutionize contract management, improve efficiency, accuracy, and adaptability, and open new avenues for intelligent, autonomous, and secure contract execution.

Furthermore, the review of the literature focuses on real-world applications and case studies that demonstrate the potential impact of AI-generated smart contracts in industries such as finance, supply chain management, and legal services. It investigates these contracts' transformative power in terms of cost savings, operational efficiency, and transparency.

1.2 Role of AI in present scenario

AI has emerged as a game-changing technology with enormous impact and disruptive potential across multiple areas. One of the primary reasons for the importance of AI is its capacity to improve efficiency and production. AI systems can analyze huge volumes of data and make data-driven decisions at unprecedented rates by automating repetitive jobs and employing machine learning algorithms, resulting in streamlined processes and resource allocation. Furthermore, AI allows sophisticated data analysis and insights, allowing businesses to make educated decisions and uncover new trends and opportunities by extracting important information from vast databases. Furthermore, AI enhances consumer experiences through tailored interactions, recommendation systems, and sentiment analysis, resulting in increased happiness and loyalty. Some of the primary considerations that demonstrate the significance of AI in the present time: -

- Efficiency and Productivity - AI can automate routine tasks, allowing businesses to streamline operations and deploy resources more effectively. Thanks to machine learning and deep learning algorithms, AI systems can analyze massive amounts of data, spot patterns, and make decisions based on that data at previously unheard-of speeds.
- Improve Customer Experience - AI enables personalized interactions and experiences. Chatbots and virtual assistants use AI to provide real-time support, whereas recommendation systems use AI to deliver personalized product recommendations. AI-powered sentiment analysis enables businesses to better understand customer sentiment and satisfaction.

- Scientific and technological advancements - AI accelerates scientific discoveries by allowing researchers to handle and analyze large amounts of data, simulate complex situations, and make advances in a variety of fields such as genomics, medicine development, and climate modelling.
- Modernization of Autonomous Systems - AI enables robots, drones, and self-driving vehicles. Artificial intelligence (AI) algorithms process real-time data to make decisions, navigate environments, and complete difficult tasks accurately. These advancements have the potential to transform industries, increase efficiency, and improve transportation safety.
- Cybersecurity - AI provides businesses with powerful tools for detecting and mitigating cyber threats. Machine learning algorithms examine network activity, detect anomalies, and proactively detect security flaws.

It is crucial to design and put into practice responsible AI practices that ensure ethical concerns, fairness, and transparency as AI continues to advance, maximizing its potential for societal good.

1.3 Problem Statement

Today the world is mostly into artificial intelligence (AI) from the basic calculations to the complex machine design or using algorithm into the complex projects. AI plays an important role in developing all of this. Our main to choose the smart contract as our final year project in real time there are many websites which will help you to run the smart contract code, but every website will give you the code. Our project will provide you with the code and you can also run that code onto our website only.

The primary challenge is the manual coding and verification of smart contracts, which frequently necessitates technical expertise and can be error prone. Existing smart contract platforms also have scalability issues, privacy concerns, and the inability to handle complex decision-making scenarios.

To overcome these limitations and improve the capabilities of smart contracts, researchers must investigate the incorporation of artificial intelligence (AI) techniques. Smart contracts can become more dynamic, intelligent, and

adaptable by leveraging AI. Smart contracts powered by AI can help with natural language interpretation and generation, contract optimization, negotiation, and enforcement.

The problem statement is to create an AI-generated smart contract system that overcomes traditional smart contract limitations and provides a more efficient and flexible approach to contract execution and enforcement. To improve the expressiveness, adaptability, and intelligence of smart contracts, the system should use AI techniques such as natural language processing (NLP), machine learning (ML), and reinforcement learning (RL).

The project focuses on incorporating artificial intelligence (AI) techniques such as natural language processing and machine learning into the development of smart contracts. The implementation will include the design and construction of a prototype AI-generated smart contract system. The evaluation will compare the system's performance and benefits to traditional smart contracts.

Smart contracts are agreements that automatically fulfill their obligations because they are encoded in code. Due to their capacity for automation, transparency, and the removal of middlemen, they have gained substantial increase in popularity. However, the existing method of manually drafting smart contracts is laborious, prone to mistakes, and requires a high level of programming and contract law expertise.

In the present scenario the main problem with the smart contracts is lack of security, in terms of providing the output or generating it for the client, but AI generated smart contracts can be used to overcome this problem, in our project we have had use the blockchain technology which is known for its security mechanism.

Before explaining the project, some important terminology like blockchain, Artificial Intelligence are used in the project. We want to explain them before starting to explain our project. Blockchain is a ledger which is used to storage the data in the form of blocks which contents different types of data including image, text and many others. We have used blockchain for its security features, like transparency and no need of trusted third-party, this makes it less frauded, because buyers can directly connect to the sellers.

In this section of the report, we have defined what is our project, what kind of technology we have used, and why it is necessary. This section shows the importance of AI in the present time and how it will change our world, AI plays an important role in every field including medical, mechanical, software and many more. We have also defined the important terminology which is used during the project.

Next section consists of the analysis, requirements, methods, Results, verification and finally the conclusion and the future scope of the project.

Chapter 2

2. Analysis

Contracts are traditionally viewed in the world of legal agreements as written instruments that embody the mutually agreed upon terms and conditions between two parties, typically in the presence of third parties with specialized legal experience. Traditional contractual agreements, on the other hand, have inherent difficulties. For starters, the physical storage and administration of large contract files is a substantial logistical strain. Furthermore, the engagement of third parties raises worries about potential biases that could jeopardize the fairness of the contracts.

To overcome this situation smart contracts come into the market, these contracts are generally generated by machine-like computers, the term smart contracts first proposed by Nick Szabo in the year 1994. Smart contracts consist of all the legal terms and conditions embedded into the computer software and the hardware. Although Szabo conceptualized many parts of the smart contract in and associated studies, until the advent of blockchain in the past decade, the smart contract was only a theoretical artefact.

Smart contracts on the blockchain were sets of contracts that were installed on blockchain platforms and written in code scripts. These contracts were automatically carried out upon the arrival of required data. Imperative and declarative languages are two categories of high-level programming languages that can be used to create blockchain smart contracts. In existing blockchain smart contract systems, imperative languages were more prevalent. They were not simple for humans to read and understand, even though they could accomplish very sophisticated tasks. Declarative languages, on the other hand, are more naturally intuitive for human understanding because they are like rules and logic. Its integration into the current blockchain ecosystem had been supported by numerous research.

The successful development and implementation of the AI-generated smart contract system will help to advance the field of smart contracts and lay the groundwork for future research and development.

The first stage of a smart contract's life cycle, which includes generation, deployment, and execution, is known as smart contract generation. There are four different generation methodologies that have been noted in the literature: arbitrary smart contract limitations, formally specified smart contract constraints, business process-based smart contracts, and legal agreements as smart contracts. Their distinctions are primarily found in the smart contract representations they use and the formalistic degrees.

2.1 Requirements

As our project is software based then we only need to have the software as requirements. Main Requirements are-: VS code (for writing the code), Node modules (used in the code), mongodb (for the data storage), HTML and CSS (for the login page), typescript (part of javascript used to write the code). We will explain all this below by their working in our project.

VS code: - It is a very popular source-code editor which was develop by Microsoft. This is widely use because of it flexible environment, we can use it in windows, macos and Linux. It has an electron framework with many features like debugging, intelligent code completion, code refactoring and many more.

Typescript: - It is similar to javascript but not completely. Typescript is a strongly build programming language which is based on javascript. It provides better tooling than javascript and we can convert the code into javascript for running into different platform using Node.js. This understands the code of javascript and gives you great tooling experience without writing the extra additional code.

Node modules: - We have to use different modules for the better coding experience and in the form of result generation. Some of the modules are – eslint, chatscope, mongodb connections, API etc...

API: - Using a set of definitions and protocols, apis are techniques that allow two software components to communicate with one another. It stands for application programming interface. In the context of apis, the term Application refers to any software that performs a specified purpose. Interface can be thought of as a service contract between two programs. This contract details how the two will communicate with one another via requests and responses.

Their API definition outlines how developers should organize those calls and responses.

Mongodb: - mongodb is a cross-platform document-oriented database program that is open source. Mongodb was created by mongodb Inc. And is licensed under the Server-Side Public License (SSPL), which several distributions consider to be non-free. Mongodb is a MACH Alliance member.

Next js: - Next.js is a React library that adds functionality such as server-side rendering and the generation of static webpages. React is a javascript package that is commonly used to create web applications that are rendered in the client's browser using javascript.

Javascript: - javascript is the most widely used programming language on the planet. Javascript is the Web programming language. Javascript is simple to grasp.

Chapter 3

3. Implementation

3.1 Methodology

The design of the AI-generated smart contract system is the first step in the methodology. This includes defining the architecture and components of the system, such as the contract interpreter, AI algorithms, and decision-making module. To enable automated contract generation, adaptation, and enforcement, the design should ensure seamless integration and interaction among these components.

For creating this project, we have used an algorithm called generative AI (same algorithm which is used for chatgpt). Generative AI is an algorithm which is used to create a human-like response from the computer based on the data provided. For example, our project is mainly focused on smart contracts, so we data containing all the legal terms of a contract, and the pre-processed data according to the requirements of the contract. This algorithm could create or generate the answer based on the previously loaded data and this also gets the feedback from the user and learn from it.

Thoroughly describe the smart contract's unique criteria and constraints. Understanding the scope, parties involved, contract terms, obligations, and any set norms or conditions are all part of this.

Collect important data that will be used to inform the smart contract's content and settings. Historical contract data, legal papers, industry laws, and other relevant information may be included. Ascertain that the data is complete and indicative of the contract domain.

To verify the quality and usefulness of the collected data for analysis, preprocess it. This could include duties like cleaning the data, dealing with missing information, standardizing formats, and putting it into a format suited for AI algorithms.

Create artificial intelligence algorithms in our case that is generative AI capable of generating smart contract content. Natural language processing (NLP) techniques may be used to interpret legal text, machine learning algorithms may

be used to find patterns and produce clauses, and rule-based systems may be used to include predefined rules and conditions.

The integration of the trained AI model into a smart contract platform is the focus of the implementation stage. Data integration, smart contract template generation, and deployment considerations are all covered. In addition, the section emphasizes the significance of scalability, interoperability, and consensus mechanisms in the implementation process.

During the performance evaluation stage, the AI-generated smart contract system is tested using real-world contract scenarios. It explains how to choose evaluation metrics, performance benchmarks, and gather user feedback. The section emphasizes performance evaluation's iterative nature and its role in system optimization and refinement.

Preprocessed data can be used to train AI algorithms to learn patterns, correlations, and legal language structures. This entails entering the data into the AI model and optimizing its parameters so that accurate and meaningful contract clauses are generated. For this we have used the API data is already present there and do the further calculations or generation.

Following the training of the AI models, the AI-generated smart contract system is implemented. This includes creating the necessary software infrastructure, integrating the trained models into the system, and ensuring that different modules communicate effectively. The system should be implemented using best software engineering practices to ensure scalability, reliability, and security.

The system is optimized and refined based on the results of the performance evaluation. This could entail fine-tuning the AI models, increasing system efficiency, and addressing any identified limitations or challenges. Iterative refinement ensures the effectiveness of the system and aligns it with the desired goals of intelligent, autonomous, and efficient contract execution.

Once optimized, the system is ready for deployment and testing in a real-world or controlled environment. This entails installing the system on a relevant blockchain platform or infrastructure and thoroughly testing it to ensure its

functionality, security, and compatibility with existing smart contract ecosystems. The testing phase aids in the identification of any potential issues or bugs that must be addressed before the system is released for production use.

Various metrics and evaluation techniques are used to assess the performance of the AI-generated smart contract system. This section goes over performance evaluation criteria like contract accuracy, execution time, and adaptability to changing conditions. The evaluation results are presented, emphasizing the system's benefits and areas for improvement. The above methodology outlines a structured approach for developing the AI-generated smart contract system. Following this methodology ensures that the system is effectively designed, implemented, and evaluated, ensuring the successful integration of AI techniques into smart contract automation.

Use the taught AI model to generate the smart contract's content independently. Based on the learning patterns and contract needs, this includes drafting clauses, identifying variables, and constructing conditions. For the storage of the data which is given by the user during the login process, is stored in mongo DB.

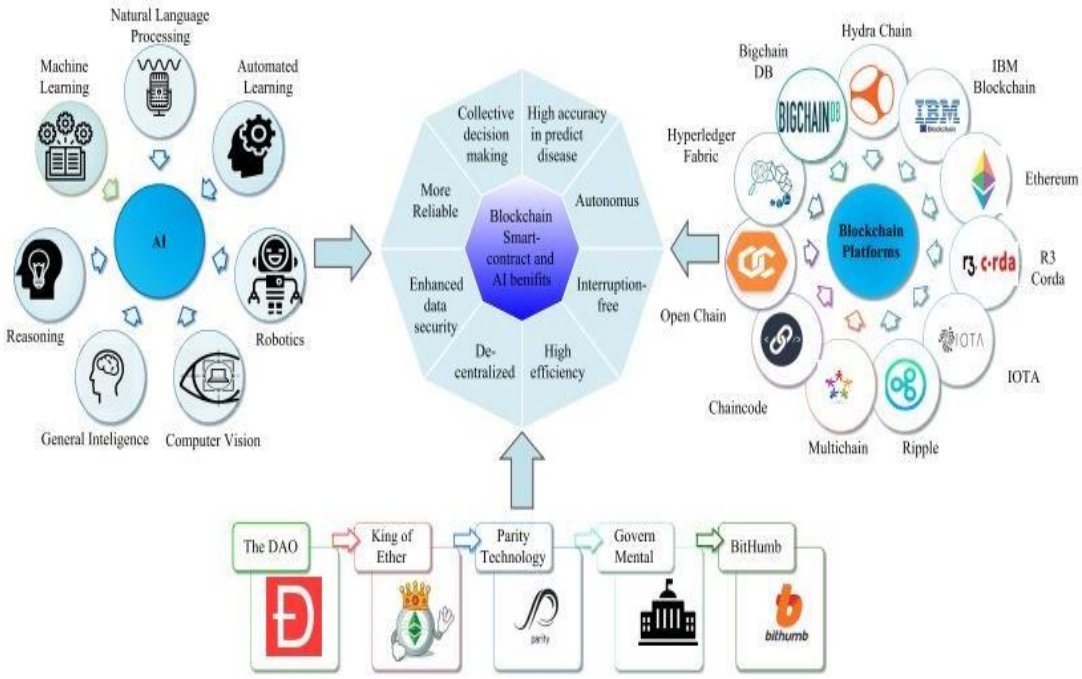


Figure 3.1

Smart contracts enabled projects.

The above image shows what kind of applications the smart contract has, and technologies use to do that. This image shows the technology which is being used for the development of smart contracts on the left side and the right-side platform are there where we can run our code there and we can also store our data in the form of blocks in blockchain which is highly secure. In the bottom it shows some projects which is develop by using this platform and the technology.

3.2 Code of the project

CHAT.TSX FILE

```tsx
import { ChatContainer, MainContainer, Message, MessageInput, MessageList, TypingIndicator } from "@chatscope
import styles from '@/styles/Chat.module.css';
import { useEffect, useRef, useState } from "react";
import { ChatCompletionRequestMessageRoleEnum, Configuration, OpenAIApi } from "openai";

type Message = {
    content: string | null;
    sentTime: number;
    sender: string;
    direction: 'incoming' | 'outgoing';
}

const CHAT_USER = "Ujjwal";
const DEFAULT_BEHAVIOR = "General Conversation";
const CONFIGURATION = new Configuration({

})
const OPENAI_CLIENT = new OpenAIApi(CONFIGURATION);

export default function Chat() {
    const messageInput = useRef<HTMLDivElement>(null);
    const [messages, setMessages] = useState<Message[]>([]);
    const [behaviorInput, setBehaviorInput] = useState(DEFAULT_BEHAVIOR);
    const [behavior, setBehavior] = useState(DEFAULT_BEHAVIOR);

    const [waitingForResponse, setWaitingForResponse] = useState(false);
    const [rateLimited, setRateLimited] = useState(false);

    useEffect(() => {
        if (!waitingForResponse) {
```

```
            messageInput.current?.focus();
        }
    }, [waitingForResponse]);

    const sendMessage = async (innerHtml: string, textContent: string, innerText: string, nodes: NodeList)
        const newMessageList = [...messages];
        const newMessage: Message = {
            content: textContent,
            sentTime: Math.floor(Date.now() / 1000),
            sender: 'You',
            direction: 'outgoing',
        }
        newMessageList.push(newMessage);
        setMessages([...newMessageList]);

        setWaitingForResponse(true);
        const response = await getResponse(newMessageList);

        const newMessageResponse: Message = {
            content: response?.content,
            sentTime: Math.floor(Date.now() / 1000),
            sender: CHAT_USER,
            direction: 'incoming',
        }

        newMessageList.push(newMessageResponse);
        setMessages([...newMessageList]);
        setWaitingForResponse(false);
    }
```

```
const getResponse = async (newMessageList: Message[]) => {
    const systemMessage = {
        role: ChatCompletionRequestMessageRoleEnum.System,
        content: behavior,
    }

    const input = newMessageList.map((message) => {
        return {
            role: message.sender === CHAT_USER ? ChatCompletionRequestMessageRoleEnum.Assistant : ChatCo
            content: message.content,
        }
    });

    const response = await OPENAI_CLIENT.createChatCompletion({
        model: 'gpt-3.5-turbo',
        messages: [systemMessage, ...input],
    });
    console.log(response);

    return {
        content: response.data.choices[0].message?.content,
    }

    await new Promise(f => setTimeout(f, 1000));
    return {
        content: `${Math.random()}`,
    }
}
```

```
const updateBehavior = () => {
    const finalBehavior = behaviorInput.trim().length ? behaviorInput.trim() : DEAFULT_BEHAVIOR;
    setBehavior(finalBehavior);
}

return (
    <div className={styles.container}>
        <div className={styles.inputContainer} >
            <input className={styles.input} value={behaviorInput} onChange={e => setBehaviorInput(e.targe
            <button className={styles.submit} onClick={updateBehavior}>Update Behavior</button>
        </div>
        <div className={styles.chatWrapper}>
            <div className={styles.chatContainer}>
                <MainContainer>
                    <ChatContainer>
                        <MessageList classNames={styles.chatMessageList}
                            typingIndicator={waitingForResponse && <TypingIndicator content="Ujjwal is th
                            {
                                messages.map((message) => {
                                    return (
                                        // eslint-disable-next-line react/jsx-key
                                        <Message
                                            model={{
                                                message: message.content,
                                                sentTime: `${message.sentTime}`,
                                                sender: message.sender,
                                                direction: message.direction,
                                                position: 'normal'
                                                model={{
                                                    message: message.content,
                                                    sentTime: `${message.sentTime}`,
                                                    sender: message.sender,
                                                    direction: message.direction,
                                                    position: 'normal',
                                                    type: 'text',
                                                }}
                                            />
                                        )
                                    })
                                }
                            </MessageList>
                            <MessageInput placeholder="Type message here"
                                style={{ background: '#432A74' }}
                                onSend={sendMessage}
                                autoFocus={true}
                                attachButton={false}
                                disabled={waitingForResponse}
                                ref={messageInput}
                            />
                        </ChatContainer>
                    </MainContainer>
                </div>
            </div>
        </div>
    )
}
```

INDEX.TSX

```tsx
import Head from 'next/head'
import Image from 'next/image'
import { Inter } from 'next/font/google'
import styles from '@/styles/Home.module.css'

import Router from 'next/router'

const inter = Inter({ subsets: ['latin'] })

export default function Home() {
  return (

    <>
      <Head>
        <title>Create Next App</title>
        <meta name="description" content="Generated by create next app" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <main className={styles.main}>

      </main>
    </>
  )
}
```

LOGIN_PAGE.TSX

```tsx
import { useState } from 'react';
import { loginUser } from './api/api';
import Router from 'next/router'

export default function LoginPage() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');



  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    try {
      const response = await loginUser(email, password);
      if(response.message === "Login successful!"){
        Router.push('/chat')
      }else{
        alert("PassWord incorrect")
      }
      console.log(response); // Handle successful login
    } catch (error) {
      setError(error.response.data.message); // Handle login error
    }
  };
```

```jsx
return (
  <div>
    <h1>Login</h1>
    {error && <p>{error}</p>}
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        placeholder="Email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
      />
      <br />
      <input
        type="password"
        placeholder="Password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
      />
      <br />
      <button type="submit">Login</button>
    </form>
  </div>
);
```

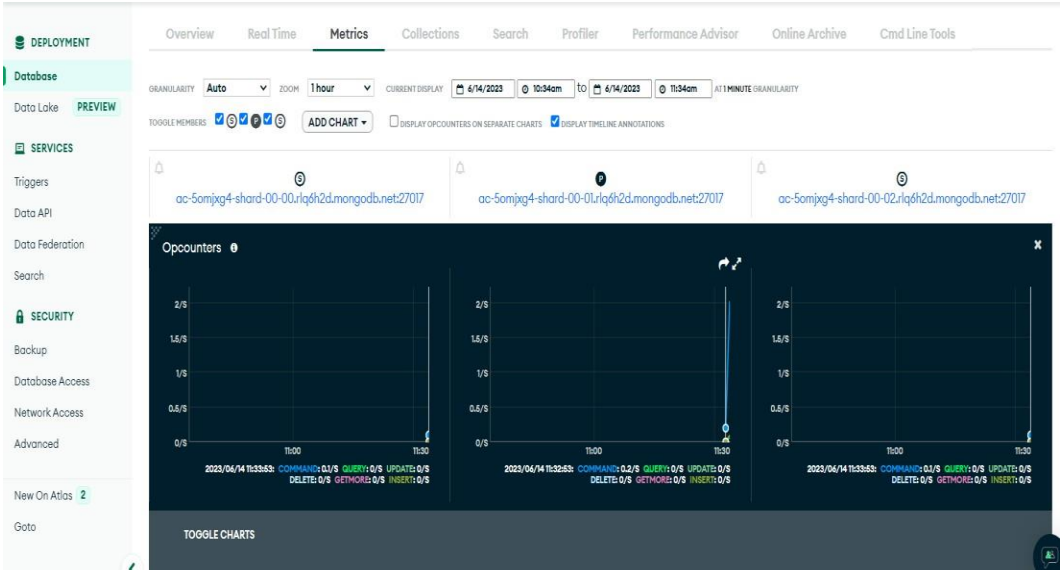
Chapter 4

4. Result

4.1 Images of the result

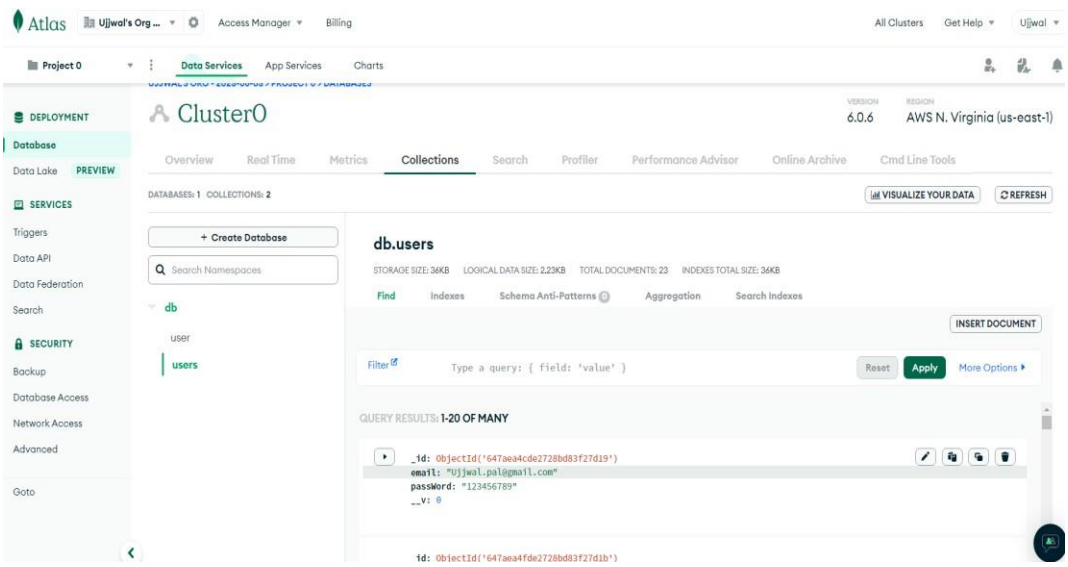
Figure 4.1
Project main screen

The above image is the front or main page of our project. This is the main page of our project here you can show that there is a screen where you can type your questions and after sending the question our system will run or process that question in their system and from the trained data which is already feed it gives the output in the form of code, you can copy that code and run that code on any platform which supports the generation of the smart contract from the code, and you will get your contract from it. This will save the time of both the parties for the contract and running behind the law person, the traditional method is time consuming.



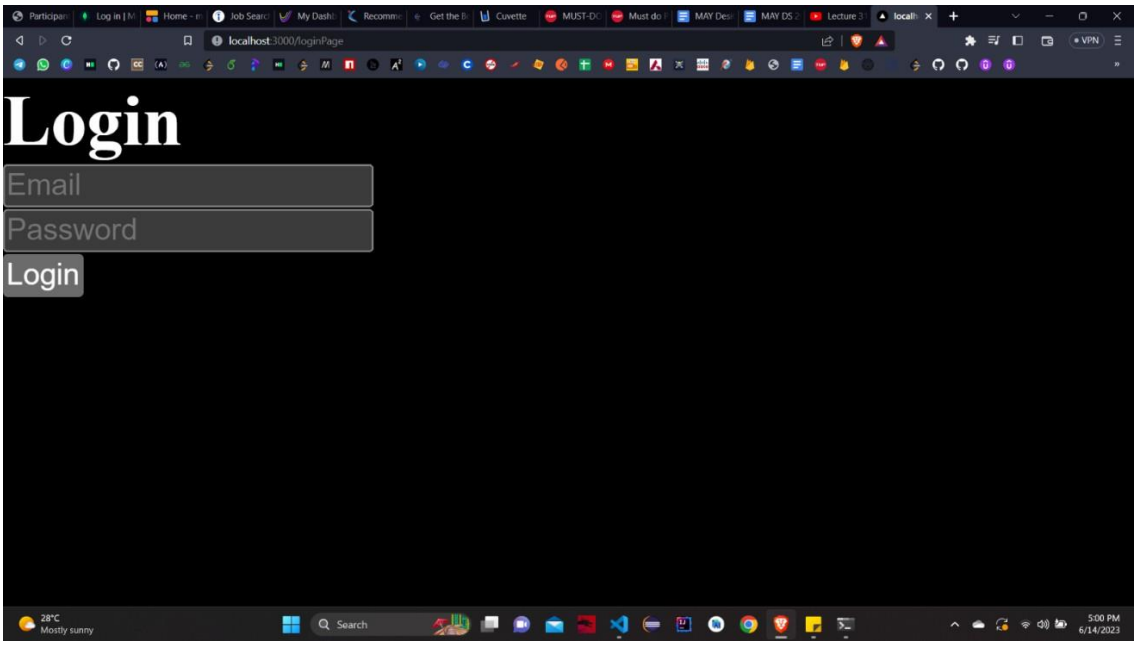
Graph of the data store

Figure 4.2

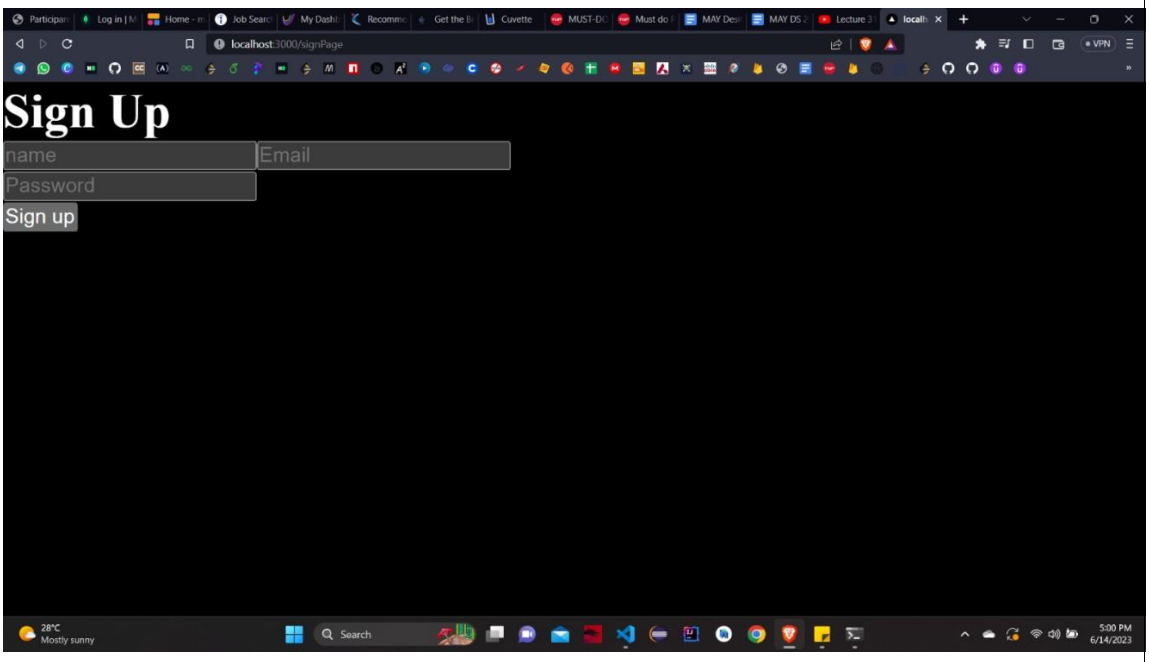


Data store in database

The above two images show the data storage in mongodb, data like username, password, and email address.



Login page
Figure 4.4



Sign-up page Figure 4.5

The above two images show the login page and the sign up page.

4.2 Conclusion from the result

The goal of generating a smart contract with AI is to automate and streamline the process, saving time and effort. When using an AI-generated smart contract, the result is typically a draught or template of the contract text generated based on the information provided. The generated contract may contain clauses, conditions, and provisions related to the contract's subject matter. It can include sales agreements, employment contracts, intellectual property licenses, and other things. This is the basic result which we usually get from the smart Contract.

In our case we are generating smart contracts for the different types of business like for e-commerce and more. The screenshots which we have uploaded show where the credentials are stored and how the data looks like. The main screen of our project looks very similar to ChatGPT but very different in the form of working.

If you give any condition to generate the smart contract our system will generate that, and you can directly use that generated for your contracts and you can that on any available smart contract running platforms.

Finally, by combining AI and blockchain technologies, AI-generated smart contracts have the potential to revolutionize contract management. They provide advantages such as precision, efficiency, and transparency. However, legal frameworks, data privacy, and system scalability issues must be addressed. Organizations can unlock the potential of AI-generated smart contracts and transform the way contracts are created, adapted, and enforced by following a systematic methodology.

4.3 Example After Deployment

The following Smart Contract is described: -

STEP 1: Create the contract file.

Make a new Remix Ecom.sol file and a new contract shopping.

STEP 2: Register as a Seller and pay a fee of 1000 Wei to the owner.

In the function Object () { [native code] }, make a Public address variable owner and a deployer address owner. Address mappings pointing to our struct seller should be used. The seller struct contains all of the necessary information about the seller.

Include a method seller. Sign up to see if the seller is already registered and if the message value, along with the function, is equal to 1000 Wei.

STEP 3: Create a product list with all of the necessary information.

Add a struct product with all necessary variables and create a string mapping to the struct so that you can update or buy Product with productid String.

Add a function to determine whether the seller paid the bank guarantee and whether a product with the same productid is already active.

STEP 4: Monitor Orders Placed by Buyers

Add a struct orders Placed with the necessary Tracking Detail variables, as well as a mapped struct array with the seller's address.

When the buyer placed an order, the order details were pushed to the seller's Orders.

Orders placed by buyers with purchaseid can be tracked.

STEP 5: Ship Products and Update Shipment Information

Create a Struct seller Shipment with all the variables needed for tracking and updating details, and map this structure to an address and nested unit. With each unique purchase, each seller can update shipment details. Id Shipment details can be updated using the function update Shipments with purchaseid.

STEP 6: Cancelled Orders Are Refunded

Create a refund function to determine whether a product with a specific purchaseid is active or not, whether the buyer cancelled the order, and whether the seller released an amount equal to the product price.

STEP 7: Establish a Buyer's Account

Create a Struct user with the necessary user details and map it to an address.

Add a function create Account to the users mapping and push function arguments (user details).

STEP 8: Ordering and Tracking Orders

STEP 9: Order Cancellation

Add a function to cancel an order that takes a purchaseid as an argument and determines whether or not the product with that purchaseid is ordered by the current caller and whether or not the product with that purchaseid is active.

E-commerce Smart Contract for test

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;


contract shopping {

    address payable public owner;

    constructor() {
        owner = payable(msg.sender);
    }
    uint id;
    uint purchaseId;
    struct seller {
        string name;
        address addre;
        uint bankGuaraantee;
        bool bgPaid;
    }
    struct product {
        string productId;
        string productName;
        string Category;
        uint price;
        string description;
        address payable seller;
        bool isActive;

    }
    struct ordersPlaced {
```

```
    string productId;
    uint purchaseId;
    address orderedBy;
}
struct sellerShipment {
    string productId;
    uint purchaseId;
    string shipmentStatus;
    string deliveryAddress;
    address payable orderedBy;
    bool isActive;
    bool isCanceled;
}
struct user {
    string name;
    string email;
    string deliveryAddress;
    bool isCreated;
}
struct orders {
    string productId;
    string orderStatus;
    uint purchaseId;
    string shipmentStatus;
}
mapping(address => seller) public sellers;
mapping(string => product) products;
product[] public allProducts;
mapping(address => ordersPlaced[]) sellerOrders;
mapping(address => mapping(uint => sellerShipment)) sellerShipments;
```

```
mapping(address => user) users;
mapping(address => orders[]) userOrders;

event OrderPlaced(string _status, address indexed _buyer, string _productId);

function sellerSignUp(string memory _name) public payable {
    require(!sellers[msg.sender].bgPaid, "You are Already Registered");
    require(msg.value == 1000 wei, "Bank Guarantee of 1000 wei Required");
    owner.transfer(msg.value);
    sellers[msg.sender].name = _name;
    sellers[msg.sender].addre = msg.sender;
    sellers[msg.sender].bankGuaraantee = msg.value;
    sellers[msg.sender].bgPaid = true;
}

function createAccount(string memory _name, string memory _email, string memory _deliveryAddress) public {

    users[msg.sender].name = _name;
    users[msg.sender].email = _email;
    users[msg.sender].deliveryAddress = _deliveryAddress;
    users[msg.sender].isCreated = true;
}

function buyProduct(string memory _productId) public payable {


    require(msg.value == products[_productId].price, "Value Must be Equal to Price of Product");
    require(users[msg.sender].isCreated, "You Must Be Registered to Buy");

    products[_productId].seller.transfer(msg.value);
```

```
    purchaseId = id++;
    orders memory order = orders(_productId, "Order Placed With Seller", purchaseId, sellerShipments[produ
    userOrders[msg.sender].push(order);
    ordersPlaced memory ord = ordersPlaced(_productId, purchaseId, msg.sender);
    sellerOrders[products[_productId].seller].push(ord);

    sellerShipments[products[_productId].seller][purchaseId].productId = _productId;
    sellerShipments[products[_productId].seller][purchaseId].orderedBy = payable(msg.sender);
    sellerShipments[products[_productId].seller][purchaseId].purchaseId = purchaseId;
    sellerShipments[products[_productId].seller][purchaseId].deliveryAddress = users[msg.sender].deliveryA
    sellerShipments[products[_productId].seller][purchaseId].isActive = true;

    emit OrderPlaced("Order Placed With Seller", msg.sender, _productId);
}

function addProduct(string memory _productId, string memory _productName, string memory _category, uint _p
    require(sellers[msg.sender].bgPaid, "You are not Registered as Seller");
    require(!products[_productId].isActive, "Product With this Id is already Active. Use other UniqueId");


    product memory newProduct = product(_productId, _productName, _category, _price, _description, payable
    products[_productId].productId = _productId;
    products[_productId].productName = _productName;
    products[_productId].Category = _category;
    products[_productId].description = _description;
    products[_productId].price = _price;
    products[_productId].seller = payable(msg.sender);
    products[_productId].isActive = true;
```

```
function myOrders(uint _index) public view returns(string memory, string memory, uint, string memory) {
    return (userOrders[msg.sender][_index].productId, userOrders[msg.sender][_index].orderStatus, userOrders
}

function getOrdersPlaced(uint _index) public view returns(string memory, uint, address, string memory) {
    return (sellerOrders[msg.sender][_index].productId, sellerOrders[msg.sender][_index].purchaseId, seller
}

function getShipmentDetails(uint _purchaseId) public view returns(string memory, string memory, address, str

    return (sellerShipments[msg.sender][_purchaseId].productId, sellerShipments[msg.sender][_purchaseId].shi
}
```

Test file for above Smart Contract

```
const { expect } = require("chai");
const { ethers } = require("hardhat");

describe("shopping contract", function () {
  let owner;
  let seller;
  let buyer;
  let shopping;

  beforeEach(async function () {
    [owner, seller, buyer] = await ethers.getSigners();

    const Shopping = await ethers.getContractFactory("shopping");
    shopping = await Shopping.connect(owner).deploy();
    await shopping.deployed();
  });

  it("should add a product", async function () {
    await shopping.connect(seller).sellerSignUp("Seller 1", { value: 1000 });
    await shopping.connect(buyer).createAccount("Buyer 1", "buyer1@example.com", "123 Main St");

    const productId = "product1";
    const productName = "Product 1";
    const category = "Category 1";
    const price = 100;
    const description = "This is product 1";

    await shopping.connect(seller).addProduct(productId, productName, category, price, description);

    const product = await shopping.allProducts(0);
```

```
  const product = await shopping.allProducts(0);

  expect(product.productId).to.equal(productId);
  expect(product.productName).to.equal(productName);
  expect(product.Category).to.equal(category);
  expect(product.description).to.equal(description);
  expect(product.seller).to.equal(seller.address);

});

it("should buy a product and create an order", async function () {
  await shopping.connect(seller).sellerSignUp("Seller 1", { value: 1000 });
  await shopping.connect(buyer).createAccount("Buyer 1", "buyer1@example.com", "123 Main St");

  const productId = "product1";
  const productName = "Product 1";
  const category = "Category 1";
  const price = 100;
  const description = "This is product 1";

  await shopping.connect(seller).addProduct(productId, productName, category, price, description);


  await shopping.connect(buyer).buyProduct(productId, { value: price });

  await expect(shopping.connect(buyer).buyProduct("product1", { value: 100 }))
  .to.emit(shopping, "OrderPlaced");
});
```

```
  it("should allow a buyer to cancel an order", async function () {
    await shopping.connect(seller).sellerSignUp("My Store", { value: 1000 });
    await shopping.connect(seller).addProduct("product1", "Product 1", "Category A", 100, "Description");
  });
});
```

Result of Test File After Running

```
PS D:\SMart contract\Tutorials> cd EccomerceSmartContract
PS D:\SMart contract\Tutorials\EccomerceSmartContract> cd test
PS D:\SMart contract\Tutorials\EccomerceSmartContract\test> npx hardhat test
WARNING: You are using a version of Node.js that is not supported, and it may work incorrectly, or not work at all. See https://hard
hat.org/nodejs-versions



  shopping contract
    ✓ should add a product (119ms)
    ✓ should buy a product and create an order (128ms)
    ✓ should allow a buyer to cancel an order (39ms)


  3 passing (2s)

PS D:\SMart contract\Tutorials\EccomerceSmartContract\test>
```

Chapter 5

5. Conclusion and Future Scope

5.1 Future scope

In recent years, the integration of artificial intelligence (AI) and blockchain technology has paved the way for novel applications, with smart contracts being one prominent example. Smart contracts are self-executing agreements that automatically enforce the terms and conditions that they contain. While traditional smart contracts have already demonstrated their ability to streamline business processes, the introduction of AI-generated smart contracts holds enormous promise for the future. This paper investigates the potential benefits, challenges, and implications of AI-generated smart contracts across various industries.

1. Efficiency and Accuracy

- Enhanced automation and speed in contract execution

- Reduction of human errors and biases

2. Intelligent Contract Creation

- Natural language processing and machine learning for contract drafting

- Tailoring contracts based on contextual data and variables

3. Risk Mitigation and Compliance

- Automated auditing and monitoring of contract terms and conditions

- Real-time risk assessment and mitigation strategies

Industry Applications

1. Supply Chain Management

- AI-generated smart contracts for tracking and verifying shipments

- Efficient management of inventory, payments, and logistics

2. Financial Services

- Smart contracts for automated loan agreements, insurance claims, and payments

- Fraud detection and prevention through AI-powered analytics

3. Real Estate

- Streamlining property transactions, rental agreements, and escrow processes

- Verification of property titles and ownership records using AI algorithms

4. Healthcare

- Smart contracts for patient consent, medical record management, and billing

- Secure sharing of health data while maintaining privacy and confidentiality

5. Intellectual Property

- AI-generated smart contracts for copyright, licensing, and royalty agreements

- Intellectual property protection through immutable blockchain records

Challenges and Considerations

1. Legal and Regulatory Frameworks

- Addressing legal complexities surrounding AI-generated smart contracts

- Ensuring compliance with existing contractual laws and regulations

2. Security and Privacy

- Vulnerabilities and risks associated with AI algorithms and data handling

- Safeguarding sensitive information and preventing unauthorized access

3. Ethical Implications

- Transparency and accountability in AI-generated contract decision-making

- Mitigating biases and ensuring fairness in contract negotiation and enforcement

Future Trends and Opportunities

1. Integration with the Internet of Things (iot)

- AI-generated smart contracts enabling autonomous interactions between devices

- Smart homes, smart cities, and autonomous vehicles leveraging AI and blockchain

2. Interoperability and Standardization

- Establishing common protocols and frameworks for AI-generated smart contracts

- Seamless integration and compatibility across different blockchain platforms

3. Decentralized Autonomous Organizations (daos)

- AI-generated smart contracts facilitating the creation and operation of daos

- Democratized decision-making and governance structures

Smart contracts have the potential to revolutionize various industries by improving efficiency, accuracy, and automation. Benefits of AI-generated smart contracts include improved contract authoring, risk mitigation, and compliance. These benefits can be leveraged in various areas such as supply chain management, financial services, real estate, healthcare, and intellectual property.

Supply Chain Management streamlines processes by automating shipment tracking and verification with AI-generated smart contracts. It also enables efficient management of inventory, payments and logistics, reducing delays and optimizing resource allocation. AI-powered smart contracts bring accuracy and transparency to supply chain operations by eliminating manual errors and bias.

In the financial services space, AI-generated smart contracts will automate loan agreements, insurance claims and payments, reducing paperwork and processing time. AI algorithms can also be used for fraud detection and prevention, enabling real-time analysis of trading patterns and anomalies. This makes financial transactions safer and more reliable, benefiting both customers

and financial institutions.

The real estate industry can use AI-generated smart contracts to simplify the process of real estate transactions, leasing, and escrow. These agreements allow verification of title and ownership records, reducing the need for intermediaries and minimizing the risk of fraud. AI-powered smart contracts can save time and money for everyone involved by automating compliance checks and streamlining contract negotiations.

In healthcare, AI-generated smart contracts can play an important role in patient consent, medical record management, and billing processes. By automating these tasks, providers can increase efficiency, reduce administrative burden, and increase security of patient data. Additionally, AI-powered analytics can help identify patterns and insights from vast amounts of health data to deliver more personalized and effective treatments.

AI-generated smart contracts can enhance intellectual property protection. These agreements facilitate copyrights, licenses, and license agreements, ensure that creators are compensated, and their work is protected. AI-powered smart contracts powered by blockchain technology provide an immutable record, making it easier to enforce intellectual property rights and prevent abuse.

Despite the bright future of AI-generated smart contracts, there are challenges and considerations that need to be addressed. Legal and regulatory frameworks must keep pace with technological evolution to ensure compliance with existing contractual laws and regulations. Additionally, there are security and privacy concerns around vulnerabilities in AI algorithms and the handling of sensitive information. Ethical implications such as bias in decision-making and fairness in negotiation and execution also need to be carefully considered.

Looking ahead, future trends and opportunities for AI-generated smart contracts include integration with the Internet of Things (iot). This integration enables autonomous interactions between devices, paving the way for smart homes, smart cities, and self-driving cars. Interoperability and standardization efforts are essential for seamless integration and compatibility between different blockchain platforms. In addition, AI-generated smart contracts facilitate the establishment and operation of decentralized autonomous organizations (daos), facilitating decentralized decision-making and governance structures.

In summary, the future scope of AI-generated smart contracts is large and promising. By combining the power of AI and blockchain technology, these agreements will enable efficiency, security, and automation across industries.

5.2 Limitations

Although smart contracts developed by AI have many benefits, it is vital to be aware of their drawbacks. The following are some significant drawbacks of AI-generated smart contracts:

- Legal and Regulatory Difficulties: The enforceability, liability, and compliance with current laws and regulations of AI-generated smart contracts provide legal and governmental difficulties. Concerns concerning responsibility and legal duties may arise due to the dynamic nature of AI algorithms and the automated decision-making process.
- Contracts can contain complex legal language and require careful interpretation. Artificial intelligence (AI) models may have difficulty comprehending and interpreting the nuanced and situation-specific elements of contractual terminology and clauses. It continues to be difficult to ensure accurate interpretation and prevent misunderstandings.
- AI-generated smart contracts depend on enormous volumes of data, including private and sensitive data. There are many obstacles to securing sensitive data and guaranteeing compliance with privacy laws. Important factors to consider include preserving data integrity, limiting unauthorized access, and attending to worries about data breaches.
- Ethical Points to Keep in Mind: When creating and analyzing contracts, AI algorithms may unintentionally reinforce any biases or unethical behavior that was present in the training data. Creating moral standards and ensuring fairness and openness are crucial.
- Scalability and interoperability of AI-generated smart contracts are crucial factors to take into account as the volume and complexity of contracts rises. A hurdle that needs to be solved is making sure the system can manage numerous contracts at once and interface with current contract management systems and blockchain networks without

any issues.

- The AI models utilized for contract generation and analysis have some restrictions. They could have trouble comprehending some kinds of contracts or intricate legal ideas. A current topic of research is enhancing the precision, adaptability, and robustness of AI models to manage a variety of contract circumstances.
- Even if AI-generated smart contracts are capable of automation, human engagement and trust are still essential. Human decision-making, negotiation, and contextual awareness are frequently needed in contracts. To ensure openness, accountability, and trust in the contract management process, it's critical to strike the correct balance between automation and human involvement.
- The broad use of AI-generated smart contracts can encounter opposition from established legal frameworks and conventional contract management procedures. Successful deployment depends on overcoming opposition, educating stakeholders, and building trust in the technology.
- Smart contracts developed by AI must be implemented, which necessitates a substantial investment in AI infrastructure, data collecting, training, and system integration. Companies should carefully assess the costs and hassles involved in implementing and maintaining such systems.

5.3 Conclusion

In conclusion, the emergence of AI-generated smart contracts represents a significant advancement in the field of blockchain technology. These contracts have the potential to revolutionize numerous industries by unlocking efficiency, security, and automation in the way contracts are created, executed, and enforced.

AI-generated smart contracts offer several key benefits. They enhance efficiency and accuracy by automating contract execution, reducing human errors, and eliminating manual processes. The intelligent contract creation capabilities of AI, such as natural language processing and machine learning,

enable the creation of tailored contracts based on contextual data and variables. This customization improves the accuracy and relevance of contracts to specific business needs.

Furthermore, AI-generated smart contracts mitigate risks and ensure compliance. They automate auditing and monitoring of contract terms and conditions, providing real-time risk assessment and mitigation strategies. This proactive approach minimizes the potential for contractual disputes and non-compliance, improving overall contract management.

The potential applications of AI-generated smart contracts span across various industries. Supply chain management can benefit from automated tracking and verification of shipments, optimizing inventory management and logistics. Financial services can leverage smart contracts for automated loan agreements, insurance claims, and fraud detection, streamlining processes and enhancing security. Real estate transactions can be simplified through automated property transactions, rental agreements, and verification of ownership records. Healthcare can benefit from automated patient consent, medical record management, and billing processes, improving efficiency and data security. Intellectual property rights can be protected through AI-generated smart contracts, ensuring fair compensation and safeguarding creative works.

The report delves into the topic of AI-generated smart contracts, beginning with an overview of the importance and potential benefits of incorporating AI into the contract management domain. The literature review examines existing research, studies, and advancements in the field of AI-generated smart contracts in depth. It discusses the evolution of smart contracts, the role of AI in contract automation, and the challenges and opportunities that this technology presents. However, the adoption of AI-generated smart contracts also presents challenges and considerations. Legal and regulatory frameworks need to adapt to address the complexities of these contracts and ensure compliance with existing laws. Security and privacy concerns must be addressed to safeguard sensitive information and prevent unauthorized access. Ethical implications, such as biases in decision-making and fairness in contract negotiation, need to be carefully managed to maintain trust and transparency.

Looking to the future, the integration of AI-generated smart contracts with emerging technologies offers exciting opportunities. The combination of AI-generated smart contracts with the Internet of Things (iot) can enable autonomous interactions between devices, leading to the development of smart homes, smart cities, and autonomous vehicles. Interoperability and standardization efforts are essential to ensure seamless integration across different blockchain platforms, fostering collaboration and compatibility. Decentralized Autonomous Organizations (daos) can be facilitated through AI-generated smart contracts, enabling decentralized decision-making and governance structures.

In conclusion, AI-generated smart contracts hold immense potential to transform business processes, improve efficiency, enhance security, and automate contract management across industries. While challenges and considerations exist, ongoing research and development, along with a proactive approach to addressing legal, security, and ethical concerns, will pave the way for a future where AI-generated smart contracts play a central role in shaping a more streamlined, transparent, and trustworthy business environment.

References:-

[1] "Huang, Q., Li, L., Tao, D., & Wu, X. (2019). Automated Construction of Blockchain-based Smart Contracts: A Preliminary Study. In 2019 IEEE International Conference on Smart Internet of Things (SmartIoT) (pp. 55-61). IEEE. doi: 10.1109/SmartIoT.2019.00021"

[2] "Murer, P., & Sorniotti, A. (2017). Blockchain and Smart Contract. In A. Misra, I. Woungang, & S. C. Misra (Eds.), Guide to Blockchain Technologies (pp. 189-215). Springer. doi: 10.1007/978-3-319-62273-4_9 Samaniego, M., Cruz, E., & Murillo, L. (2020). Smart Contracts for Blockchain Platforms: A Systematic Mapping Study"

[3] "Bhatia, V., & Gupta, V. (2020). Smart Contracts for Blockchain: An Empirical Analysis and a Proposed Decision-Making Model. Journal of Information Privacy and Security, 16(3), 182-197. doi: 10.1080/15536548.2020.1744592"

[4] "De Filippi, P., & Hassan, S. (2016). Blockchain Technology as a Regulatory Technology: From Code is Law to Law is Code. First Monday, 21(12). doi: 10.5210/fm.v21i12.7113"

[5] "Bhargavan, K.; Delignat-Lavaud, A.; Fournet, C.; Gollamudi, A.; Gonthier, G.; Kobeissi, N.; Kulatova, N.; Rastogi, A.; Sibut-Pinote, T.; Swamy, N.; et al. Formal verification of smart contracts. In Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, Vienna, Austria, 24 October 2016."

[6] "Wang, Y.; Kazama, J.; Tsuruoka, Y.; Chen, W.; Zhang, Y.; Torisawa, K. Improving Chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. Proceedings of 5th International Joint Conference on Natural Language Processing, Chiang Mai, Thailand, 8–13 November 2011; pp. 309–317**"**

[7] **“**Bartoletti, M.; Zunino, R. BitML: A Calculus for Bitcoin Smart Contracts. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018**”**

[8] **“**https://en.wikipedia.org/wiki/Visual_Studio_Code**”**

[9] **“**https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-generative-ai**”**

[10] **“**https://www.mongodb.com/what-is-mongodb **”**

[11] **“**https://www.mdpi.com/2076-3417/12/9/4773 **”**

Project Report

ORIGINALITY REPORT

12%	9%	2%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Alliance University Student Paper	5%
2	archive.org Internet Source	1%
3	Submitted to John Paul College Student Paper	1%
4	academic.oup.com Internet Source	<1%
5	ijarsct.co.in Internet Source	<1%
6	en.wikipedia.org Internet Source	<1%
7	www.pelican.ai Internet Source	<1%
8	ethesis.nitrkl.ac.in Internet Source	<1%
9	Submitted to Queen's University of Belfast Student Paper	<1%

10	www.theseus.fi Internet Source	<1%
11	essay.utwente.nl Internet Source	<1%
12	www.gabormelli.com Internet Source	<1%
13	Submitted to University of Sheffield Student Paper	<1%
14	online-cig.ase.ro Internet Source	<1%
15	www.nodexlgraphgallery.org Internet Source	<1%
16	www.interviewmocks.com Internet Source	<1%
17	Submitted to Southern New Hampshire University - Continuing Education Student Paper	<1%
18	Poshan Yu, Zixuan Zhao, Emanuela Hanes. "chapter 1 Integration of the Internet of Things and Blockchain to Promote Collaboration in Smart Cities", IGI Global, 2023 Publication	<1%
19	download.excelingtech.co.uk Internet Source	<1%
	www.propharmagroup.com	

20	Internet Source	<1%
21	123dok.org Internet Source	<1%
22	uu.diva-portal.org Internet Source	<1%
23	www.jot.fm Internet Source	<1%
24	arxiv1.library.cornell.edu Internet Source	<1%
25	dspace.upt.ro Internet Source	<1%
26	openpublichealthjournal.com Internet Source	<1%
27	www.coursehero.com Internet Source	<1%
28	www.ideals.illinois.edu Internet Source	<1%
29	www.modul.ac.at Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

PROFORMA FOR APPROVAL OF PROJECT REPORT

Name of the student : SHAILENDER MAURIYA (19030141CSE075)

Name of the Department : Computer Science and Engineering

Branch : Computer Science and Engineering

Year : 2019-2023

Date of Submission :

Batch Number :15

Items	Yes	No
Whether all pages are as per instruction?		
Whether the contents of report are in correct order as given in the manual?		
Whether the margin specification and spacing for the text correctly followed as per instruction in the manual?		
Whether typed in black?		
Whether numbers given for figures, tables & other pages serially and in correct position?		
Whether tables numbered properly with captions given as top of tables?		
Whether figures & diagram numbered properly with caption given below?		
Whether binding is, OK?		
Whether references are marked in text within square brackets?		
Whether CD containing executable s/w, required libraries & source code?		
Comment:		**Rating: /10**

Internal Guide **Head of Department**