Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

CSE4054

ARTIFICIAL INTELLIGENCE IN BLOCKCHAIN

FINAL REPORT – Project Report

DECENTRALIZED BLOCKCHAIN SUPERSET PORTAL

By,

Uppanapalli Lakshmi Sowjanya – 20BAI1289

Amaravadi Dheeraj – 20BAI1191

Mandla Sheshi Kiran Reddy – 20BAI1061

Submitted to

Dr. Rajarajeswari S,

Assosiate Professor Senior,

SCOPE, VIT Chennai

School of Computer Science and Engineering

ABSTRACT

The rise of blockchain technology has sparked an era of inventive advancements, opening doors for decentralized applications (DApps) and causing significant disruptions in various sectors such as finance, event management, and crowdfunding. This project introduces a comprehensive platform that merges these sectors, harnessing the potential of blockchain and smart contracts. MetaMask's capabilities are harnessed for secure authentication and transaction management, while the Remix IDE facilitates efficient smart contract development. By embracing DeFi (Decentralized Finance) principles, users can effortlessly partake in banking operations, event coordination, and crowdfunding endeavours, all within a decentralized and trust less setting. The application of blockchain technology guarantees transparency, immutability, and security, ushering in a fresh era of effective, user-centric, and decentralized solutions for the contemporary world.

Keywords: Blockchain, Decentralized Applications (DApps), DeFi (Decentralized Finance), MetaMask, Remix IDE, Event Management, Banking, Crowdfunding.

RELATED STUDY

Event Management Ticket Booking using Blockchain

[1]A blockchain-powered ticketing system in event management cuts out costly intermediaries, ensuring hosts receive full ticket proceeds. It also provides robust protection against ticket fraud and unauthorized entry by uniquely representing tickets as non-fungible tokens on the Ethereum blockchain. With a fixed ticket quantity, ownership securely changes hands upon customer payment, preventing any tampering. This unchangeable ownership transfer guarantees authenticity, enhancing security and trust. Storing event and ticket details on the blockchain ensures transparency and permanence, effectively deterring ticket fraud. Ultimately, this groundbreaking solution enables hosts to handle ticket sales effortlessly and securely, marking a significant advancement in the event management industry.

Blockchain based Secure Event Management System using NLP and RNN Algorithm

[2] The suggested system integrates blockchain, NLP, and RNN algorithms to establish a secure event management system. It tackles the issue of unreliable information exchange on social media, especially during incidents or disasters. By harnessing blockchain's unchangeable nature and decentralized structure, the system ensures heightened security and trustworthiness. NLP and RNN algorithms are utilized to efficiently process and analyse natural language data, crucial for detecting and confirming the authenticity and relevance of event information. The confirmed data is then securely stored and shared on the blockchain, guaranteeing a decentralized and tamper-resistant record. This comprehensive approach transforms event management, providing a highly dependable platform for secure information distribution.

Data immutability and event management via blockchain in the Internet of things

[3] The Internet of Things (IoT) plays a crucial role in various industries, utilizing cost-effective lightweight devices with limited resources. However, IoT networks often incorporate advanced elements like gateways and servers, leading to centralization and potential bottlenecks. This centralized structure poses challenges in maintaining data integrity, non-repudiation, and efficient event management. Blockchain technology addresses these issues by offering a decentralized ledger for secure data storage, ensuring integrity, immutability, and non-repudiation in IoT setups. Customized smart contracts further enhance event management by introducing decentralization and immutability. This architecture utilizes an Ethereum-based private blockchain (Quorum), featuring a unique ad-hoc smart contract and MQTT-based communication between sensor and actuator nodes. The proof-of-

concept successfully demonstrates functionality, scalability, and efficiency, particularly in forest fire risk detection. Performance tests reveal stability in processing up to 12.5 transactions per second for incoming data and distributing commands at 4 TPS, with potential for higher throughput depending on network conditions. The scheme exhibits polynomial message and time complexity, showcasing a promising approach for secure and efficient IoT systems.

Artificial Intelligence Applications for Event Management and Marketing

[4] Artificial Intelligence (AI) is playing a pivotal role in revolutionizing event management and marketing. This segment delves into its present-day applications, the influence it wields on the industry, and forthcoming trends. Significantly, AI's ascendancy is closely tied to the facilitation provided by big data. Within the event sector, there is a notable integration of robotic advancements, including telepresence robots, robotic concierges, and entertainment robots, alongside AI-driven digital assistants and chatbots. These innovations not only confer a competitive edge to stakeholders but also furnish invaluable insights for marketing efforts. They streamline manual processes through digitalization, elevate customer interactions, and amplify event participation while curbing costs. Moreover, AI fuels the genesis of fresh products and services, culminating in substantial value addition. The trajectory points towards a continuous surge in the adoption of AI in the event industry, promising even greater strides in the future.

Blockchain-Based NFT Ticketing System

[5] This research delves into the potential of Non-Fungible Tokens (NFTs) in event ticketing, addressing a previously unexplored area. It introduces a prototype NFT-based ticketing system and conducts a rigorous evaluation of its effectiveness. NFTs prove their capability to tokenize digital assets, thwart fraud, and exercise control over secondary market transactions. The system establishes a traceable ownership chain, ensuring ticket validity through blockchain encryption. NFT ticketing systems promise a safer, more efficient, and transparent ticket purchasing process, with the potential to reshape the industry. While still in its early stages, further growth and advancements are anticipated, promising enhanced user experiences and operational efficiency. NFT ticketing systems hold potential in various sectors beyond events, including transportation, museums, conferences, and theme parks. They also open avenues for revenue generation through limited-edition NFTs and loyalty programs. However, challenges such as scalability, compatibility with other blockchain systems, and managing price fluctuations in the secondary market remain areas of concern.

Enabling Blockchain Based SCM Systems with a Real Time Event Monitoring Function for Pre-emptive Risk Management

[6] This study centres on elevating supply chain management (SCM) through the integration of real-time event monitoring and blockchain technology. It tackles the challenge of swiftly identifying critical events, spanning from routine fluctuations to emergencies like earthquakes or epidemics, which could disrupt SCM operations. Current decision support systems face difficulties in promptly notifying users of such events, prompting the need for a pre-emptive risk management solution. The proposed approach entails embedding a real-time event monitoring system within SCM to enable the early sharing of emergency information. Moreover, to securely exchange confidential supply chain data among stakeholders, the adoption of blockchain-based SCM systems is advocated. This ensures transparency and traceability in the supply chain, upholding product quality and data privacy. The study melds real-time event detection using Twitter data with blockchain technology to amplify visibility and facilitate pre-emptive risk mitigation. Experimental findings reveal encouraging results, indicating potential avenues for further progress in agile supply chain practices.

Highly Secure Residents Life Event Management System Based on Blockchain by Hyperledger Fabric

[7] This article outlines the creation and deployment of a highly secure Residents Life Event Management System (RLEMS) utilizing blockchain technology, specifically Hyperledger Fabric. The blockchain system guarantees a reliable and secure data access environment by organizing data fragments into interconnected blocks using hash values. This proves especially valuable in regions with limited ICT infrastructure, such as many developing African countries. The switch from Multichain to Hyperledger Fabric was driven by the latter's suitability for government-level data management systems. The implementation also features a user-friendly web application interface developed with the Java web application framework, Prime Face. The study concludes that Hyperledger Fabric is better suited for this purpose than Multichain. Future endeavours will focus on enhancing processing speed, potentially through GPU parallel processing, and exploring the creation of a blockchain from scratch using Java.

A Hybrid Blockchain-Based Event Ticketing System

[8] This thesis advocates for a hybrid blockchain-powered event ticketing system that addresses critical issues like ticket forgery, scalping, and ensures privacy and information transparency. It leverages blockchain for transparent ticketing data and employs asymmetric encryption to safeguard privacy. Additionally, the system incorporates digital signatures to authenticate tickets and introduces a unique verification mechanism to combat scalping. Through comprehensive experiments and subsequent analysis, the system's performance is thoroughly evaluated. This hybrid blockchain-based ticketing system effectively resolves challenges related to forged ticket prevention, privacy protection, information transparency, traceability, and scalping prevention. Given the diverse nature of information shared in the

event ticketing system, tailored strategies for transparency, privacy, and information sharing policies are imperative, necessitating careful categorization based on specific user groups.

Ethereum Based Smart Contract for Event Management System

[9] The paper underscores the use of Ethereum, a prominent blockchain platform, and its native cryptocurrency, Ether (ETH), in the development of a smart contract tailored for event management systems. These smart contracts are self-executing programs housed within the blockchain, negating the need for a central governing authority. The research underscores the creation of a blockchain-based smart contract for event management using Solidity, deployed on the Rinke by test network. Moreover, it offers insights into the distinctions between smart contracts and conventional contracts. The Ethereum platform presents vast opportunities for integrating blockchain across a range of applications, signalling its potential for widespread adoption in everyday scenarios. This advancement represents a notable stride in the progress of blockchain technology, broadening its practical applications.

Real Estate Land Transaction System Using Blockchain

[10] This paper confronts the inefficiencies and security issues prevalent in global real estate management, particularly in India. It advocates for the adoption of a secure blockchain-based system to streamline and elevate the land registration process. Blockchain technology, renowned for its robustness and security, holds substantial potential for transforming the land industry. The paper introduces a blockchain-driven real estate management system, with a focus on instilling transparency, efficiency, and heightened security. It underscores the importance of decentralized data storage and its integration with the Ethereum Virtual Machine (EVM) to facilitate the creation of smart contracts in real estate management. The design and interaction mechanisms for both estate owners and users, operating within the framework of a smart contract, are elucidated. The system ensures safeguarded transaction storage through distributed blockchain, drastically minimizing susceptibility to hacking. The delineated functions for initiating, creating, modifying, or terminating a smart contract aim to provide a more engaging, user-friendly, and visualized contracting process. This solution presents a pragmatic approach to addressing real-world challenges in real estate management.

Application of Cryptocurrencies using Blockchain for E-Commerce Online Payment

[11] This study explores the practical implementation of cryptocurrencies in online e-commerce payments, leveraging the decentralized nature of Blockchain technology. Cryptocurrencies are gaining popularity for their unique attributes like decentralized consensus, anonymity, and shared ledger, making them a fitting match for Blockchain. However, it's crucial to recognize that, like any payment method, Blockchain and cryptocurrencies are susceptible to security breaches. The research addresses these concerns

through a questionnaire distributed to 100 educated professionals. The findings reveal a cautious approach to adopting cryptocurrencies for online payments due to perceived security risks. Primary worries include potential attacks on cryptocurrency wallets, 51% attacks, double spending, and other technical vulnerabilities. The absence of a universal international framework and regulation emerges as a critical factor contributing to potential abuses of Blockchain technology, underscoring the importance of global cooperation in establishing robust regulations and policies.

SECURE BANKING SYSTEM USING BLOCK CHAIN TECHNOLOGY

[12] The paper introduces secure banking systems using blockchain technology, emphasizing its role as a trust-enabling distributed ledger. Given the swift technological progress, a comprehensive grasp of core technologies and their data processing capabilities is crucial. Private blockchains, which involve authenticated parties, are a focal point. The assessment covers distributed ledger, cryptography, consensus protocol, and smart contracts in both current and research frameworks. The BLOCKBENCH benchmarking framework is employed to gauge private blockchain performance in data processing workloads, evaluating Ethereum, Parity, and Hyperledger Fabric. Notably, significant performance discrepancies emerge between blockchain and database systems. The paper advocates research avenues to narrow this performance gap, proposing a hybrid blockchain development approach that combines proof-of-work and stake-based consensus mechanisms to bolster security.

Decentralised Blockchain Technology: Application in Banking Sector

[13] This paper delves into the integration of decentralized blockchain technology within the banking sector, representing a significant departure from traditional methods towards a digital, immutable, and distributed ledger system. The inherent nature of blockchain, operating on a distributed peer-to-peer network, provides an effective solution for recording and managing transactions within banking systems. It introduces critical features including transparency, robustness, auditability, and security. The central aim of this paper is to implement these functionalities into a distributed banking system through blockchain, harmonizing it with existing practices. Moreover, the paper acknowledges potential hurdles in the adoption of blockchain technology and outlines its prospective applications in the banking sector. This transition is poised to revolutionize banking operations by elevating efficiency, transparency, and security measures.

Decentralized Banking Application using Block chain Technology

[14] This passage introduces the idea of utilizing blockchain technology for a decentralized banking application. Blockchain serves as a secure, distributed ledger for digital currency transactions. It ensures transparency and transaction validation by granting all network participants access to the latest encrypted ledger version. The blockchain ledger records all

Bitcoin transactions in a chronological, unchangeable data structure made up of blocks. Each block contains a timestamp and references the previous block. Bitcoin functions on a peer-to-peer, permissionless network, allowing users to connect, validate transactions, and generate new blocks. Satoshi Nakamoto's 2008 research paper on Bitcoin's design brought a groundbreaking solution to cryptographic challenges, laying the foundation for this innovative form of currency.

A Blockchain-based Decentralized Data Storage and Access Framework for PingER

[15] This research introduces a decentralized data storage and access framework for PingER, a global Internet performance measurement project. It leverages blockchain technology and Distributed Hash Tables (DHT) to break free from reliance on a centralized repository. In this proposed system, file metadata is recorded on the blockchain, while the actual files are stored off-chain using DHT across multiple locations through a peer-to-peer network of PingER Monitoring Agents. This architecture ensures decentralized storage, distributed processing, and efficient data retrieval for the PingER system. By employing a permissioned blockchain, the framework eliminates the need for a centralized repository, decentralizing PingER and reducing dependency on centralized computing resources. This approach enables scalable and sustainable implementation of the project, ultimately enhancing Internet performance monitoring to meet the demands of current and future technologies.

Decentralized Banking: The Future of Finance?

[16] In this paper, the advantages of decentralized banking systems are explored, encompassing heightened transparency, diminished risk of fraudulent activities, and decreased transaction expenses. The authors contend that decentralized systems have the potential to promote a more stable financial landscape by diminishing the impact of prominent financial institutions.

Blockchain-Based Decentralized Banking: A Regulatory and Economic Analysis

[17] The present investigation assesses the economic and regulatory ramifications of decentralized banking systems utilizing blockchain technology. The authors assert that such decentralized systems can effectively lower transaction expenses, enhance financial accessibility, and augment transparency. However, they emphasize the necessity of fitting regulations to tackle concerns like money laundering and terrorist financing.

Decentralized Finance (DeFi): An Overview

[18] In this paper, an outline of the decentralized finance (DeFi) ecosystem is presented, encompassing decentralized banking systems. The authors examine the advantages and hurdles associated with DeFi, incorporating considerations about scalability and interoperability. Moreover, they emphasize DeFi's capacity to potentially revolutionize conventional financial systems.

Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets

[19] This study examines the emergence of decentralized finance and its potential impact on traditional financial systems. The author discusses the benefits of decentralized systems, including increased transparency and reduced transaction costs, and explores the challenges of implementing such systems.

"Decentralized Finance: Overview and Opportunities"

[20] In this paper, a comprehensive review of the decentralized finance ecosystem is presented, encompassing decentralized banking systems. The authors delve into the possible advantages of DeFi, including heightened financial inclusivity and decreased transaction expenditures. Additionally, they examine the obstacles and prospects linked with the integration of decentralized systems.

Crowdfunding Using Blockchain Technology

[21]The project explores crowdfunding as an innovative and disruptive financial system that has seen substantial growth. Crowdfunding presents an affordable way to access capital, expand the audience for innovative ideas and business ventures, decrease investment risk, and furnish customized financing solutions for a range of projects. It offers an alternative to conventional borrowing and is accessible to both individuals and businesses. The primary objective of the project is to inform aspiring young entrepreneurs about crowdfunding as an alternative means of funding. It concentrates on elucidating the characteristics and advantages of crowdfunding, as well as the obstacles that may motivate young entrepreneurs to utilize crowdfunding platforms. Online crowdfunding enables individuals to gather financial support for their projects through digital donations, which project managers can then utilize to bring their projects to fruition. The text also touches upon the relatively recent introduction of blockchain technology in crowdfunding, underscoring the notion that it may require some time for blockchain-based crowdfunding applications to become widely recognized and comprehended. The proposed blockchain-based crowdfunding platform aims to deliver transparent transactions within a decentralized framework. In brief, the project's purpose is to impart knowledge and enhance understanding of crowdfunding as a viable funding option among young entrepreneurs. It underscores the potential benefits of

crowdfunding and underscores the role of blockchain technology in enhancing the transparency and security of crowdfunding transactions.'

Blockchain-Based Crowdfunding Application

[22] The content explores how blockchain technology has the potential to reshape the perception and management of valuable and sensitive data. It points out that blockchain offers features like timestamping, irreversible logging, and smart contracts, which can automate transactions, boost efficiency, and remove the need for third-party validation. The text also underscores the significance of transparency and security, particularly in crowdfunding platforms. It introduces the idea of a crowdfunding web application built on blockchain, which not only enables investments but also ensures returns and fosters transparency between backers and startups. The application is described as having three user types: Administrators, Backers, and Start-ups, each with their specific roles and capabilities.

Blockchain based crowdfunding systems

[23] Blockchain technology has progressed beyond its original application within the realm of cryptocurrency, experiencing growing adoption in a diverse range of industries. It is envisioned that blockchain will emerge as a widespread and indispensable tool for facilitating online transactions in the forthcoming era. Crowdfunding platforms, which have often grappled with issues related to regulatory compliance and fraudulent activities, stand to benefit significantly from the incorporation of blockchain solutions. This initiative aims to tackle these challenges head-on by harnessing Ethereum smart contracts to automate the intricate processes inherent in crowdfunding. This automation not only serves as a safeguard against fraudulent activities but also guarantees that projects adhere to their designated timelines. The primary objective is the establishment of a crowdfunding platform that boasts heightened transparency and operates on a decentralized basis. Furthermore, our future endeavours encompass the integration of ERC-223 tokens into these intelligent contracts. This integration promises greater operational efficiency, reduced gas consumption, and improved handling of incoming token transactions. In doing so, we aspire to elevate the functionality and security of blockchain-based crowdfunding to new heights.

Crowd-Funding Using Blockchain Technology

[24] The content delves into the evolution of crowdfunding, originally a means for the public to support creative projects, now extended to investing in start-ups. It underscores the deficiencies in current crowdfunding platforms, notably the absence of a Donor Guarantee Policy and donor control over their contributions. The paper suggests leveraging blockchain technology to establish a secure and transparent crowdfunding platform. Its objective is to offer interactive functionalities for campaign creation, donations, and request approval, streamlining the participation and tracking process for both campaign creators and donors. Blockchain records all transactions as blocks, mitigating risks associated with traditional crowdfunding, such as fees and project failures. The paper concludes that blockchain-driven crowdfunding is a relatively nascent concept within the ICT community, having successfully

implemented Solidity code and deployed it on the blockchain. Acknowledging ongoing legal and technical challenges, it expresses optimism about blockchain's future role in crowdfunding, anticipating further enhancements and the expansion of the proposed research.

Role of Blockchain Technology in Crowdfunding (International Banking and Finance)

[25] Crowdfunding represents a modern and inventive fundraising approach where individuals seek financial support for a wide range of ventures, encompassing both profit-driven and social initiatives, typically offering future products or equity in return. This method harnesses the power of internet and social media platforms to link investors with entrepreneurs, playing a pivotal role in mobilizing capital. The paper explores the influence of technology on crowdfunding and delves into the various crowdfunding platforms that have arisen recently. Blockchain, renowned for its transparency and ability to foster trust, is emphasized as a disruptive force in the crowdfunding arena. New technologies, particularly crowdfunding platforms integrated with blockchain technology, have the potential to bolster the credibility of projects and attract substantial investments. Despite inherent challenges, crowdfunding serves as a valuable resource for startups, innovators, and creators, with blockchain poised to further streamline the process, enhancing transparency and accessibility in the future. In summary, emerging technologies, notably blockchain, are set to revolutionize crowdfunding, addressing current issues and promising a bright future for both investors and entrepreneurs.

CROWDSOURCE FUNDING SOLUTION BASED ON BLOCKCHAIN

[26] This paper introduces a novel application designed to revolutionize traditional crowdfunding platforms like Kickstarter by eliminating the need for intermediaries. It provides a direct avenue for entrepreneurs to secure funding and for investors to back projects, utilizing tokens backed by secure and unalterable smart contracts on the Ethereum blockchain. The application's development utilizes tools like Remix IDE and Visual Studio code, simplifying fundraising processes and enhancing transparency. In a world increasingly embracing decentralized solutions in the era of Web 3.0, this project seeks to empower users by removing intermediaries, making it effortless for them to create and invest in campaigns that pique their interest. While alternative blockchain platforms such as EOS, Stellar, Cardano, and NEO offer a broader array of language options and configurations, their potential has yet to be fully validated. Among these alternatives, EOS shows promise, and if it proves superior to Ethereum, there is potential for the project to transition to EOS in the future.

Crowd Funding Platform Using Blockchain

[27] Crowdfunding has revolutionized fundraising, utilizing online platforms and social media to connect entrepreneurs and organizations with global investors and donors. Blockchain technology has enhanced trust and transparency in crowdfunding, making it more inclusive and efficient. The rise of distributed autonomous organizations (DAOs) has disrupted traditional fundraising by directing more capital towards blockchain-based campaigns, eliminating centralized intermediaries' limitations. The proposed web-based crowdfunding system in this paper, powered by blockchain and Ethereum smart contracts, addresses key issues seen in conventional platforms. It ensures trust, transparency, and secure transaction storage. The use of Infura ensures a reliable connection between the web system and the Ethereum network, ensuring secure transaction recording. This innovative crowdfunding method removes fees and provides a secure fundraising avenue for startups, potentially revolutionizing the fundraising landscape.

Smart Contract and Blockchain for Crowdfunding Platform

[28] In the context of the COVID-19 pandemic, numerous organizations are mobilizing funds to assist local governments in securing additional resources for those in need. Trust plays a pivotal role for all parties involved in this fundraising endeavour, encompassing funders, fundraising platforms, and fundraisers. The author explores the integration of blockchain technology and smart contracts into the prevalent crowdfunding processes. The study's findings suggest that smart contracts based on blockchain technology can be effectively applied in the three primary crowdfunding process schemes. Smart contracts, driven by blockchain technology, not only enhance trust but also streamline the fundraising processes considerably. However, the adoption of smart contracts using blockchain technology can incur substantial costs if organizations decide to undertake this independently. It's noteworthy that the use of cryptocurrencies, commonly associated with smart contract service providers, may not be legally recognized by all governments.

Evaluation of Blockchain-Based Crowdfunding Campaign Success Factors Based on VASMA-L Criteria Weighting Method

[29] This research delves into the identification and ranking of key success factors that impact investors' choices in supporting blockchain-based crowdfunding campaigns. To achieve this, an online survey was conducted to gather insights from experts, employing the VASMA-L methodology to assess the significance of these factors. These success factors were categorized into two groups: those relevant to both traditional and blockchain-based crowdfunding and those specific to the blockchain context. The study's findings highlight that investors tend to prioritize factors common to both traditional and blockchain-based crowdfunding when making investment decisions. The most influential factors in their choices are the industry, early investments, and the percentage of retained equity or tokens. Conversely, criteria such as the use of Ethereum, KYC/pre-registration processes, and the availability and content of white papers were considered less crucial. In the future, further

research could explore additional criteria groups or aspects related to blockchain-based crowdfunding, broadening the utility of this weighting methodology.

Blockchain Based Crowdfunding Platform using Ethereum

[30] Blockchain technology, initially designed for cryptocurrencies, is now being adopted across various industries, particularly in the realm of online transactions. Crowdfunding platforms are among the beneficiaries of this technology. Current issues plaguing the global crowdfunding landscape include a lack of regulation, fraudulent campaigns, and project delays. To tackle these problems, this project proposes the integration of smart contracts into crowdfunding platforms, automating processes and bolstering security.With blockchain, transactions within crowdfunding platforms gain transparency, instilling confidence in contributors. Smart contracts empower contributors to monitor fund utilization, ensuring transparency in expenditure requests. This blockchain-powered platform guarantees transaction transparency, assuring users that their funds are in safe hands, while also recording all transactions and associated account addresses to reduce fraud risks.

DETAILED EXPLANATION OF THE SYSTEM

Banking System Smart Contract

Smart Contract Overview

This smart contract implements a simple banking system on the Ethereum blockchain. It allows for the creation of accounts, deposit and withdrawal of funds, processing cheques, opening and closing accounts, and managing loans.

Variables

- **count:** Tracks the total number of accounts.
- **l:** Tracks the total number of loans.
- **AccountMap:** Maps account numbers to AccountHolder structures.
- **PendingChequeMap:** Maps cheque numbers to Cheque structures for pending cheques.
- **AllChequeMap:** Maps cheque numbers to Cheque structures for all cheques.
- **LoanMap:** Maps loan numbers to Loan structures.
- **custAcc:** Array to store account numbers of customers.
- **loans:** Array to store loan numbers.
- **PendingCheques:** Array to store pending cheque numbers.
- **ProcessedCheques:** Array to store processed cheque numbers.

Functions

- **OpenAccount:** Allows the owner to open an account for a customer.
- **Deposit:** Allows users to deposit funds into their accounts.
- **Withdrawal:** Allows users to withdraw funds from their accounts.
- **setChequeData:** Allows the owner to set cheque data for processing.
- **ProcessCheque:** Processes a cheque, transferring funds from the issuer to the receiver.
- **CloseAccount:** Allows the owner to close an account.
- **GetLoan:** Allows the owner to provide a loan to an account.
- **PayLoan:** Allows users to pay off their loans.
- **getLoanData:** Retrieves loan details.
- **getAccountData:** Retrieves account details.
- **getChequeData:** Retrieves cheque details.
- **getAccounts:** Retrieves the list of account numbers.
- **getLoans:** Retrieves the list of loan numbers.
- **getPendingCheques:** Retrieves the list of pending cheque numbers.
- **getProcessedCheques:** Retrieves the list of processed cheque numbers.

Main Functionalities:

- **Account Management:**
  - Users can open an account.
  - Users can deposit and withdraw funds.
  - Users can check their account balance.
- **Cheque Processing:**
  - Users can issue and process cheques.
  - Cheque status changes from "Pending" to "Processed" upon processing.
  - Funds are transferred from the issuer to the receiver upon processing.
- **Loan Management:**
  - Users can apply for a loan (if within the loan limit).
  - Users can pay off their loans.

Components:

- **AccountHolder struct:** Represents account holders with account information.
- **Cheque struct:** Stores information about cheques.
- **Loan struct:** Stores loan-related information.
- **AccountMap:** Mapping of account numbers to AccountHolder structs.
- **PendingChequeMap:** Mapping of cheque numbers to pending Cheque structs.
- **AllChequeMap:** Mapping of cheque numbers to all Cheque structs.
- **LoanMap:** Mapping of loan numbers to Loan structs.

Functions to manage account, cheques, loans, and handle transactions.

Event Management Smart Contract

Smart Contract Overview

This smart contract enables event management on the Ethereum blockchain. It allows event organizers to create events, sell tickets, and transfer tickets to other addresses.

Variables

- **events:** Maps event IDs to Event structures.
- **tickets:** Maps addresses and event IDs to the number of tickets owned by each address for a specific event.
- **nextId:** Tracks the ID for the next event.

Functions

- **createEvent:** Allows event organizers to create an event.
- **buyTicket:** Allows users to buy tickets for an event.
- **transferTicket:** Allows users to transfer tickets to another address.

Main Functionalities:

- **Event Creation:**
    - Event organizers can create events with details like name, date, price, and ticket count.
- **Ticket Management:**
    - Users can purchase tickets for events.
    - Users can transfer their tickets to other addresses.
- **Components:**
    - **Event struct:** Represents an event with relevant details.
    - **events:** Mapping of event IDs to Event structs.
    - **tickets:** Mapping of addresses and event IDs to ticket quantities.

Functions to create events, purchase tickets, and transfer tickets.

Crowdfunding Smart Contract

Smart Contract Overview

This smart contract facilitates crowdfunding campaigns on the Ethereum blockchain. It allows users to contribute funds to a campaign, request a refund, and vote on expenditure requests.

Variables

- **contributors:** Maps addresses to the amount they've contributed.
- **manager:** Stores the address of the campaign manager.
- **minimumContribution:** Specifies the minimum contribution allowed.
- **deadline:** Stores the timestamp indicating the deadline for the campaign.
- **target:** Specifies the target amount to be raised.
- **raisedAmount:** Tracks the total amount raised so far.
- **noOfContributors:** Tracks the total number of contributors.
- **requests:** Maps request numbers to Request structures.
- **numRequests:** Tracks the total number of requests.

Functions

- **sendEth:** Allows users to contribute Ether to the campaign.
- **getContractBalance:** Returns the contract's current balance.
- **refund:** Allows users to request a refund if the campaign deadline has passed.
- **createRequests:** Allows the manager to create expenditure requests.
- **voteRequest:** Allows contributors to vote on expenditure requests.
- **makePayment:** Allows the manager to make a payment based on approved requests.

Main Functionalities:

- **Campaign Setup:**
    - Campaign managers can set the target amount and campaign deadline.
- **Fundraising:**
    - Users can contribute funds to the campaign.
- **Request Management:**
    - Campaign managers can create expenditure requests.
    - Contributors can vote on these requests.
    - Approved requests can be processed for payment.

Components:

- **Request struct:** Represents an expenditure request.
- **contributors:** Mapping of addresses to their contributions.
- **requests:** Mapping of request numbers to Request structs.

Functions for contributing funds, requesting refunds, creating and voting on requests, and making payments.

These are the core functionalities and components for each of the provided smart contracts. Each contract serves a distinct purpose, implementing the features necessary for the specified domain (banking, event management, crowdfunding).

USE OF BLOCK CHAIN IN OUR SYSTEM

- **Introduction to Blockchain and Its Innovative Potential:**

The project involves by underlining the transformative capacity inherent in blockchain technology. It recognizes the substantial innovative strides that blockchain has initiated, particularly through the advent of decentralized applications (DApps). This sets the foundation for comprehending the fundamental paradigm shift experienced across various sectors.

- **Integration of Decentralized Applications (DApps) Across Multiple Domains:**

The project strives to seamlessly incorporate DApps into diverse domains such as event management, banking, and crowdfunding. This indicates that the proposed platform aspires to be a versatile ecosystem, offering tailored solutions for these varied sectors.

- **Harnessing Blockchain and Smart Contracts:**

The abstract underscores the utilization of blockchain technology and smart contracts to fuel the platform. This underscores the significance of decentralized and tamper-resistant ledgers, ensuring transparency, security, and immutability of transactions and data.

- **Incorporation of MetaMask for Enhanced Security:**

The inclusion of MetaMask, a renowned cryptocurrency wallet and gateway to blockchain applications, is highlighted. Its integration signifies a commitment to security and user-friendly experience, enabling users to securely authenticate and manage transactions within the platform.

- **Streamlined Development with Remix IDE:**

The mention of Remix IDE indicates an optimized and efficient process for smart contract development. This selection of the development environment suggests a dedication to robust and well-organized code, which is paramount for a dependable and functional blockchain platform.

- **Integration of DeFi Principles for Inclusive Finance:**

The reference to DeFi (Decentralized Finance) implies the amalgamation of DeFi principles into the platform. DeFi is cantered on delivering financial services in a decentralized manner, advocating for financial inclusion and accessibility for all users.

➢ **<u>Empowering Users with Trust less Solutions:</u>**

By underscoring the merits of blockchain technology, the abstract implies that users will have access to solutions devoid of the need for absolute trust. Trust lessness in this context signifies that users can place their reliance on the protocol and smart contracts, bypassing the necessity to trust a centralized authority."

KEY AREAS OF BLOCKCHAIN IN OUR SYSTEM

In the context of the smart contracts of our project which are Banking System, Event Management, and Crowdfunding, blockchain technology is used to enhance security, transparency, and trust in various aspects of these applications.

1. Immutable Ledger and Historical Transaction Records

Within the banking system, blockchain functions as an unchangeable ledger, meticulously documenting every transaction, event, and account creation in a secure and immutable manner. Each transaction manifests as a block, intricately linked to its predecessor via cryptographic hashes, forming an irreversible and transparent timeline. This process ensures a steadfast and lucid record of all operations within the system.

2. Decentralization and Widely Distributed Data

The pivotal role of blockchain's decentralization is underscored in this project. In contrast to conventional centralized systems, there exists no singular point of authority or vulnerability. Every participant (node) in the network possesses a replica of the entire blockchain. In the realm of event management and crowdfunding, this decentralization guarantees a broad dispersion of data, augmenting dependability and resilience against failures.

3. Security Fortified by Cryptography

Blockchain employs sophisticated cryptographic methodologies to fortify the security of both data and transactions. Transactions are securely sealed through cryptography, rendering unauthorized access nearly insurmountable. Particularly within the banking system, this ensures the safeguarding of sensitive financial data, maintaining its privacy and integrity.

4. Transparency and Verifiability

All transactions are laid bare and accessible to every participant, championing transparency. In applications pertaining to event management and crowdfunding, this transparency assures that ticket sales, fund contributions, and fund allocation remain open to public scrutiny. Users have the ability to authenticate transactions, amplifying trust and fostering a culture of accountability.

5. Smart Contracts Enabling Automation and Trust

Smart contracts epitomize self-executing agreements with contractual clauses directly encoded into lines of code. Within the provided applications, smart contracts streamline processes such as ticket sales, fund transfers, loan administration, and cheque processing. This automation instils trust and efficiency, given that execution is dictated by code, eliminating the necessity of intermediaries.

6. Decentralized Finance (DeFi) Principles Promoting Financial Inclusion

Through the infusion of DeFi principles, the project facilitates users in engaging with banking-like activities directly on the blockchain. Users can securely obtain loans and manage their finances, fostering financial inclusion without reliance on traditional banking systems. This becomes especially potent in regions where traditional financial services are limited.

7. Tokenization for Event Tickets

In the context of event management, event tickets can be embodied as unique non-fungible tokens (NFTs) on the blockchain. Tokenization guarantees the distinctiveness and security of each ticket. Users can verify the legitimacy of their tickets, while organizers can mitigate ticket replication and fraudulent activities.

8. Augmented Security Measures

Blockchain's cryptographic attributes and consensus mechanisms elevate security. Particularly within the banking system, sensitive information is shielded through encryption, substantially reducing the risk of unauthorized access and fraudulent activities. This proves critical in upholding user trust and system integrity.

9. Fund Management and Transparent Transactions in Crowdfunding

Within the crowdfunding application, blockchain ensures transparent fund management. Requests for expenditures and the utilization of funds are meticulously recorded on the blockchain, enabling contributors to monitor and validate the usage of their funds. This cultivates trust and upholds accountability in the crowdfunding process.

10. Interoperability for Seamless Integration

Blockchain's interoperability facilitates seamless integration with other blockchain-based applications or platforms. In this project, it paves the way for users to access an array of services without undergoing repetitive authentication processes, elevating the user experience and expanding the potential ecosystem of the project.

In essence, blockchain technology forms the bedrock of this project, furnishing a secure, lucid, and efficient infrastructure for the banking system, event management, and crowdfunding applications. Its decentralized and immutable essence, combined with smart contracts and cryptographic security, presents distinct advantages critical for the prosperity and efficacy of the project.

IMPACT OF BLOCKCHAIN IN OUR SYSTEM

Advantages:

1) **Robust Security and Immovability:** Blockchain employs advanced cryptographic hashing and encryption, ensuring a high level of security. Once data is logged, it becomes practically unalterable, significantly enhancing transaction and data security.
   Example: In the banking system, any transaction recorded on the blockchain remains immutable, guaranteeing the integrity and security of financial transactions and account balances.

2) **Transparency and Fostered Trust:** Transactions within blockchain networks are openly visible to all participants, promoting transparency. This fosters trust as users can independently verify transactions and account for all activities.
   Example: Within event management, users have complete visibility into the ticket sales and transfers history, ensuring a fair and transparent ticketing process that builds trust with attendees.

3) **Decentralization for Enhanced Control:** Blockchain operates on a decentralized network, eliminating the necessity of a central authority. This diminishes the risk of a single point of failure, bolsters system resilience, and provides users with greater control.
   Example: In crowdfunding, smart contracts manage funds on the blockchain, removing the need for intermediaries. This decentralization ensures secure fund handling without reliance on a central authority.

4) **Streamlined Processes and Increased Efficiency:** Smart contracts enable automation of processes, reducing manual intervention and associated errors. This results in heightened efficiency and expedited processing of transactions or activities.
   Example: In the banking system, processes like loan approvals and fund transfers can be automated using smart contracts, streamlining operations and minimizing delays.

5) **Enhanced Accountability and Traceability:** The blockchain records all transactions and actions comprehensively, ensuring accountability and traceability for every activity conducted within the system.
   Example: In crowdfunding, transactions related to fund utilization are meticulously recorded, enabling contributors to monitor how their funds are spent, ensuring accountability from project managers.

Disadvantages:

1) **Scalability Challenges:** Blockchain networks can face challenges in scalability as they expand, resulting in slower transaction times and increased fees. This can limit application scalability, especially during peak usage.
   Example: During a highly anticipated event with a surge in ticket sales, the blockchain may struggle to process a large number of ticket purchases promptly.

2) **Transaction Costs:** Transactions on the blockchain involve transaction fees, which can escalate during network congestion. For microtransactions or applications with frequent transactions, these costs can accumulate.
   Example: In the banking system, frequent daily transactions by users can lead to accumulated transaction fees associated with each transfer.

3) **Irreversible Transactions:** Once confirmed, transactions on the blockchain are irreversible. In cases of errors or fraud, rectifying the transactions is challenging, potentially resulting in financial loss or incorrect data.
   Example: If an erroneous fund transfer occurs in the banking system, it cannot be undone or corrected, potentially leading to financial loss.

4) **Privacy Concerns:** Despite secure and encrypted transactions, blockchain's transparency can raise privacy concerns, as all transactions are visible to all participants, which may be unsuitable for sensitive data.
   Example: In the banking system, exposing account balances and transaction history to all users could raise significant privacy issues.

5) **Regulatory Uncertainty:** The regulatory landscape for blockchain is in constant flux and varies across jurisdictions. Adhering to evolving legal and regulatory requirements can be challenging and may impact the adoption and operation of blockchain-based systems.
   Example: Alterations in regulations concerning financial transactions could impact the functioning and acceptance of blockchain-based banking systems.

In summary, while blockchain offers considerable advantages such as robust security, transparency, and decentralization, it also presents challenges like scalability issues and irreversible transactions. A comprehensive understanding of these pros and cons is pivotal for the effective implementation of blockchain in the project and the mitigation of potential drawbacks.

IMPLEMENTATION PROCEDURE

Our project leverages a public and permissionless blockchain, specifically the Ethereum blockchain. It utilizes smart contracts for automated transactions, tokenization for representing assets, and employs the Proof of Work (PoW) consensus mechanism for transaction validation and network security. These features collectively enable transparency, decentralization, and secure execution of transactions and contracts within the Ethereum blockchain.

This is implementation for the provided Solidity contracts in Remix IDE using MetaMask. Let's start with each contract.

Banking System Contract

The Banking System contract is a simple implementation of a banking system with functionalities like opening accounts, deposits, withdrawals, cheques, and loans.

Event Management Contract

The Event Management contract allows the creation and management of events, including ticketing and ticket transfers.

Crowd Funding Contract

The Crowd Funding contract enables crowdfunding campaigns with features such as contributors, refunds, and request creation for fund utilization.

- **Implementation Procedure in Remix IDE using MetaMask**

➢ Open Remix IDE:
➢ Go to Remix IDE.
➢ Copy and paste the contracts into separate files (e.g., BankingSystem.sol, EventManagement.sol, CrowdFunding.sol) in the Remix IDE.

- **Compile Contracts:**

➢ Select the appropriate Solidity compiler version for each contract.
➢ Click on the "Solidity Compiler" tab in Remix and compile each contract by clicking "Compile".

- **Deploy Contracts:**

➢ Click on the "Deploy & Run Transactions" tab in Remix.
➢ Select the contract you want to deploy from the dropdown.
➢ Deploy the contract by clicking the "Deploy" button.

- **Interact with Contracts:**

➢ After deploying, you can interact with the contract using the provided functions.
➢ Set parameters accordingly for each function (e.g., OpenAccount in the Banking contract, createEvent in the Event Management contract, etc.).

- Use MetaMask to fund transactions and interact with the contracts.

Examples of Parameters for Calling Functions

- **Banking System Contract:**
  - Open an account:
  - Function: OpenAccount
  - Parameters: "John Doe", "Main Branch", 1234567890
- **Event Management Contract:**
  - Create an event:
  - Function: createEvent
  - Parameters: "Concert", 1665700000 (for a future date), 100000000000000000 wei (0.1 ether), 100 (ticket count)
- **Crowd Funding Contract:**
  - Contribute to the crowdfunding campaign:
  - Function: sendEth
  - Parameters: (send an amount higher than or equal to the minimumContribution)

We have to use the appropriate data types and units (wei, ether) when sending transactions and calling functions. Additionally, we should be connected to the correct network in MetaMask when interacting with the contracts.

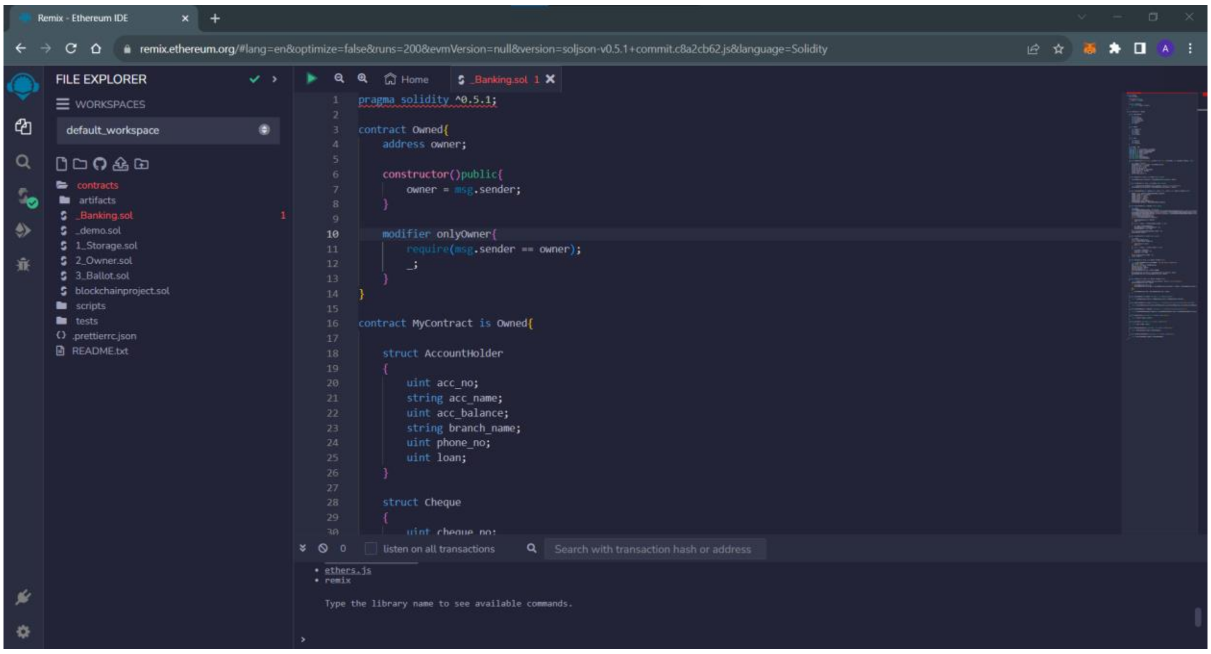
IMPLEMENTATION DETAILS (CODE & SCREENSHOTS)

- **Banking System Contract:**
  - **CODE:**

```
pragma solidity ^0.5.1;

contract Owned{
    address owner;

    constructor()public{
        owner = msg.sender;
    }

    modifier onlyOwner{
        require(msg.sender == owner);
        _;
    }
}

contract MyContract is Owned{

    struct AccountHolder
    {
        uint acc_no;
        string acc_name;
        uint acc_balance;
        string branch_name;
        uint phone_no;
        uint loan;
    }

    struct Cheque
    {
        uint cheque_no;
        uint issuer;
        uint reciver;
        uint amount;
        string status;
    }

    struct Loan
    {
        uint loan_no;
        uint acc_no;
        uint loan_amt;
    }

    uint count = 110;
    uint l = 0;
    mapping(uint => AccountHolder) AccountMap;
    mapping(uint => Cheque) PendingChequeMap;
    mapping(uint => Cheque) AllChequeMap;
    mapping(uint => Loan) LoanMap;
    uint [] private custAcc;
    uint [] private loans;
    uint [] private PendingCheques;
    uint [] private ProcessedCheques;

    function OpenAccount(string memory _accname,string memory
_branchName, uint _phoneNo) onlyOwner  public
    {
        uint _accno = ++count;
        AccountHolder storage account = AccountMap[_accno];
        account.acc_name = _accname;
        account.acc_no = _accno;
        account.acc_balance = 0;
        account.branch_name = _branchName;
        account.phone_no = _phoneNo;
        account.loan = 0;
        custAcc.push(_accno) -1;
    }

    function Deposit(uint accno, uint amount) public payable
    {
        AccountMap[accno].acc_balance =
AccountMap[accno].acc_balance + amount;
    }

     function Withdrawl(uint accno, uint amount) public payable
    {
        require((amount)<=AccountMap[accno].acc_balance, "Balance is
not sufficient");
        AccountMap[accno].acc_balance =
AccountMap[accno].acc_balance - amount;
    }

    function setChequeData(uint _chequeno,uint _issuer, uint
_reciver, uint _amount) onlyOwner public
    {
        Cheque storage cheques = PendingChequeMap[_chequeno];
        cheques.cheque_no = _chequeno;
        cheques.issuer = _issuer;
        cheques.reciver = _reciver;
        cheques.amount = _amount;
        cheques.status = "Pending";
        PendingCheques.push(_chequeno) -1;
        AllChequeMap[_chequeno] = PendingChequeMap[_chequeno];
    }

    function ProcessCheque(uint chequeno) public payable
    {
        uint index;
        AllChequeMap[chequeno].status = "Processed";
        require((PendingChequeMap[chequeno].amount)<=(AccountMap[Pen
dingChequeMap[chequeno].issuer].acc_balance), "Balance is not
sufficient");
        AccountMap[PendingChequeMap[chequeno].issuer].acc_balance =
AccountMap[PendingChequeMap[chequeno].issuer].acc_balance -
PendingChequeMap[chequeno].amount;
        AccountMap[PendingChequeMap[chequeno].reciver].acc_balance =
AccountMap[PendingChequeMap[chequeno].reciver].acc_balance +
PendingChequeMap[chequeno].amount;
        ProcessedCheques.push(chequeno) -1;
        delete (PendingChequeMap[chequeno]);
        for(uint i=0;i<PendingCheques.length;i++)
        {
            if(PendingCheques[i] == chequeno)
            index = i;
        }
        for (uint i = index; i < PendingCheques.length - 1; i++)
        {
            uint temp = PendingCheques[i];
            PendingCheques[i] = PendingCheques[i + 1];
            PendingCheques[i + 1] = temp;
        }
        delete PendingCheques[PendingCheques.length - 1];
        PendingCheques.length--;
    }

    function CloseAccount(uint accno) public payable
    {
        uint index;
        delete (AccountMap[accno]);
        for(uint i=0;i<custAcc.length;i++)
        {
            if(custAcc[i] == accno)
            index = i;
        }
        for (uint i = index; i < custAcc.length - 1; i++)
        {
            uint temp = custAcc[i];
            custAcc[i] = custAcc[i + 1];
            custAcc[i + 1] = temp;
        }
        delete custAcc[custAcc.length - 1];
        custAcc.length--;
    }

    function GetLoan(uint accno, uint amount) onlyOwner public
    {
        require(((AccountMap[accno].loan)<50000), "you loan limit is
maxed out");
        uint loanno = ++l;
        Loan storage CustLoan = LoanMap[loanno];
        CustLoan.acc_no = accno;
        CustLoan.loan_amt = amount;
        loans.push(loanno) -1;
        if((AccountMap[accno].loan + amount)<=50000)
        {
        AccountMap[accno].acc_balance =
AccountMap[accno].acc_balance + amount;
        AccountMap[accno].loan = AccountMap[accno].loan + amount;
        }
    }

    function PayLoan(uint accno, uint amount) onlyOwner public
    {
        require((amount)<=(AccountMap[accno].acc_balance), "Balance
is not sufficient");
        if(AccountMap[accno].loan < amount)
        {
            AccountMap[accno].loan = 0;
            AccountMap[accno].acc_balance =
AccountMap[accno].acc_balance + ( amount - AccountMap[accno].loan );
        }
        else
        {
            AccountMap[accno].loan = AccountMap[accno].loan -
amount;
        }
    }

    function getLoanData(uint loanno) view public
returns(uint,uint,uint)
    {
        return (LoanMap[loanno].loan_no, LoanMap[loanno].acc_no,
LoanMap[loanno].loan_amt);
    }

    function getAccountData(uint accno) view public returns(string
memory,uint,uint,string memory,uint,uint)
    {
        return
(AccountMap[accno].acc_name,AccountMap[accno].acc_no,AccountMap[accn
o].acc_balance,AccountMap[accno].branch_name,AccountMap[accno].phone
_no, AccountMap[accno].loan);
    }

    function getChequeData(uint chequeno) view public
returns(uint,uint,uint,uint,string memory)
    {
        return (AllChequeMap[chequeno].cheque_no,
AllChequeMap[chequeno].issuer, AllChequeMap[chequeno].reciver,
AllChequeMap[chequeno].amount, AllChequeMap[chequeno].status);
    }

    function getAccounts() view public returns(uint, uint[] memory)
    {
        return (custAcc.length, custAcc);
    }

    function getLoans() view public returns(uint, uint[] memory)
    {
        return (loans.length, loans);
    }

    function getPendingCheques() view public returns(uint, uint[]
memory)
    {
        return (PendingCheques.length, PendingCheques);
    }

     function getProseccedCheques() view public returns(uint, uint[]
memory)
    {
        return (ProcessedCheques.length, ProcessedCheques);
    }
}
```

➢ SCREENSHOTS:

1) Banking Smart Contract Code in Remix IDE



remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js&language=Solidity

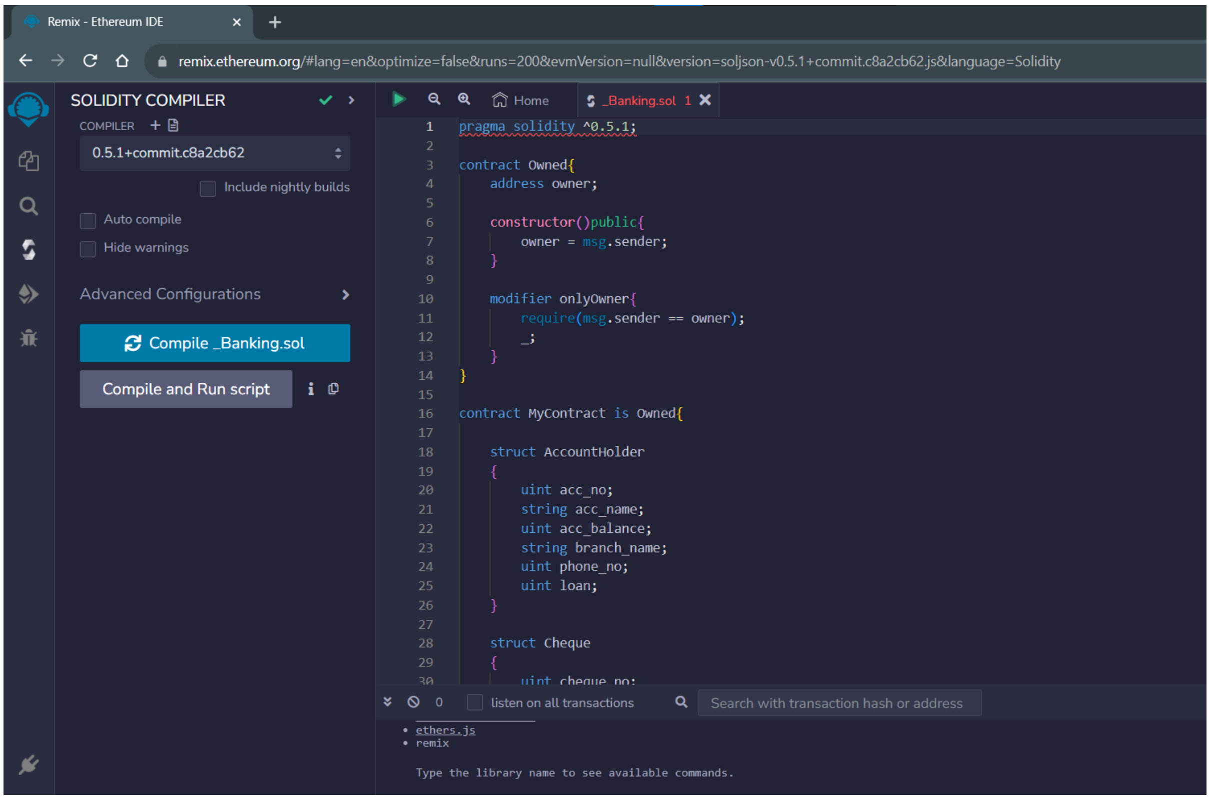
FILE EXPLORER — WORKSPACES: default_workspace

```
contracts
artifacts
_Banking.sol
_demo.sol
1_Storage.sol
2_Owner.sol
3_Ballot.sol
blockchainproject.sol
scripts
tests
.prettierrc.json
README.txt
```

```solidity
pragma solidity ^0.5.1;

contract Owned{
    address owner;

    constructor()public{
        owner = msg.sender;
    }

    modifier onlyOwner{
        require(msg.sender == owner);
        _;
    }
}

contract MyContract is Owned{

    struct AccountHolder
    {
        uint acc_no;
        string acc_name;
        uint acc_balance;
        string branch_name;
        uint phone_no;
        uint loan;
    }

    struct Cheque
    {
        uint cheque_no;
```

```
• ethers.js
• remix

  Type the library name to see available commands.
```

2) Banking Smart Contract Code compiling with 0.5.1+commit.c8a2cb62



remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js&language=Solidity

SOLIDITY COMPILER — COMPILER: 0.5.1+commit.c8a2cb62

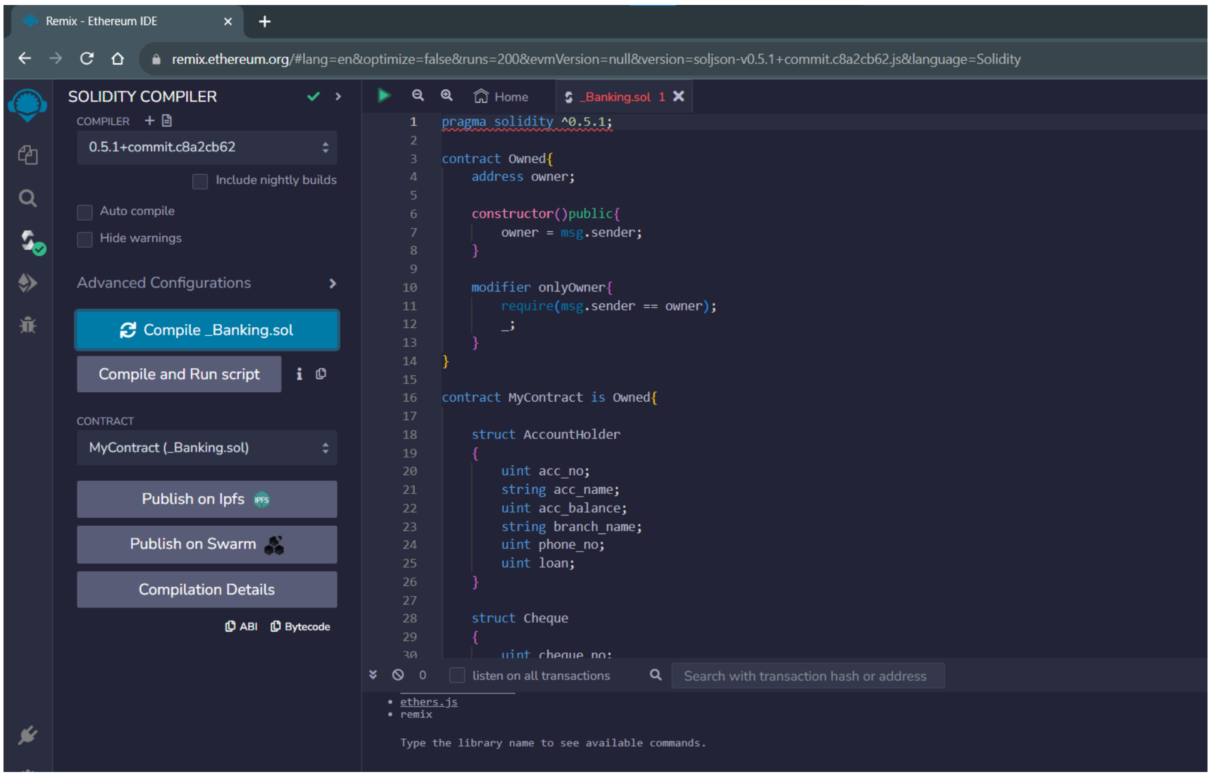
Include nightly builds: ☐
Auto compile: ☐
Hide warnings: ☐

```solidity
pragma solidity ^0.5.1;

contract Owned{
    address owner;

    constructor()public{
        owner = msg.sender;
    }

    modifier onlyOwner{
        require(msg.sender == owner);
        _;
    }
}

contract MyContract is Owned{

    struct AccountHolder
    {
        uint acc_no;
        string acc_name;
        uint acc_balance;
        string branch_name;
        uint phone_no;
        uint loan;
    }

    struct Cheque
    {
        uint cheque_no;
```

```
• ethers.js
• remix

  Type the library name to see available commands.
```

Advanced Configurations
Compile _Banking.sol
Compile and Run script
listen on all transactions
Search with transaction hash or address

3) Compilation done successfully


Remix - Ethereum IDE

remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js&language=Solidity

SOLIDITY COMPILER

COMPILER: 0.5.1+commit.c8a2cb62

Include nightly builds

Auto compile

Hide warnings

Advanced Configurations

Compile _Banking.sol

Compile and Run script

CONTRACT: MyContract (_Banking.sol)

Publish on Ipfs

Publish on Swarm

Compilation Details

ABI Bytecode

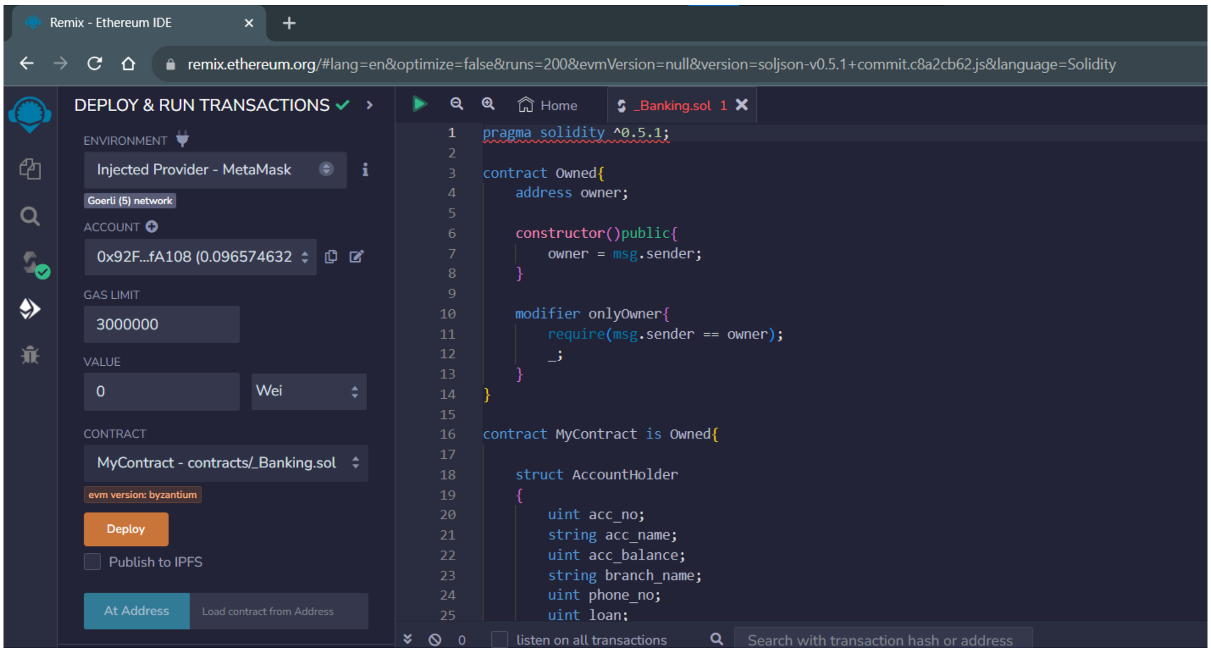
_Banking.sol

```
pragma solidity ^0.5.1;

contract Owned{
    address owner;

    constructor()public{
        owner = msg.sender;
    }

    modifier onlyOwner{
        require(msg.sender == owner);
        _;
    }
}

contract MyContract is Owned{

    struct AccountHolder
    {
        uint acc_no;
        string acc_name;
        uint acc_balance;
        string branch_name;
        uint phone_no;
        uint loan;
    }

    struct Cheque
    {
        uint cheque_no;
```

listen on all transactions   Search with transaction hash or address

```
• ethers.js
• remix

Type the library name to see available commands.
```

4) Deploying with Metamask waalet


Remix - Ethereum IDE

remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js&language=Solidity

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

Goerli (5) network

ACCOUNT: 0x92F...fA108 (0.096574632

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: MyContract - contracts/_Banking.sol

evm version: byzantium

Deploy

Publish to IPFS

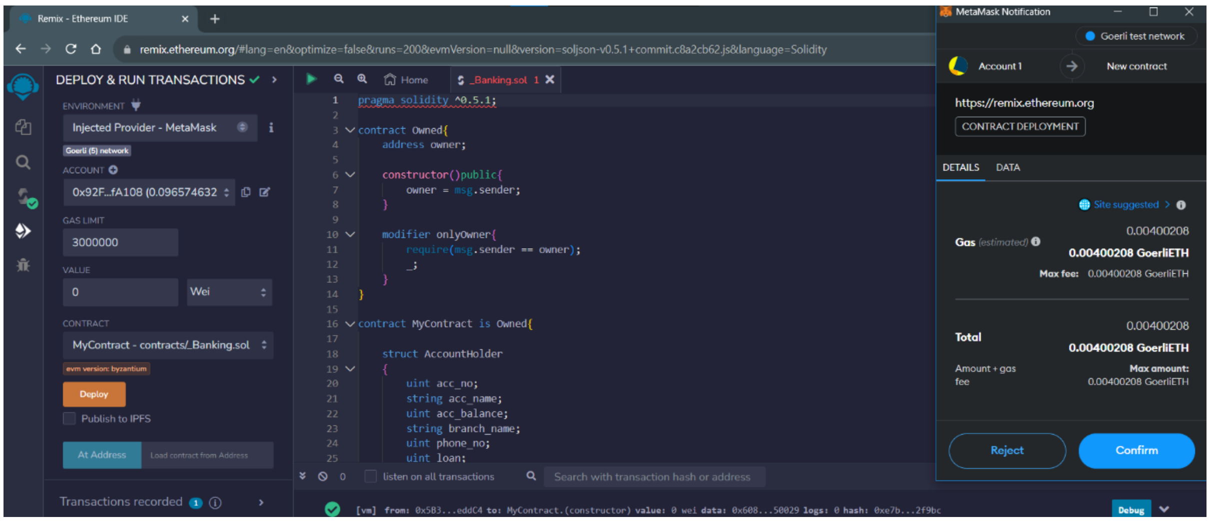
At Address   Load contract from Address

_Banking.sol

```
pragma solidity ^0.5.1;

contract Owned{
    address owner;

    constructor()public{
        owner = msg.sender;
    }

    modifier onlyOwner{
        require(msg.sender == owner);
        _;
    }
}

contract MyContract is Owned{

    struct AccountHolder
    {
        uint acc_no;
        string acc_name;
        uint acc_balance;
        string branch_name;
        uint phone_no;
        uint loan;
```

listen on all transactions   Search with transaction hash or address

5) Meta mask wallet opens successfully. (We are paying the gas price or transaction fee which is 0.004 GoerliETH)



remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js&language=Solidity

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask (Goerli (5) network)

ACCOUNT: 0x92F...fA108 (0.096574632

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: MyContract - contracts/_Banking.sol (evm version: byzantium)

```
pragma solidity ^0.5.1;

contract Owned{
    address owner;

    constructor()public{
        owner = msg.sender;
    }

    modifier onlyOwner{
        require(msg.sender == owner);
        _;
    }
}

contract MyContract is Owned{

    struct AccountHolder
    {
        uint acc_no;
        string acc_name;
        uint acc_balance;
        string branch_name;
        uint phone_no;
        uint loan;
```

```
[vm] from: 0x5B3...eddC4 to: MyContract.(constructor) value: 0 wei data: 0x608...50029 logs: 0 hash: 0xe7b...2f9bc
```

MetaMask Notification

Goerli test network

Account 1 → New contract

https://remix.ethereum.org

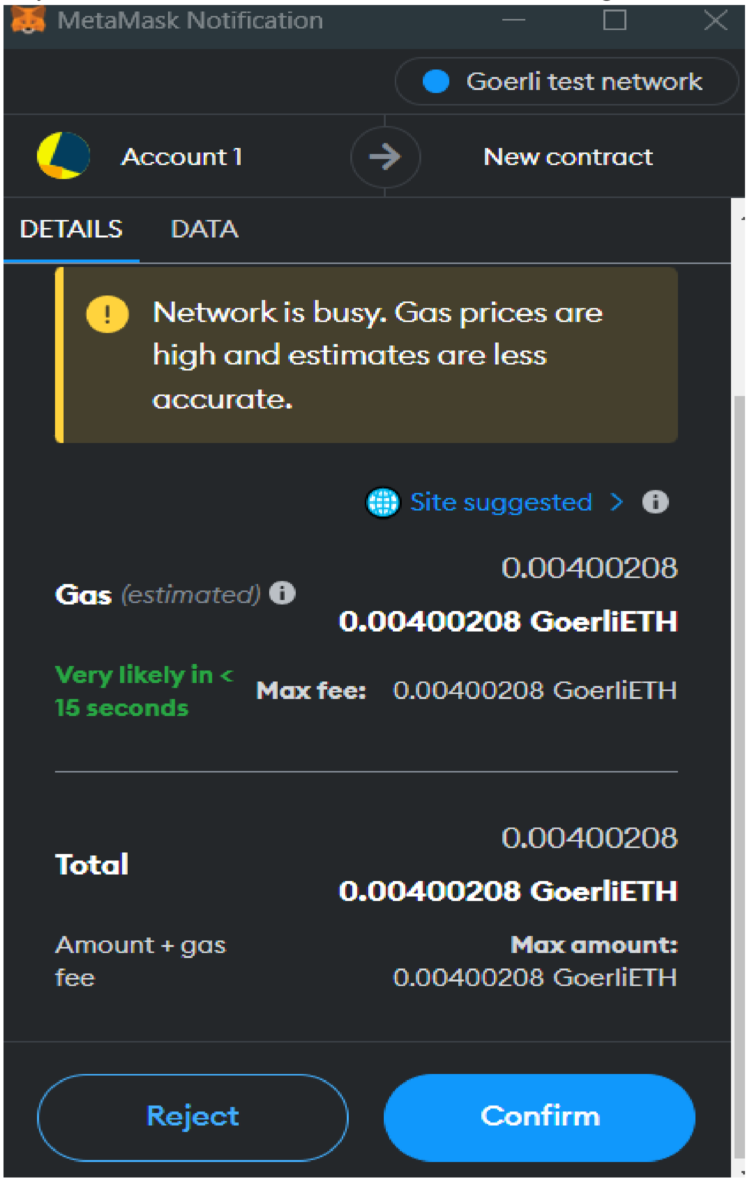
CONTRACT DEPLOYMENT

Gas (estimated): 0.00400208 — 0.00400208 GoerliETH — Max fee: 0.00400208 GoerliETH

Total: 0.00400208 — 0.00400208 GoerliETH

Amount + gas fee — Max amount: 0.00400208 GoerliETH

6) Payment Confirmation Notification for creating the smartcontract.



MetaMask Notification

Goerli test network

Account 1 → New contract

DETAILS DATA

Network is busy. Gas prices are high and estimates are less accurate.

Site suggested >

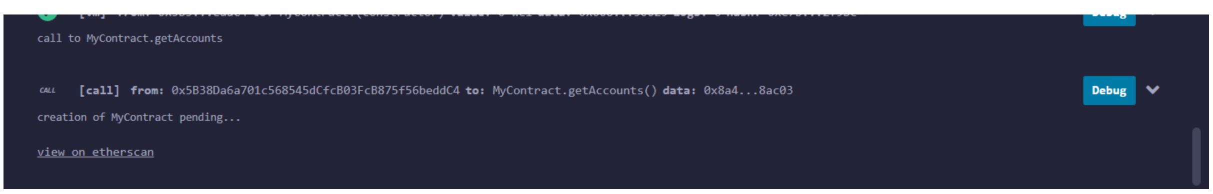
Gas *(estimated)* 0.00400208
0.00400208 GoerliETH

Very likely in < 15 seconds
Max fee: 0.00400208 GoerliETH

Total 0.00400208
0.00400208 GoerliETH

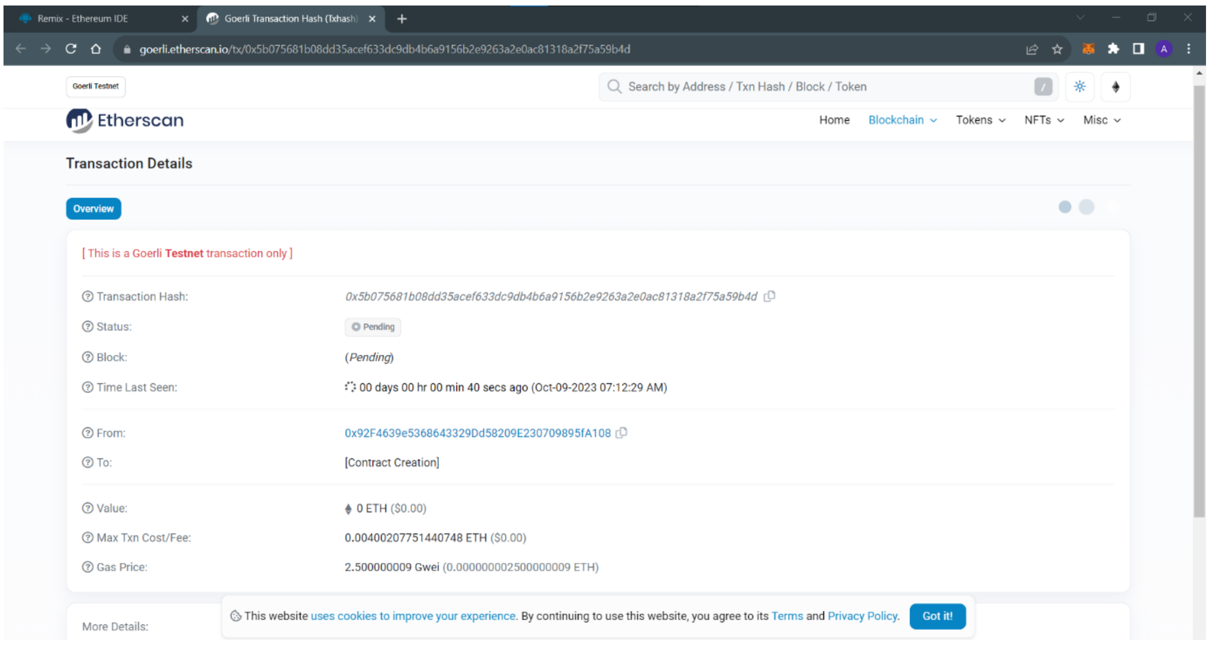
Amount + gas fee
Max amount: 0.00400208 GoerliETH

Reject Confirm



```
call to MyContract.getAccounts

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MyContract.getAccounts() data: 0x8a4...8ac03
creation of MyContract pending...

view on etherscan
```

7) Transaction Details on Etherscan


Remix - Ethereum IDE
Goerli Transaction Hash (Txhash)
goerli.etherscan.io/tx/0x5b075681b08dd35acef633dc9db4b6a9156b2e9263a2e0ac81318a2f75a59b4d
Goerli Testnet
Search by Address / Txn Hash / Block / Token
Etherscan
Home
Blockchain
Tokens
NFTs
Misc
Overview

Transaction Details

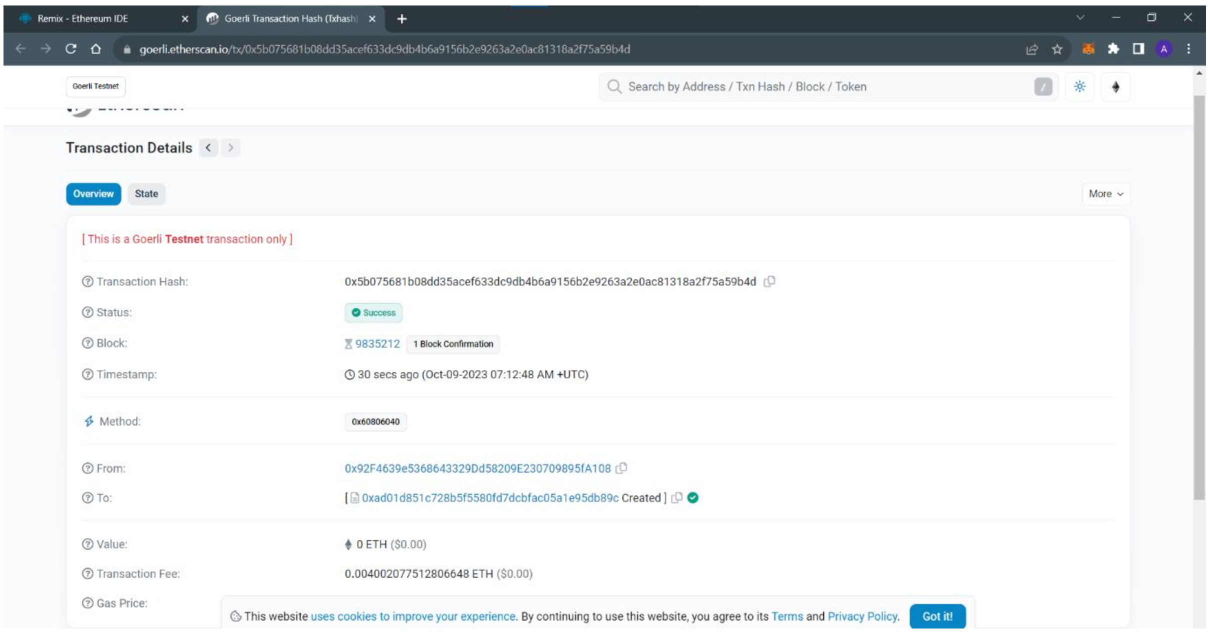
[This is a Goerli **Testnet** transaction only]

Transaction Hash:	*0x5b075681b08dd35acef633dc9db4b6a9156b2e9263a2e0ac81318a2f75a59b4d*
Status:	Pending
Block:	*(Pending)*
Time Last Seen:	00 days 00 hr 00 min 40 secs ago (Oct-09-2023 07:12:29 AM)
From:	0x92F4639e5368643329Dd58209E230709895fA108
To:	[Contract Creation]
Value:	0 ETH ($0.00)
Max Txn Cost/Fee:	0.00400207751440748 ETH ($0.00)
Gas Price:	2.500000009 Gwei (0.000000002500000009 ETH)

More Details:

This website uses cookies to improve your experience. By continuing to use this website, you agree to its Terms and Privacy Policy. Got it!

8) Successful Confirmation Message.


Remix - Ethereum IDE
Goerli Transaction Hash (Txhash)
goerli.etherscan.io/tx/0x5b075681b08dd35acef633dc9db4b6a9156b2e9263a2e0ac81318a2f75a59b4d
Goerli Testnet
Search by Address / Txn Hash / Block / Token
Overview
State
More

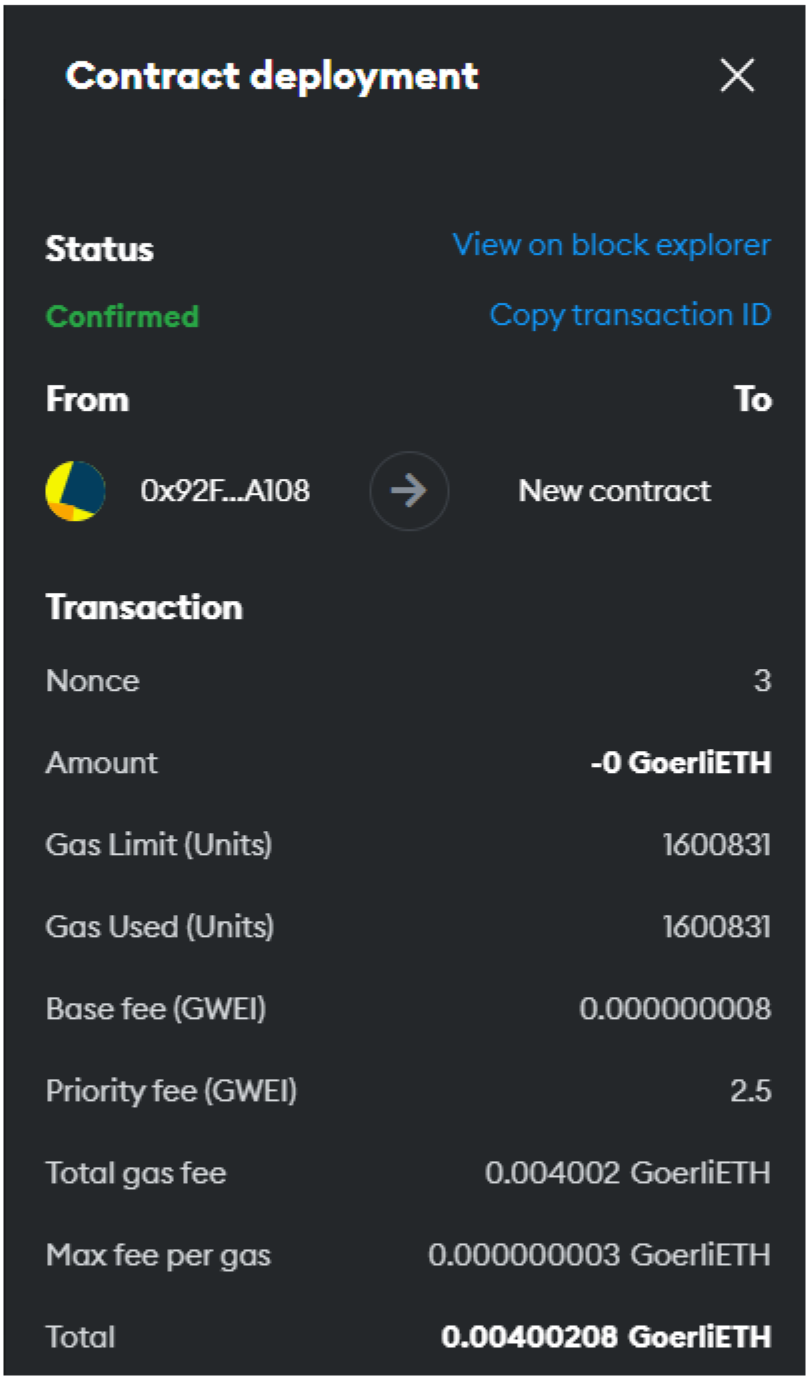
Transaction Details

[This is a Goerli **Testnet** transaction only]

Transaction Hash:	0x5b075681b08dd35acef633dc9db4b6a9156b2e9263a2e0ac81318a2f75a59b4d
Status:	Success
Block:	9835212 1 Block Confirmation
Timestamp:	30 secs ago (Oct-09-2023 07:12:48 AM +UTC)
Method:	0x60806040
From:	0x92F4639e5368643329Dd58209E230709895fA108
To:	[0xad01d851c728b5f5580fd7dcbfac05a1e95db89c Created]
Value:	0 ETH ($0.00)
Transaction Fee:	0.004002077512806648 ETH ($0.00)
Gas Price:	

This website uses cookies to improve your experience. By continuing to use this website, you agree to its Terms and Privacy Policy. Got it!

9) Transaction details in GoerliETJ



Contract deployment

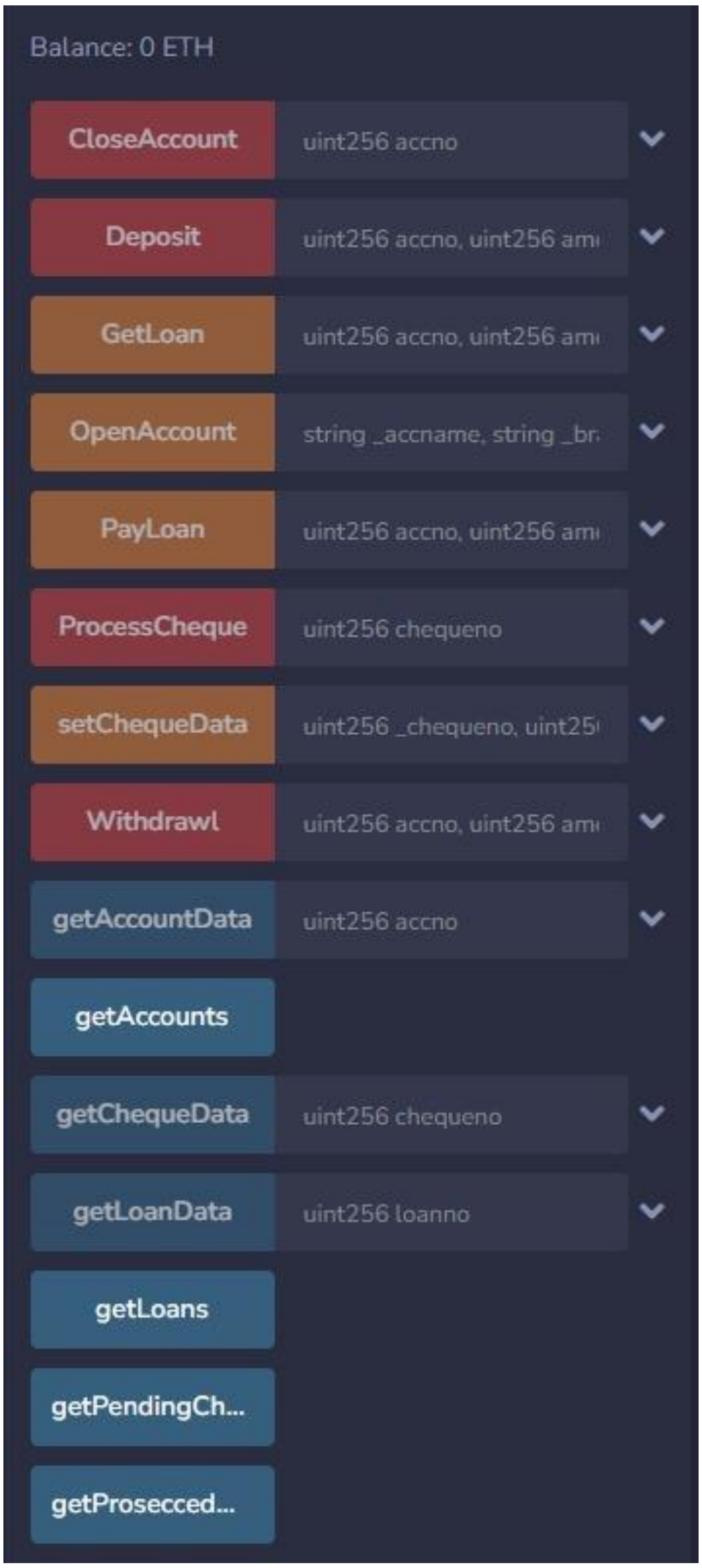
Status — Confirmed

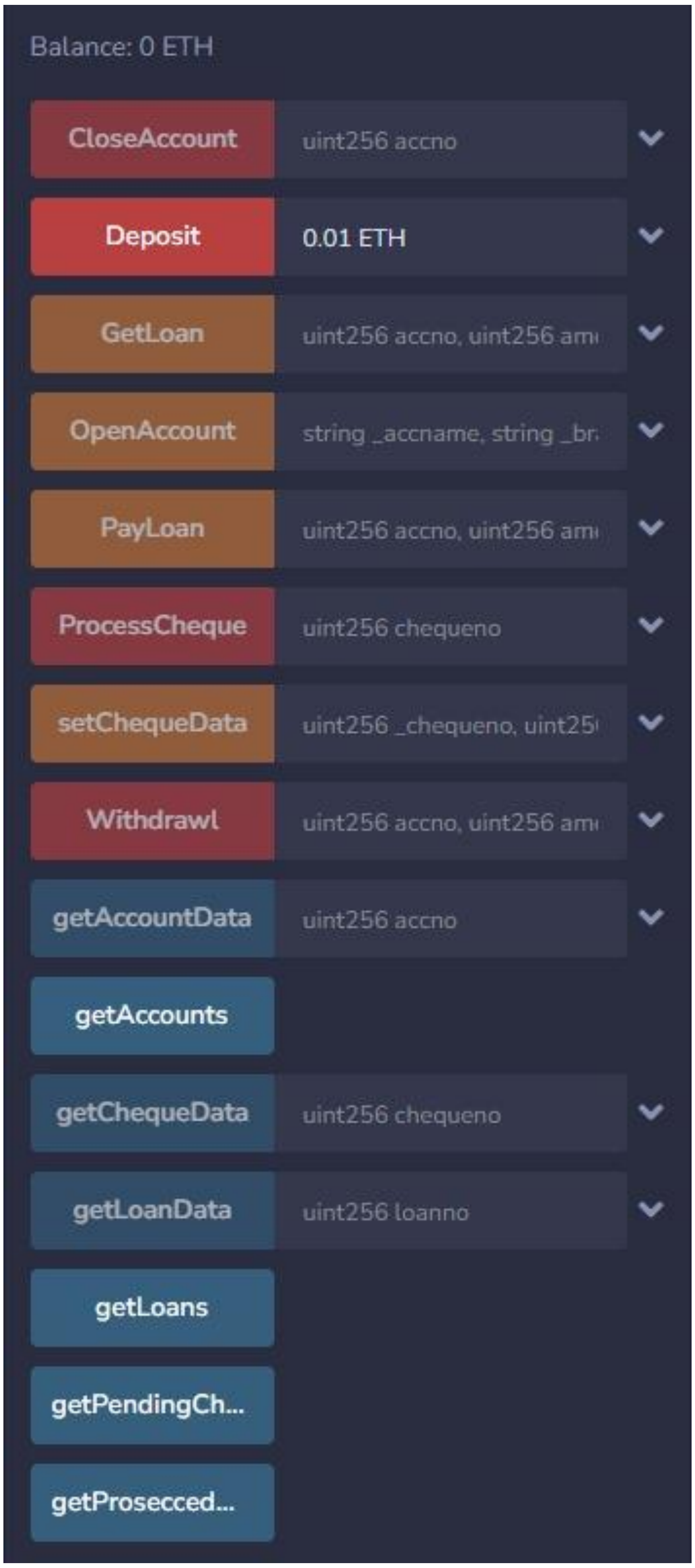
From	To
0x92F...A108	New contract

Transaction

Nonce	3
Amount	**-0 GoerliETH**
Gas Limit (Units)	1600831
Gas Used (Units)	1600831
Base fee (GWEI)	0.000000008
Priority fee (GWEI)	2.5
Total gas fee	0.004002 GoerliETH
Max fee per gas	0.000000003 GoerliETH
Total	**0.00400208 GoerliETH**

10) Parameter details Of our Banking SmartContract (But we need real Ethereum to do transactions here)





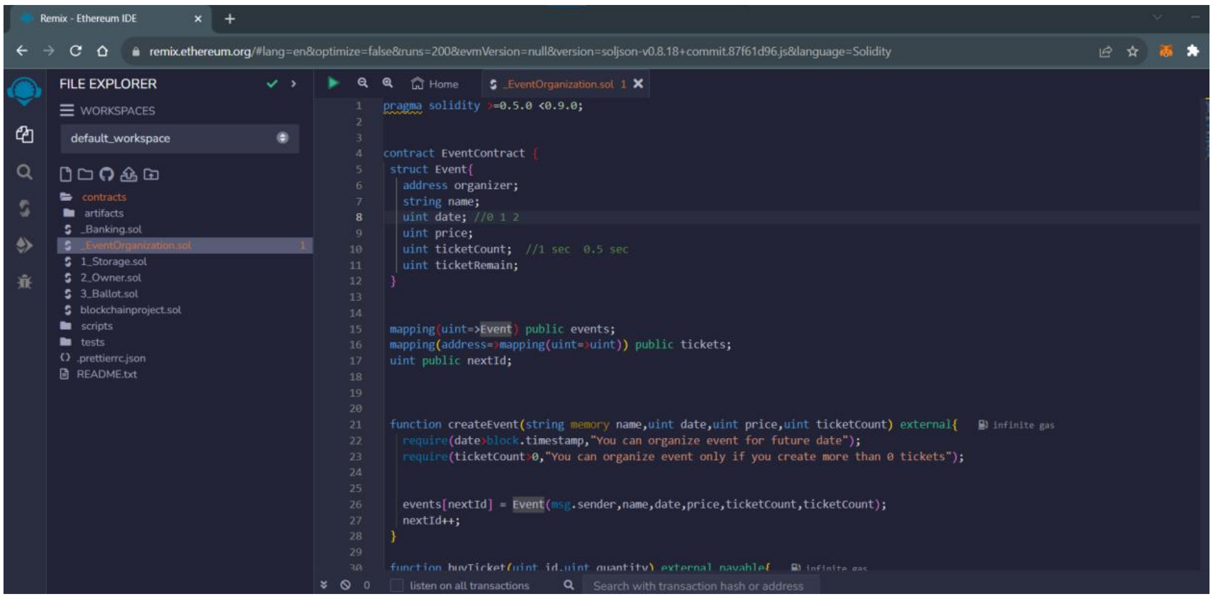
We can pay loans , do deposits , make payments , can do withdrawals using this application by entering the preferred values.

- **<u>Event Management Contract:</u>**
    - **CODE:**

```
pragma solidity >=0.5.0 <0.9.0;


contract EventContract {
 struct Event{
   address organizer;
   string name;
   uint date; //0 1 2
   uint price;
   uint ticketCount;  //1 sec  0.5 sec
   uint ticketRemain;
 }


 mapping(uint=>Event) public events;
 mapping(address=>mapping(uint=>uint)) public tickets;
 uint public nextId;



 function createEvent(string memory name,uint date,uint price,uint
ticketCount) external{
   require(date>block.timestamp,"You can organize event for future
date");
   require(ticketCount>0,"You can organize event only if you create
more than 0 tickets");


   events[nextId] =
Event(msg.sender,name,date,price,ticketCount,ticketCount);
   nextId++;
 }

 function buyTicket(uint id,uint quantity) external payable{
   require(events[id].date!=0,"Event does not exist");
   require(events[id].date>block.timestamp,"Event has already
occured");
   Event storage _event = events[id];
   require(msg.value==(_event.price*quantity),"Ethere is not
enough");
   require(_event.ticketRemain>=quantity,"Not enough tickets");
   _event.ticketRemain-=quantity;
   tickets[msg.sender][id]+=quantity;
 }
 function transferTicket(uint id,uint quantity,address to) external{
   require(events[id].date!=0,"Event does not exist");
   require(events[id].date>block.timestamp,"Event has already
occured");
   require(tickets[msg.sender][id]>=quantity,"You do not have enough
tickets");
   tickets[msg.sender][id]-=quantity;
   tickets[to][id]+=quantity;
 }
}
```

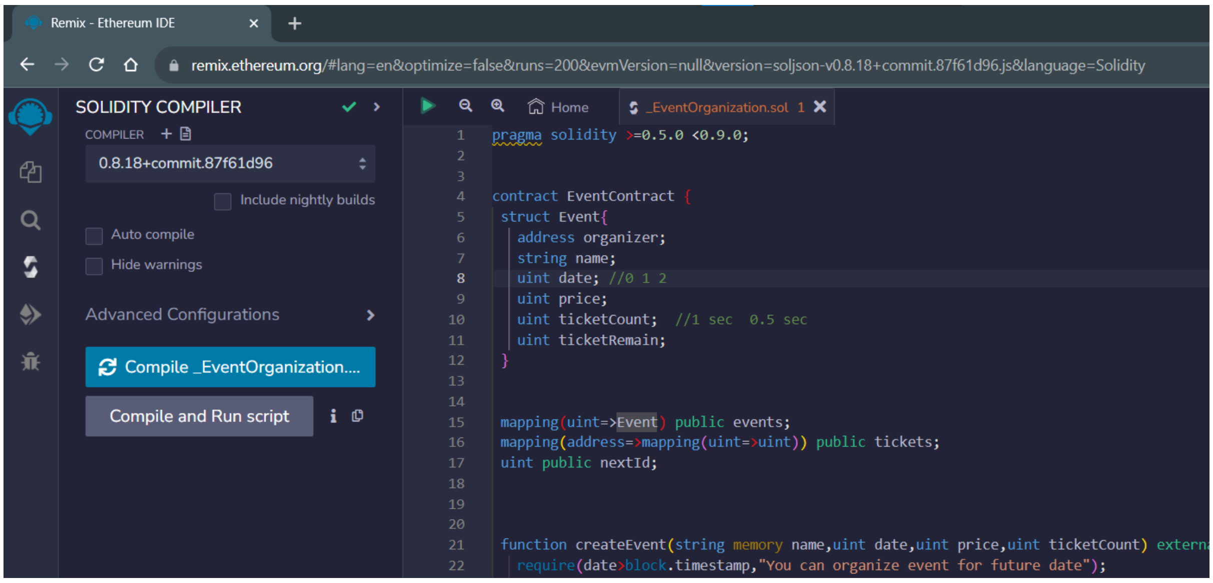
➢ SCREENSHOTS:

1) The Event Management Code in Remix IDE

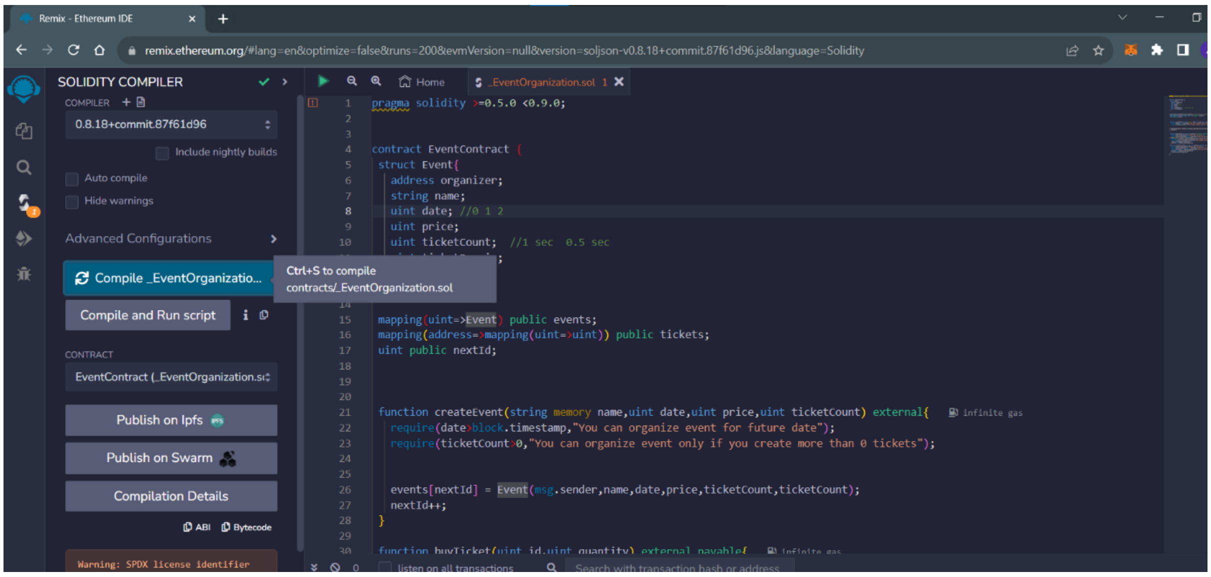


```
pragma solidity >=0.5.0 <0.9.0;


contract EventContract {
 struct Event{
   address organizer;
   string name;
   uint date; //0 1 2
   uint price;
   uint ticketCount;  //1 sec  0.5 sec
   uint ticketRemain;
 }


 mapping(uint=>Event) public events;
 mapping(address=>mapping(uint=>uint)) public tickets;
 uint public nextId;



 function createEvent(string memory name,uint date,uint price,uint ticketCount) external{    infinite gas
   require(date>block.timestamp,"You can organize event for future date");
   require(ticketCount>0,"You can organize event only if you create more than 0 tickets");


   events[nextId] = Event(msg.sender,name,date,price,ticketCount,ticketCount);
   nextId++;
 }

 function buyTicket(uint id,uint quantity) external payable{    infinite gas
```

2) Compiling the event management contract



```
pragma solidity >=0.5.0 <0.9.0;


contract EventContract {
 struct Event{
   address organizer;
   string name;
   uint date; //0 1 2
   uint price;
   uint ticketCount;  //1 sec  0.5 sec
   uint ticketRemain;
 }


 mapping(uint=>Event) public events;
 mapping(address=>mapping(uint=>uint)) public tickets;
 uint public nextId;



 function createEvent(string memory name,uint date,uint price,uint ticketCount) externa
   require(date>block.timestamp,"You can organize event for future date");
```


Remix - Ethereum IDE

remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js&language=Solidity

SOLIDITY COMPILER

COMPILER: 0.8.18+commit.87f61d96

Include nightly builds

Auto compile

Hide warnings

Advanced Configurations

Compile _EventOrganizatio...

Ctrl+S to compile
contracts/_EventOrganization.sol

Compile and Run script

CONTRACT: EventContract (_EventOrganization.sol)

Publish on Ipfs

Publish on Swarm

Compilation Details

ABI Bytecode

Warning: SPDX license identifier

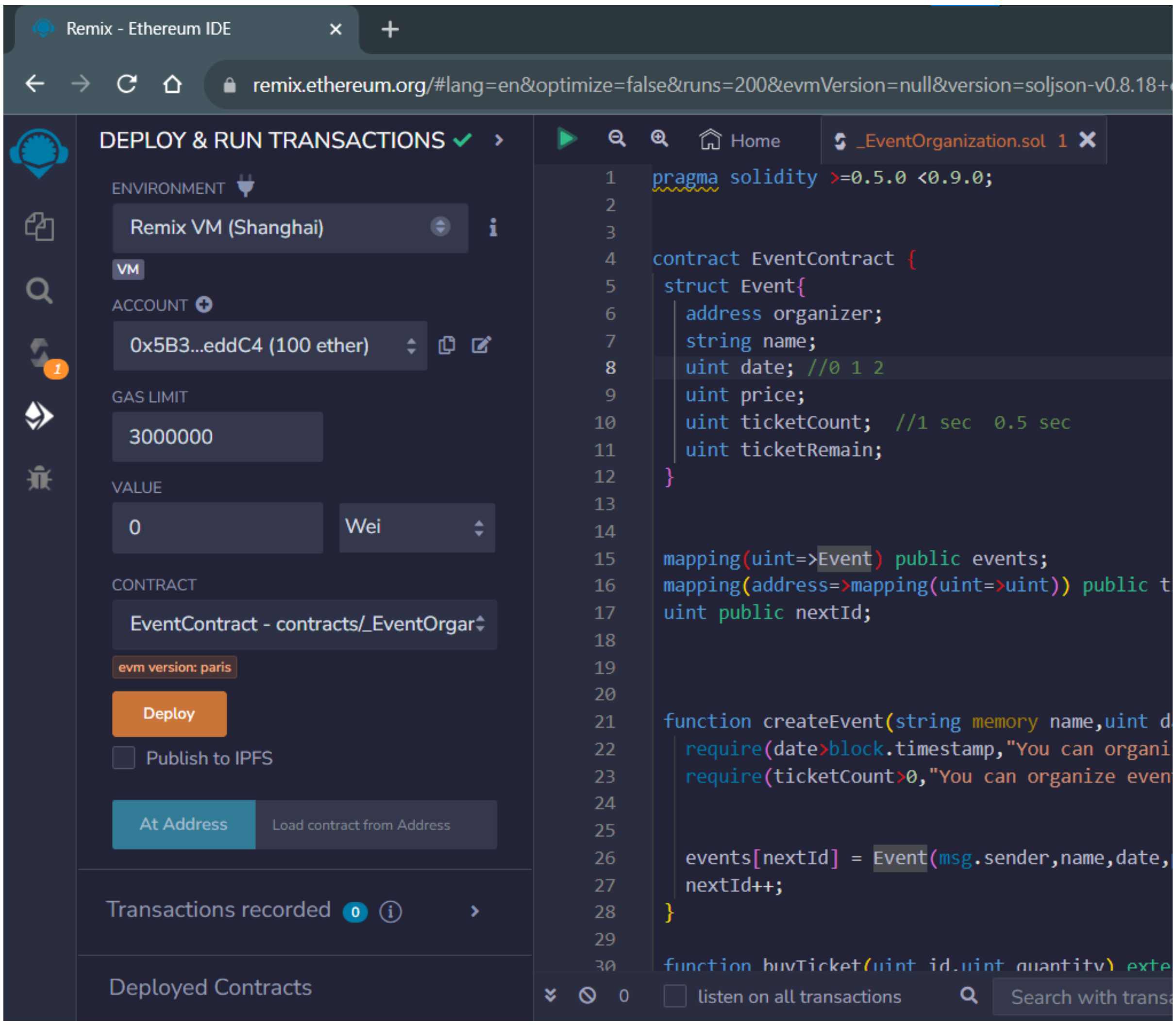
_EventOrganization.sol

```
pragma solidity >=0.5.0 <0.9.0;


contract EventContract {
 struct Event{
   address organizer;
   string name;
   uint date; //0 1 2
   uint price;
   uint ticketCount;  //1 sec  0.5 sec

 mapping(uint=>Event) public events;
 mapping(address=>mapping(uint=>uint)) public tickets;
 uint public nextId;



 function createEvent(string memory name,uint date,uint price,uint ticketCount) external{   infinite gas
   require(date>block.timestamp,"You can organize event for future date");
   require(ticketCount>0,"You can organize event only if you create more than 0 tickets");


   events[nextId] = Event(msg.sender,name,date,price,ticketCount,ticketCount);
   nextId++;
 }

 function buyTicket(uint id,uint quantity) external payable{   infinite gas
```

listen on all transactions

Search with transaction hash or address

3) Deploying the event management contract


Remix - Ethereum IDE

remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Remix VM (Shanghai)

VM

ACCOUNT: 0x5B3...eddC4 (100 ether)

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: EventContract - contracts/_EventOrgar

evm version: paris

Deploy

Publish to IPFS

At Address

Load contract from Address

Transactions recorded 0

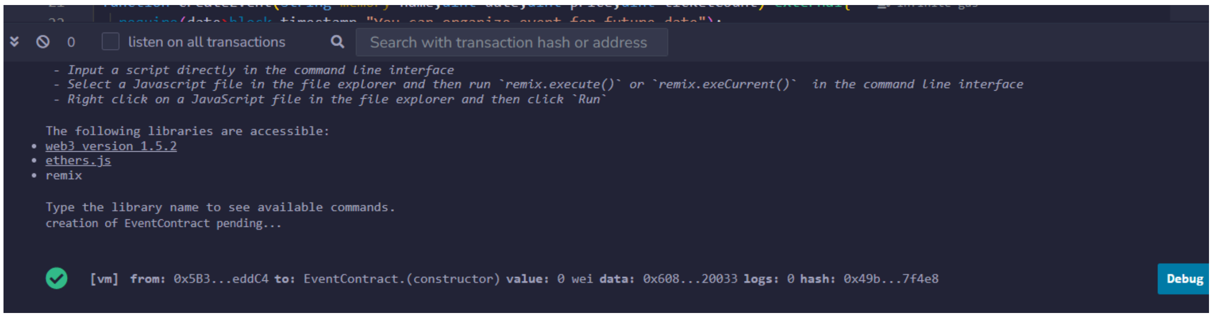
Deployed Contracts

_EventOrganization.sol

```
pragma solidity >=0.5.0 <0.9.0;


contract EventContract {
 struct Event{
   address organizer;
   string name;
   uint date; //0 1 2
   uint price;
   uint ticketCount;  //1 sec  0.5 sec
   uint ticketRemain;
 }


 mapping(uint=>Event) public events;
 mapping(address=>mapping(uint=>uint)) public t
 uint public nextId;



 function createEvent(string memory name,uint d
   require(date>block.timestamp,"You can organi
   require(ticketCount>0,"You can organize even


   events[nextId] = Event(msg.sender,name,date,
   nextId++;
 }

 function buyTicket(uint id,uint quantity) exte
```

listen on all transactions

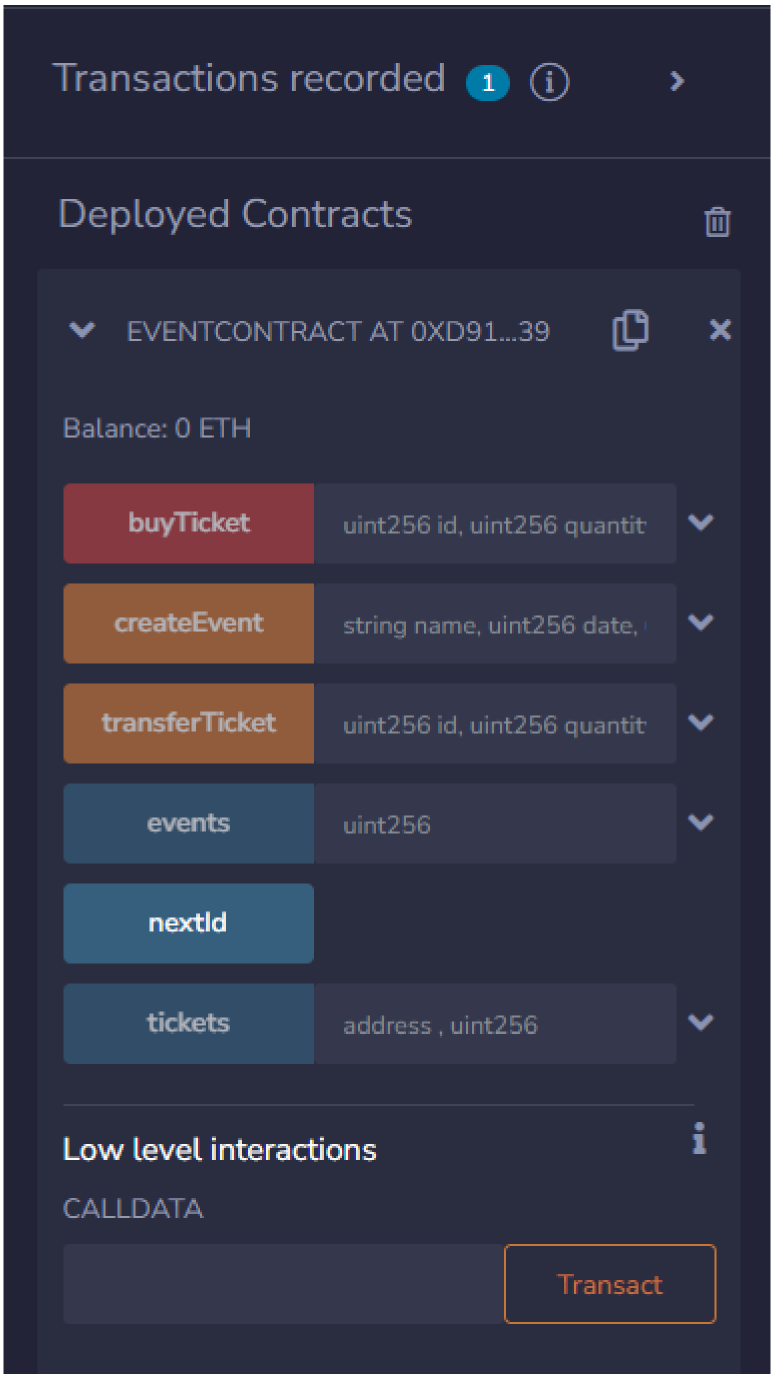
Search with trans



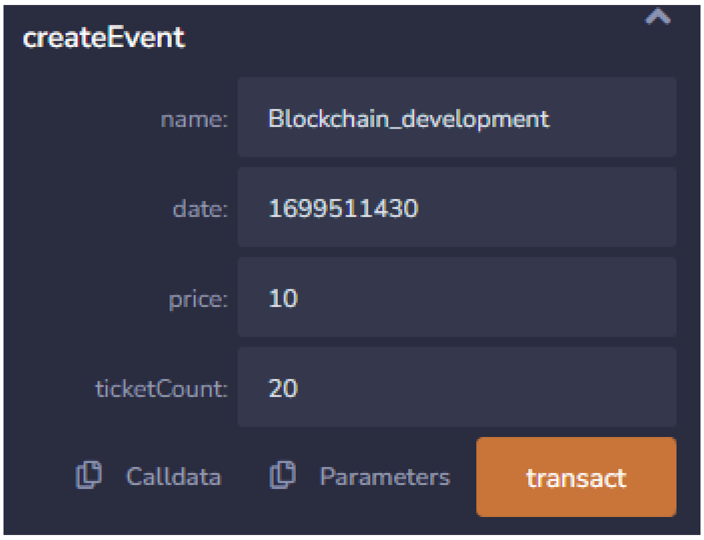
listen on all transactions
Search with transaction hash or address

```
 - Input a script directly in the command line interface
 - Select a Javascript file in the file explorer and then run `remix.execute()` or `remix.exeCurrent()`  in the command line interface
 - Right click on a JavaScript file in the file explorer and then click `Run`

The following libraries are accessible:
• web3 version 1.5.2
• ethers.js
• remix

Type the library name to see available commands.
creation of EventContract pending...

[vm] from: 0x5B3...eddC4 to: EventContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x49b...7f4e8
```

4) Now we have options like buying tickets , creating events , transferring tickets etc.

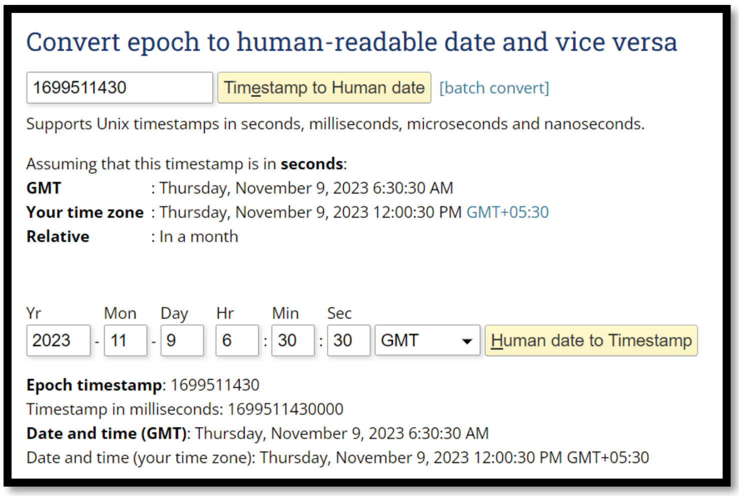


5) Functions for creating events.



createEvent

name:	Blockchain_development
date:	1699511430
price:	10
ticketCount:	20



Convert epoch to human-readable date and vice versa

1699511430

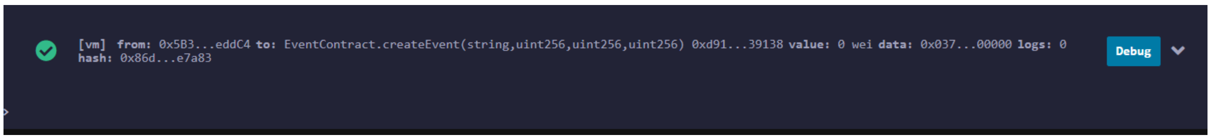
Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **seconds**:

GMT	: Thursday, November 9, 2023 6:30:30 AM
Your time zone	: Thursday, November 9, 2023 12:00:30 PM GMT+05:30
Relative	: In a month

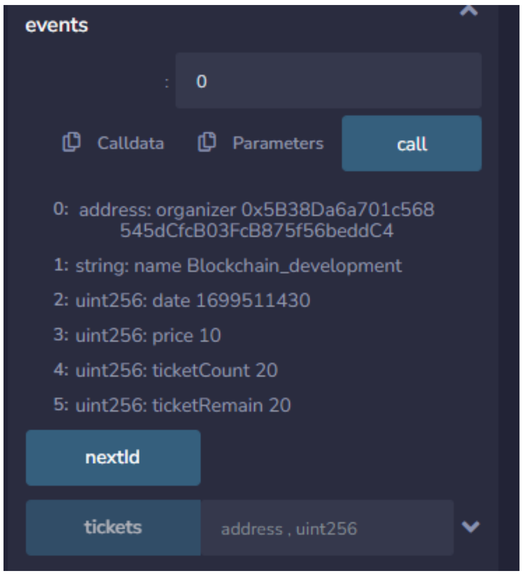
Yr	Mon	Day	Hr	Min	Sec	
2023	11	9	6	30	30	GMT

Epoch timestamp: 1699511430
Timestamp in milliseconds: 1699511430000
Date and time (GMT): Thursday, November 9, 2023 6:30:30 AM
Date and time (your time zone): Thursday, November 9, 2023 12:00:30 PM GMT+05:30



```
[vm] from: 0x5B3...eddC4 to: EventContract.createEvent(string,uint256,uint256,uint256) 0xd91...39138 value: 0 wei data: 0x037...00000 logs: 0
hash: 0x86d...e7a83
```

6)Functions for buying tickets for the created event “Blockchain Development” which is indexed as 0.



events

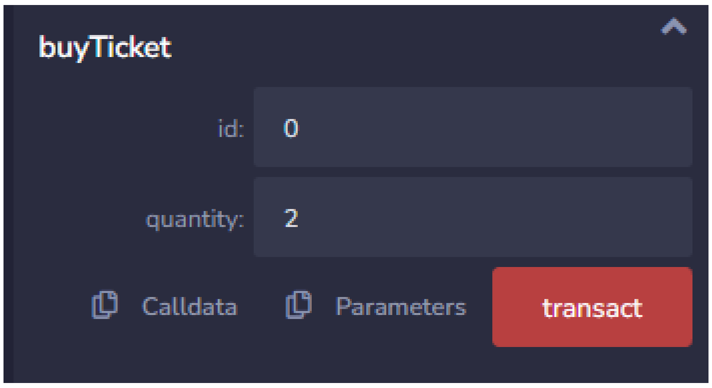
: 0

Calldata Parameters call

```
0: address: organizer 0x5B38Da6a701c568
   545dCfcB03FcB875f56beddC4
1: string: name Blockchain_development
2: uint256: date 1699511430
3: uint256: price 10
4: uint256: ticketCount 20
5: uint256: ticketRemain 20
```

nextId

tickets address , uint256

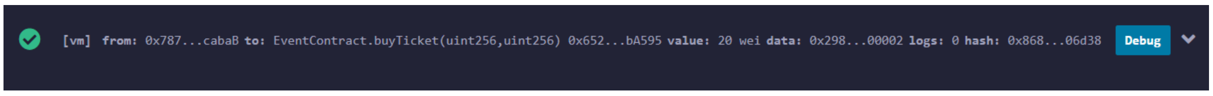


buyTicket

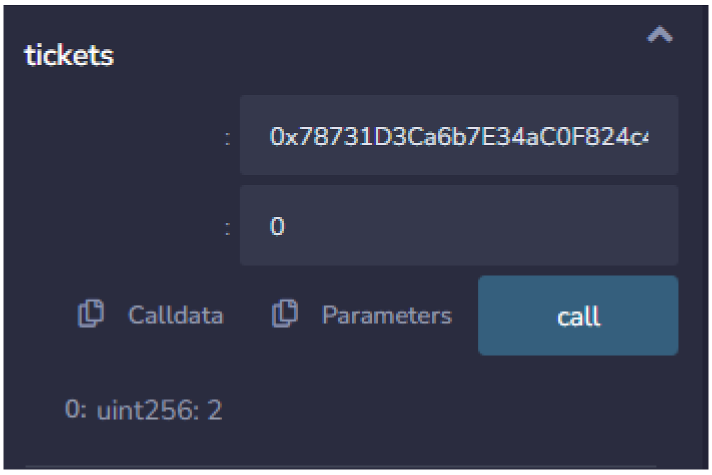
id: 0

quantity: 2

Calldata Parameters transact



```
[vm] from: 0x787...cabaB to: EventContract.buyTicket(uint256,uint256) 0x652...bA595 value: 20 wei data: 0x298...00002 logs: 0 hash: 0x868...06d38
```



tickets

: 0x78731D3Ca6b7E34aC0F824c

: 0

Calldata Parameters call

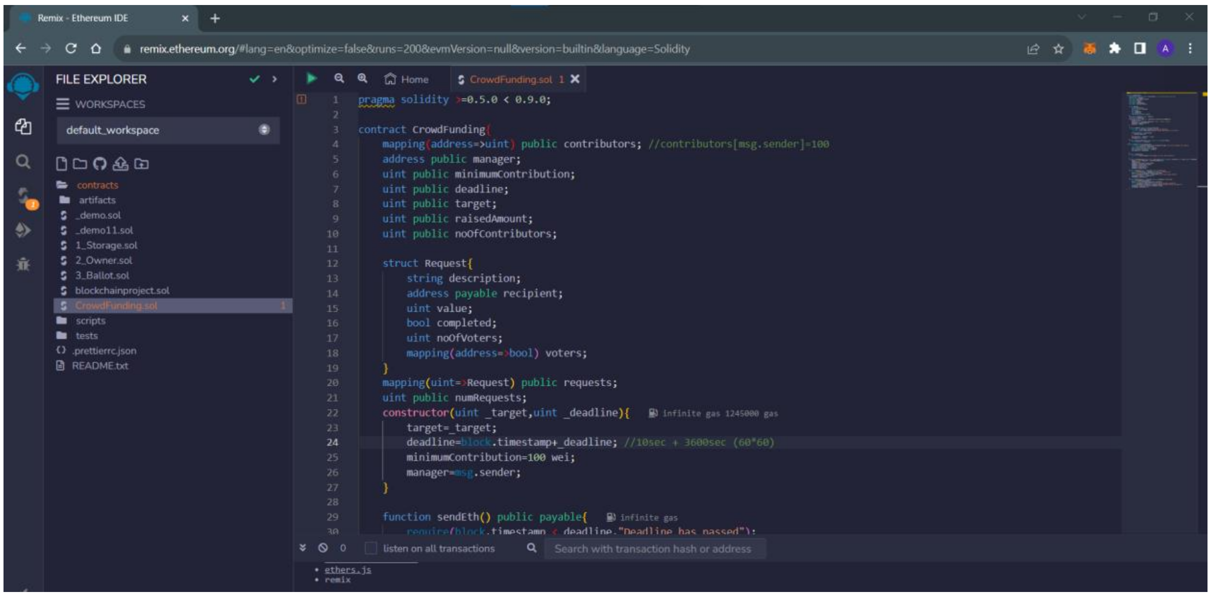
```
0: uint256: 2
```

- **<u>Crowd Funding Contract:</u>**
  - **CODE:**

```
CROWDFUNDING
pragma solidity >=0.5.0 < 0.9.0;

contract CrowdFunding{
    mapping(address=>uint) public contributors;
//contributors[msg.sender]=100
    address public manager;
    uint public minimumContribution;
    uint public deadline;
    uint public target;
    uint public raisedAmount;
    uint public noOfContributors;

    struct Request{
        string description;
        address payable recipient;
        uint value;
        bool completed;
        uint noOfVoters;
        mapping(address=>bool) voters;
    }
    mapping(uint=>Request) public requests;
    uint public numRequests;
    constructor(uint _target,uint _deadline){
        target=_target;
        deadline=block.timestamp+_deadline; //10sec + 3600sec
(60*60)
        minimumContribution=100 wei;
        manager=msg.sender;
    }

    function sendEth() public payable{
        require(block.timestamp < deadline,"Deadline has passed");
        require(msg.value >=minimumContribution,"Minimum
Contribution is not met");

        if(contributors[msg.sender]==0){
            noOfContributors++;
        }
        contributors[msg.sender]+=msg.value;
        raisedAmount+=msg.value;
    }
    function getContractBalance() public view returns(uint){
        return address(this).balance;
    }
 function refund() public{
        require(block.timestamp>deadline && raisedAmount<target,"You 
are not eligible for refund");
        require(contributors[msg.sender]>0);
        address payable user=payable(msg.sender);
        user.transfer(contributors[msg.sender]);
        contributors[msg.sender]=0;

    }
    modifier onlyManger(){
        require(msg.sender==manager,"Only manager can calll this 
function");
        _;
    }
    function createRequests(string memory _description,address 
payable _recipient,uint _value) public onlyManger{
        Request storage newRequest = requests[numRequests];
        numRequests++;
        newRequest.description=_description;
        newRequest.recipient=_recipient;
        newRequest.value=_value;
        newRequest.completed=false;
        newRequest.noOfVoters=0;
    }
    function voteRequest(uint _requestNo) public{
        require(contributors[msg.sender]>0,"YOu must be 
contributor");
        Request storage thisRequest=requests[_requestNo];
        require(thisRequest.voters[msg.sender]==false,"You have 
already voted");
        thisRequest.voters[msg.sender]=true;
        thisRequest.noOfVoters++;
    }
    function makePayment(uint _requestNo) public onlyManger{
        require(raisedAmount>=target);
        Request storage thisRequest=requests[_requestNo];
        require(thisRequest.completed==false,"The request has been 
completed");
        require(thisRequest.noOfVoters > 
noOfContributors/2,"Majority does not support");
        thisRequest.recipient.transfer(thisRequest.value);
        thisRequest.completed=true;
    }
}
```

➢ SCREENSHOTS:

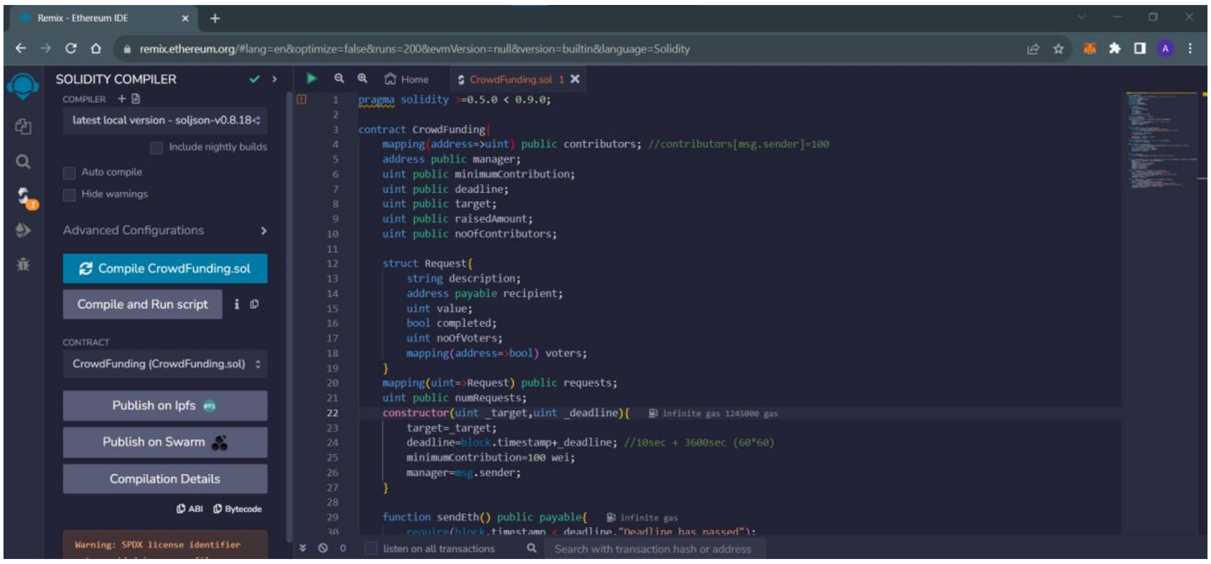
1) **The crowd funding project code in REMIX IDE**



```
pragma solidity >=0.5.0 < 0.9.0;

contract CrowdFunding{
    mapping(address=>uint) public contributors; //contributors[msg.sender]=100
    address public manager;
    uint public minimumContribution;
    uint public deadline;
    uint public target;
    uint public raisedAmount;
    uint public noOfContributors;

    struct Request{
        string description;
        address payable recipient;
        uint value;
        bool completed;
        uint noOfVoters;
        mapping(address=>bool) voters;
    }
    mapping(uint=>Request) public requests;
    uint public numRequests;
    constructor(uint _target,uint _deadline){    infinite gas 1245000 gas
        target=_target;
        deadline=block.timestamp+_deadline; //10sec + 3600sec (60*60)
        minimumContribution=100 wei;
        manager=msg.sender;
    }

    function sendEth() public payable{    infinite gas
        require(block.timestamp < deadline,"Deadline has passed");
```

```
• ethers.js
• remix
```

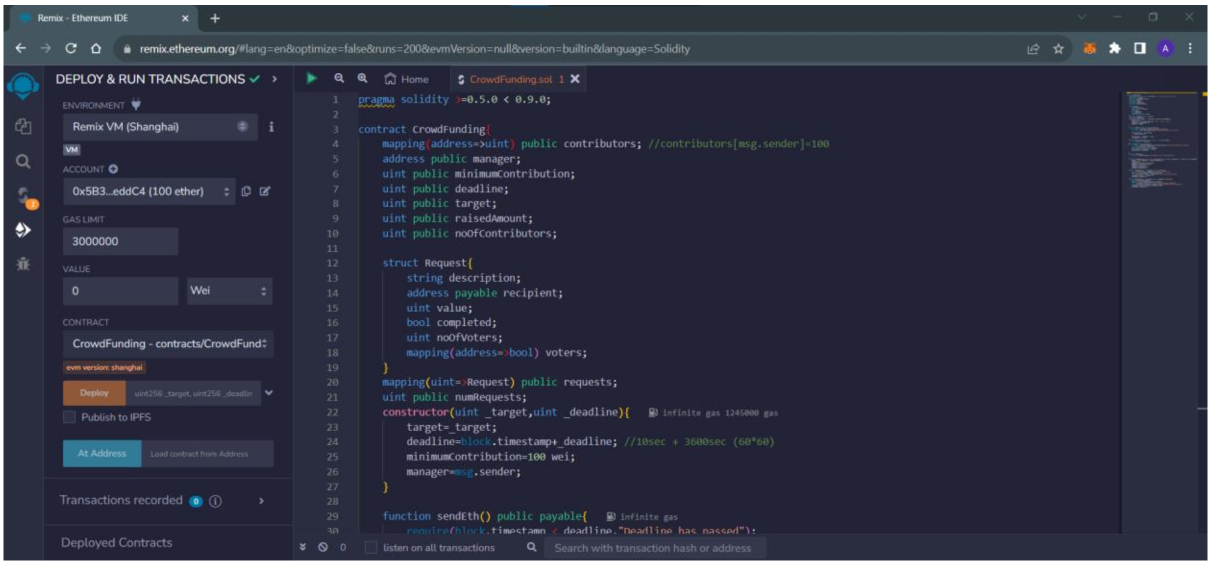
2) Compiling the contract with the latest version compiler.



```
Warning: SPDX license identifier
```

```
pragma solidity >=0.5.0 < 0.9.0;

contract CrowdFunding{
    mapping(address=>uint) public contributors; //contributors[msg.sender]=100
    address public manager;
    uint public minimumContribution;
    uint public deadline;
    uint public target;
    uint public raisedAmount;
    uint public noOfContributors;

    struct Request{
        string description;
        address payable recipient;
        uint value;
        bool completed;
        uint noOfVoters;
        mapping(address=>bool) voters;
    }
    mapping(uint=>Request) public requests;
    uint public numRequests;
    constructor(uint _target,uint _deadline){    infinite gas 1245000 gas
        target=_target;
        deadline=block.timestamp+_deadline; //10sec + 3600sec (60*60)
        minimumContribution=100 wei;
        manager=msg.sender;
    }

    function sendEth() public payable{    infinite gas
        require(block.timestamp < deadline,"Deadline has passed");
```

3) Deploying the project



remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=builtin&language=Solidity

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Remix VM (Shanghai)

ACCOUNT: 0x5B3...eddC4 (100 ether)

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: CrowdFunding - contracts/CrowdFund

evm version: shanghai

Deploy: uint256 _target, uint256 _deadlin

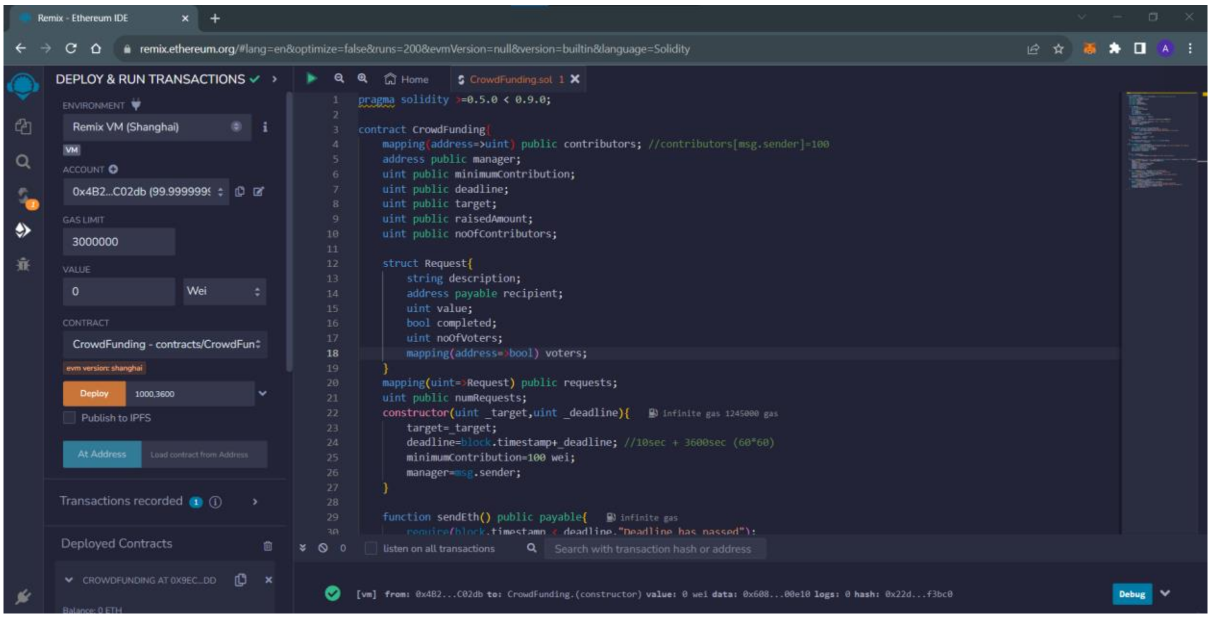
Transactions recorded 0

Deployed Contracts

CrowdFunding.sol

```
pragma solidity >=0.5.0 < 0.9.0;

contract CrowdFunding{
    mapping(address=>uint) public contributors; //contributors[msg.sender]=100
    address public manager;
    uint public minimumContribution;
    uint public deadline;
    uint public target;
    uint public raisedAmount;
    uint public noOfContributors;

    struct Request{
        string description;
        address payable recipient;
        uint value;
        bool completed;
        uint noOfVoters;
        mapping(address=>bool) voters;
    }
    mapping(uint=>Request) public requests;
    uint public numRequests;
    constructor(uint _target,uint _deadline){     infinite gas 1245000 gas
        target=_target;
        deadline=block.timestamp+_deadline; //10sec + 3600sec (60*60)
        minimumContribution=100 wei;
        manager=msg.sender;
    }

    function sendEth() public payable{     infinite gas
        require(block.timestamp < deadline,"Deadline has passed");
```

4) Target is given as 1000 units and 3600 seconds is the duration



remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=builtin&language=Solidity

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Remix VM (Shanghai)

ACCOUNT: 0x4B2...C02db (99.999999

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: CrowdFunding - contracts/CrowdFun

evm version: shanghai

Deploy: 1000,3600

Transactions recorded 1

Deployed Contracts

CROWDFUNDING AT 0X9EC...DD

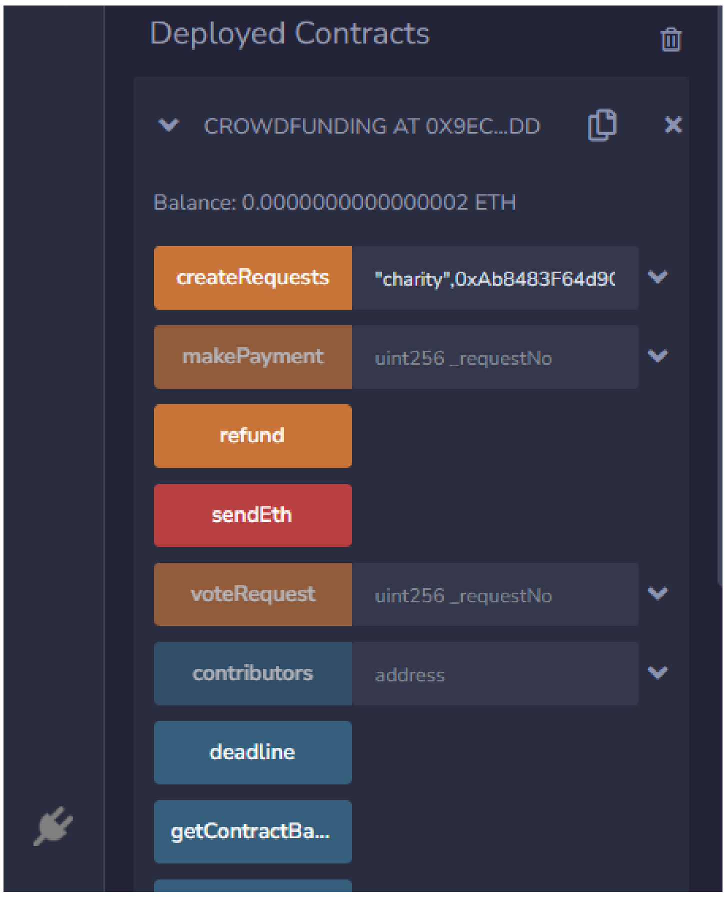
Balance: 0 ETH

CrowdFunding.sol

```
pragma solidity >=0.5.0 < 0.9.0;

contract CrowdFunding{
    mapping(address=>uint) public contributors; //contributors[msg.sender]=100
    address public manager;
    uint public minimumContribution;
    uint public deadline;
    uint public target;
    uint public raisedAmount;
    uint public noOfContributors;

    struct Request{
        string description;
        address payable recipient;
        uint value;
        bool completed;
        uint noOfVoters;
        mapping(address=>bool) voters;
    }
    mapping(uint=>Request) public requests;
    uint public numRequests;
    constructor(uint _target,uint _deadline){     infinite gas 1245000 gas
        target=_target;
        deadline=block.timestamp+_deadline; //10sec + 3600sec (60*60)
        minimumContribution=100 wei;
        manager=msg.sender;
    }

    function sendEth() public payable{     infinite gas
        require(block.timestamp < deadline,"Deadline has passed");
```

```
[vm] from: 0x4B2...C02db to: CrowdFunding.(constructor) value: 0 wei data: 0x608...00e10 logs: 0 hash: 0x22d...f3bc0
```

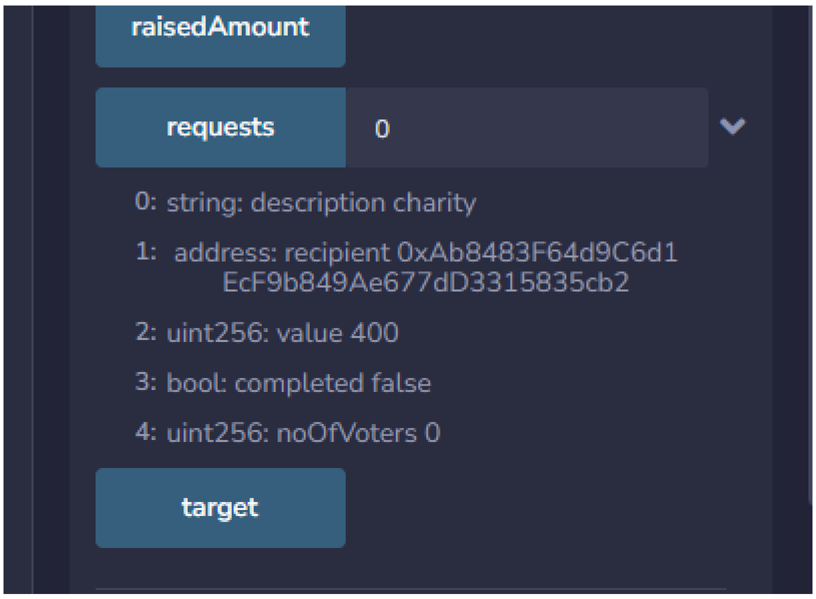
5) We can create requests and make payments. Here charity request is created with the address to send the money.



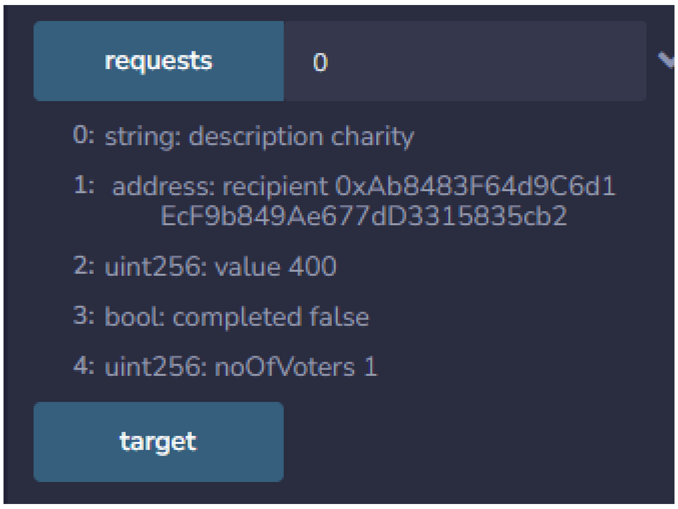
Balance: 0.0000000000000002 ETH

```
createRequests  "charity",0xAb8483F64d9(
makePayment     uint256 _requestNo
voteRequest     uint256 _requestNo
contributors    address
```

6) The charity has the index and we can know its details like target , no of voters etc.

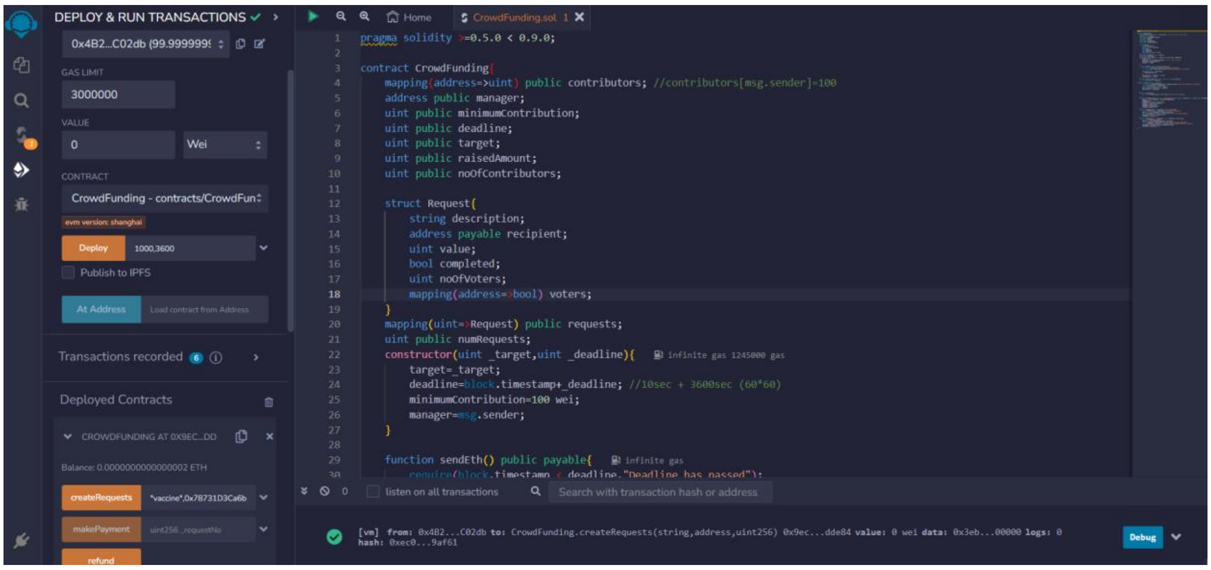


```
requests  0
0: string: description charity
1:  address: recipient 0xAb8483F64d9C6d1
       EcF9b849Ae677dD3315835cb2
2: uint256: value 400
3: bool: completed false
4: uint256: noOfVoters 0
```



```
requests    0
0: string: description charity
1:  address: recipient 0xAb8483F64d9C6d1
      EcF9b849Ae677dD3315835cb2
2: uint256: value 400
3: bool: completed false
4: uint256: noOfVoters 1
target
```

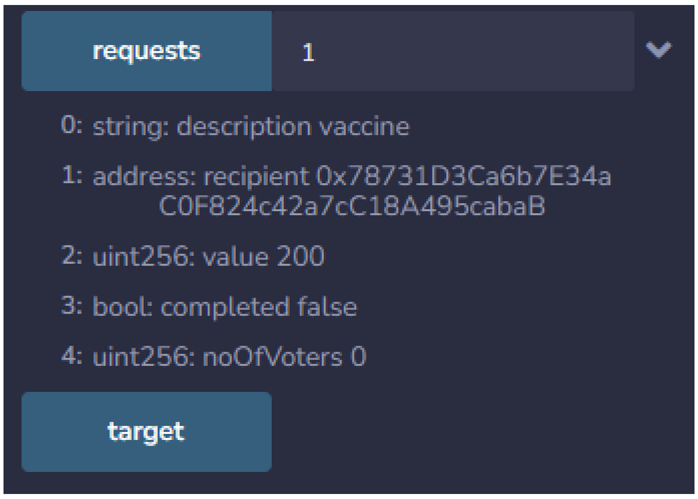
7) In the same way we created vaccine crowd funding with index 1.



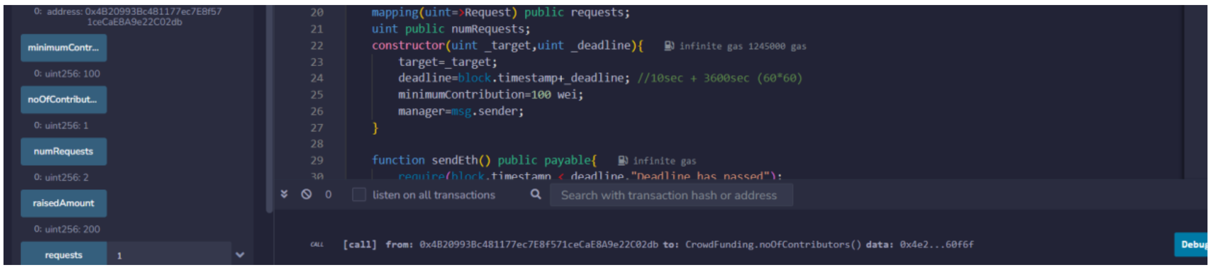
```
DEPLOY & RUN TRANSACTIONS
0x4B2...C02db (99.999999
GAS LIMIT
3000000
VALUE
0    Wei
CONTRACT
CrowdFunding - contracts/CrowdFun
evm version: shanghai
Deploy  1000,3600
Publish to IPFS
At Address  Load contract from Address
Transactions recorded 6
Deployed Contracts
CROWDFUNDING AT 0X9EC...DD
Balance: 0.0000000000000002 ETH
createRequests  "vaccine",0x78731D3Ca6b
makePayment  uint256 _requestNo
refund
```

```
pragma solidity >=0.5.0 < 0.9.0;

contract CrowdFunding{
    mapping(address=>uint) public contributors; //contributors[msg.sender]=100
    address public manager;
    uint public minimumContribution;
    uint public deadline;
    uint public target;
    uint public raisedAmount;
    uint public noOfContributors;

    struct Request{
        string description;
        address payable recipient;
        uint value;
        bool completed;
        uint noOfVoters;
        mapping(address=>bool) voters;
    }
    mapping(uint=>Request) public requests;
    uint public numRequests;
    constructor(uint _target,uint _deadline){    infinite gas 1245000 gas
        target=_target;
        deadline=block.timestamp+_deadline; //10sec + 3600sec (60*60)
        minimumContribution=100 wei;
        manager=msg.sender;
    }

    function sendEth() public payable{    infinite gas
        require(block.timestamp < deadline,"Deadline has passed");
```

```
[vm] from: 0x4B2...C02db to: CrowdFunding.createRequests(string,address,uint256) 0x9ec...dde84 value: 0 wei data: 0x3eb...00000 logs: 0
hash: 0xec0...9af61
```



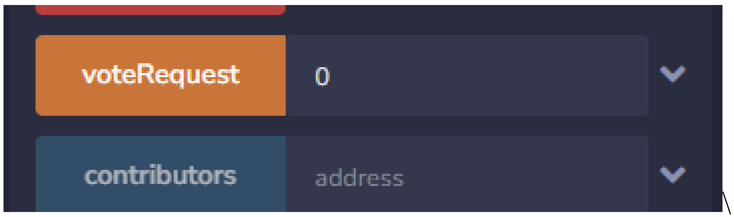
```
requests    1
0: string: description vaccine
1: address: recipient 0x78731D3Ca6b7E34a
      C0F824c42a7cC18A495cabaB
2: uint256: value 200
3: bool: completed false
4: uint256: noOfVoters 0
target
```



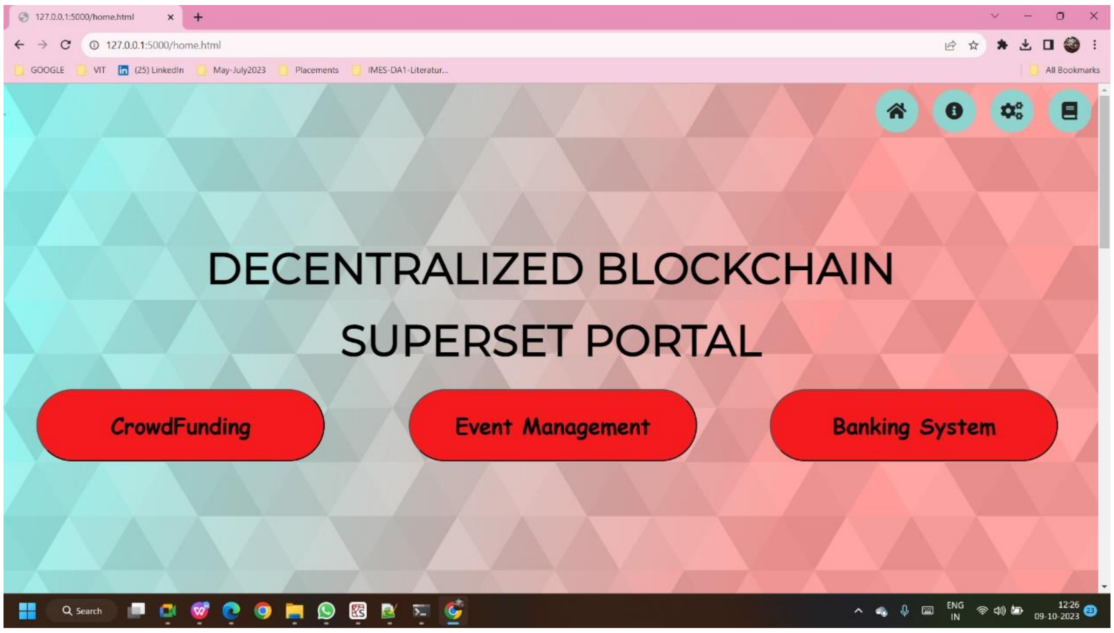
```
0: address: 0x4B20993Bc481177ec7E8f57
1ceCaE8A9e22C02db
minimumContr...
0: uint256: 100
noOfContribut...
0: uint256: 1
numRequests
0: uint256: 2
raisedAmount
0: uint256: 200
requests    1
```

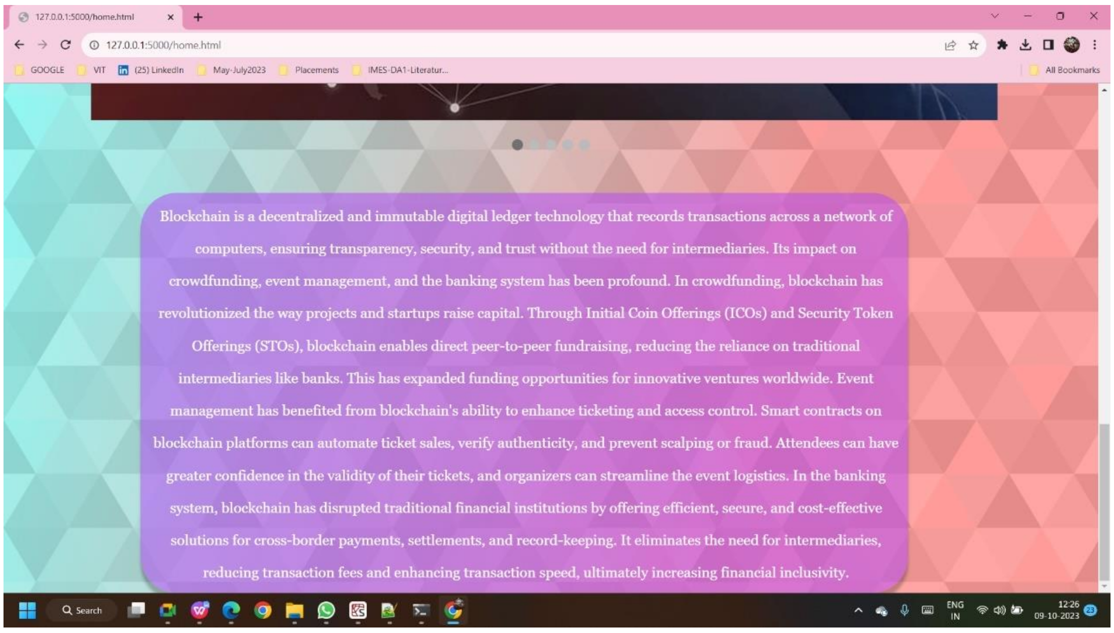
```
    mapping(uint=>Request) public requests;
    uint public numRequests;
    constructor(uint _target,uint _deadline){    infinite gas 1245000 gas
        target=_target;
        deadline=block.timestamp+_deadline; //10sec + 3600sec (60*60)
        minimumContribution=100 wei;
        manager=msg.sender;
    }

    function sendEth() public payable{    infinite gas
        require(block.timestamp < deadline,"Deadline has passed");
```

```
0    listen on all transactions    Search with transaction hash or address

CALL  [call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: CrowdFunding.noOfContributors() data: 0x4e2...60f6f
```



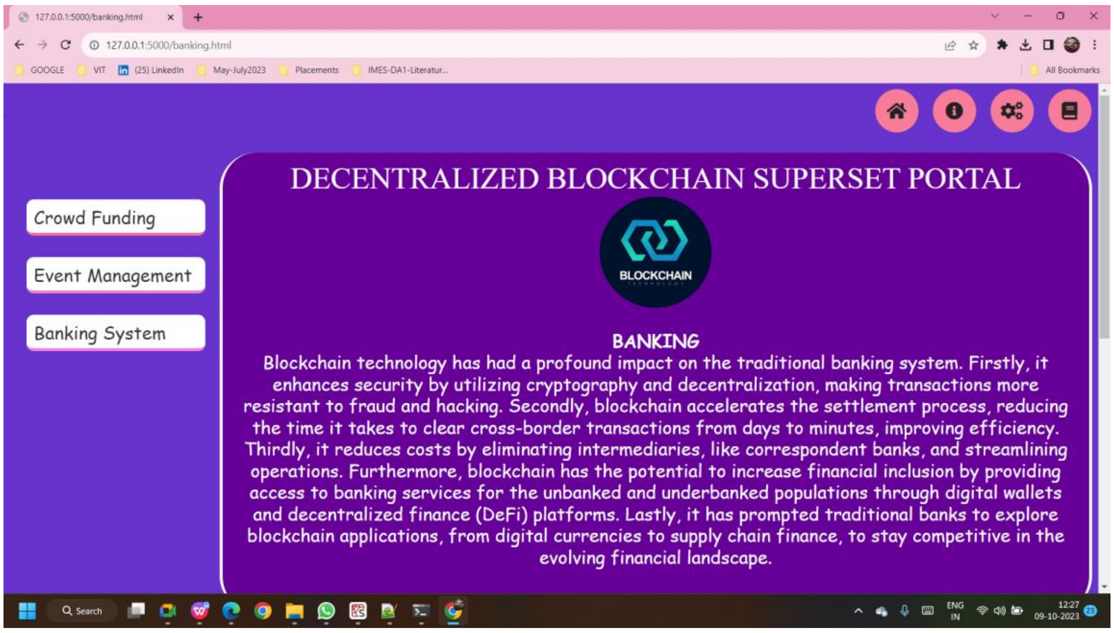
- **<u>Website Interface Screenshots:</u>**
  - ➢ **SCREENSHOTS:**
  1) Home Page





Blockchain is a decentralized and immutable digital ledger technology that records transactions across a network of computers, ensuring transparency, security, and trust without the need for intermediaries. Its impact on crowdfunding, event management, and the banking system has been profound. In crowdfunding, blockchain has revolutionized the way projects and startups raise capital. Through Initial Coin Offerings (ICOs) and Security Token Offerings (STOs), blockchain enables direct peer-to-peer fundraising, reducing the reliance on traditional intermediaries like banks. This has expanded funding opportunities for innovative ventures worldwide. Event management has benefited from blockchain's ability to enhance ticketing and access control. Smart contracts on blockchain platforms can automate ticket sales, verify authenticity, and prevent scalping or fraud. Attendees can have greater confidence in the validity of their tickets, and organizers can streamline the event logistics. In the banking system, blockchain has disrupted traditional financial institutions by offering efficient, secure, and cost-effective solutions for cross-border payments, settlements, and record-keeping. It eliminates the need for intermediaries, reducing transaction fees and enhancing transaction speed, ultimately increasing financial inclusivity.

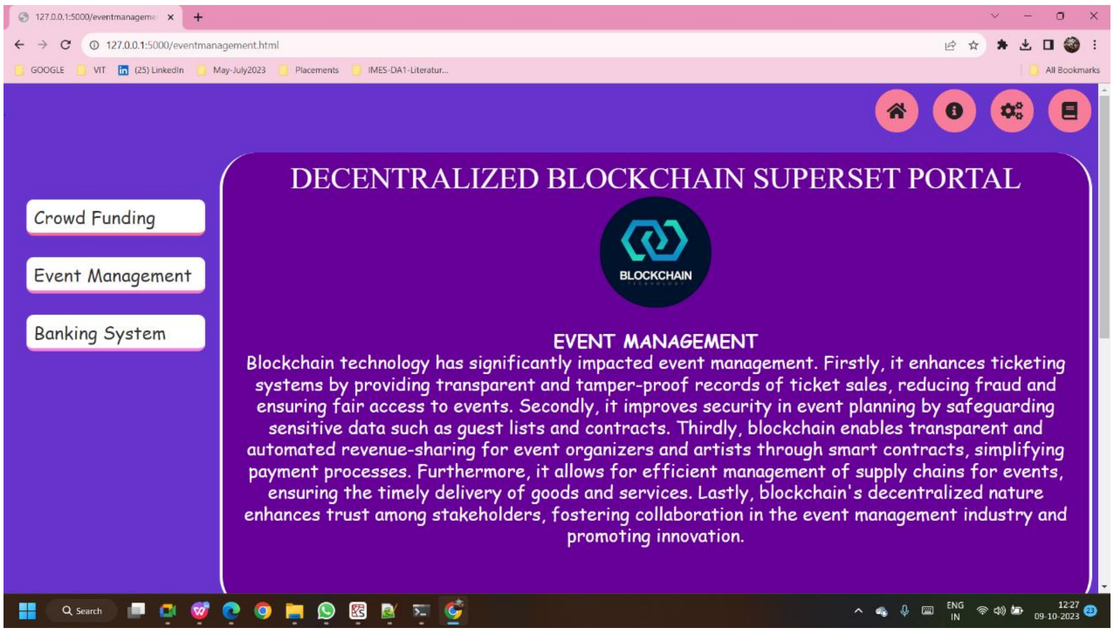
2) INFO Page about Banking Smart-Contract.


127.0.0.1:5000/banking.html
Crowd Funding
Event Management
Banking System
DECENTRALIZED BLOCKCHAIN SUPERSET PORTAL
BLOCKCHAIN

BANKING

Blockchain technology has had a profound impact on the traditional banking system. Firstly, it enhances security by utilizing cryptography and decentralization, making transactions more resistant to fraud and hacking. Secondly, blockchain accelerates the settlement process, reducing the time it takes to clear cross-border transactions from days to minutes, improving efficiency. Thirdly, it reduces costs by eliminating intermediaries, like correspondent banks, and streamlining operations. Furthermore, blockchain has the potential to increase financial inclusion by providing access to banking services for the unbanked and underbanked populations through digital wallets and decentralized finance (DeFi) platforms. Lastly, it has prompted traditional banks to explore blockchain applications, from digital currencies to supply chain finance, to stay competitive in the evolving financial landscape.

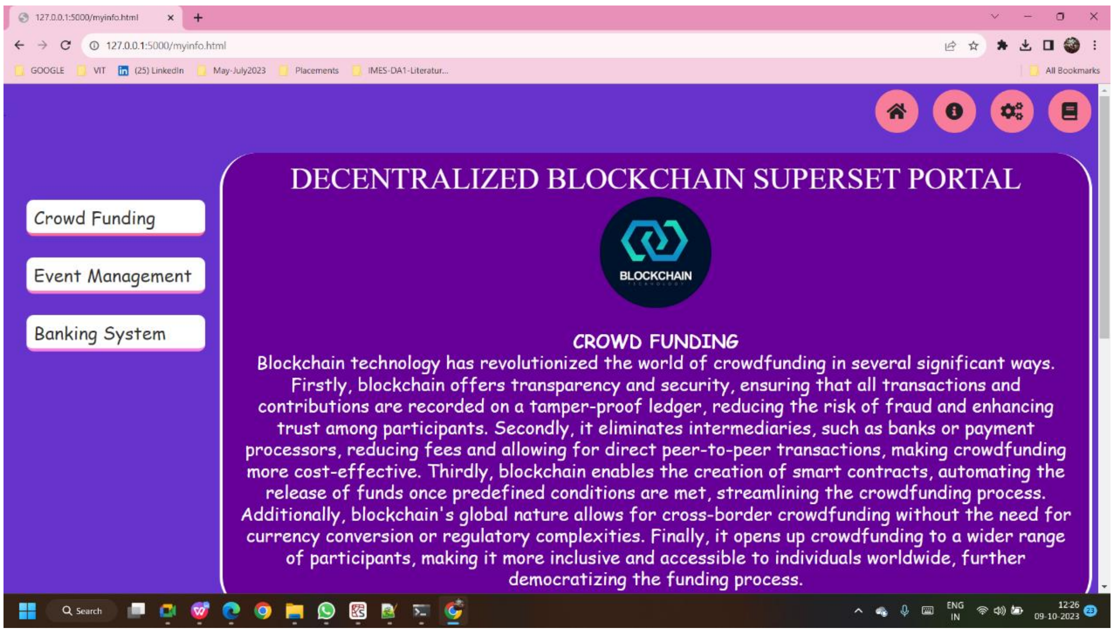
3) INFO Page about Event Management Smart-Contract.


127.0.0.1:5000/eventmanagement.html
Crowd Funding
Event Management
Banking System
DECENTRALIZED BLOCKCHAIN SUPERSET PORTAL
BLOCKCHAIN

EVENT MANAGEMENT

Blockchain technology has significantly impacted event management. Firstly, it enhances ticketing systems by providing transparent and tamper-proof records of ticket sales, reducing fraud and ensuring fair access to events. Secondly, it improves security in event planning by safeguarding sensitive data such as guest lists and contracts. Thirdly, blockchain enables transparent and automated revenue-sharing for event organizers and artists through smart contracts, simplifying payment processes. Furthermore, it allows for efficient management of supply chains for events, ensuring the timely delivery of goods and services. Lastly, blockchain's decentralized nature enhances trust among stakeholders, fostering collaboration in the event management industry and promoting innovation.

4) INFO Page about Crowd Funding Smart-Contract.

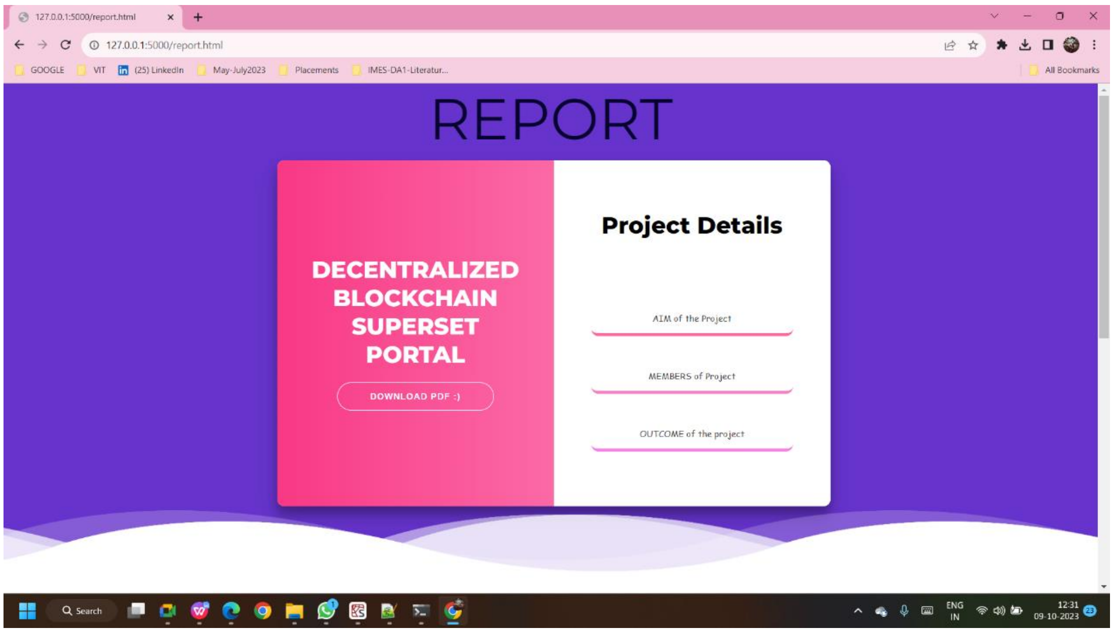


DECENTRALIZED BLOCKCHAIN SUPERSET PORTAL

CROWD FUNDING

Blockchain technology has revolutionized the world of crowdfunding in several significant ways. Firstly, blockchain offers transparency and security, ensuring that all transactions and contributions are recorded on a tamper-proof ledger, reducing the risk of fraud and enhancing trust among participants. Secondly, it eliminates intermediaries, such as banks or payment processors, reducing fees and allowing for direct peer-to-peer transactions, making crowdfunding more cost-effective. Thirdly, blockchain enables the creation of smart contracts, automating the release of funds once predefined conditions are met, streamlining the crowdfunding process. Additionally, blockchain's global nature allows for cross-border crowdfunding without the need for currency conversion or regulatory complexities. Finally, it opens up crowdfunding to a wider range of participants, making it more inclusive and accessible to individuals worldwide, further democratizing the funding process.

5) Details Page



REPORT

DECENTRALIZED BLOCKCHAIN SUPERSET PORTAL

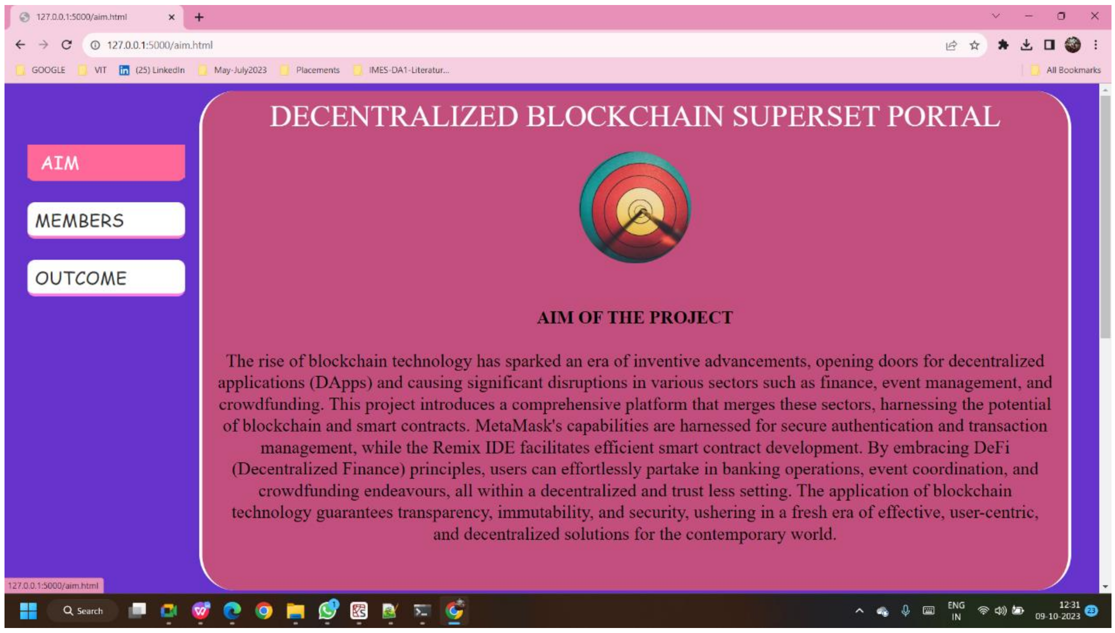
DOWNLOAD PDF :)

Project Details

AIM of the Project

MEMBERS of Project

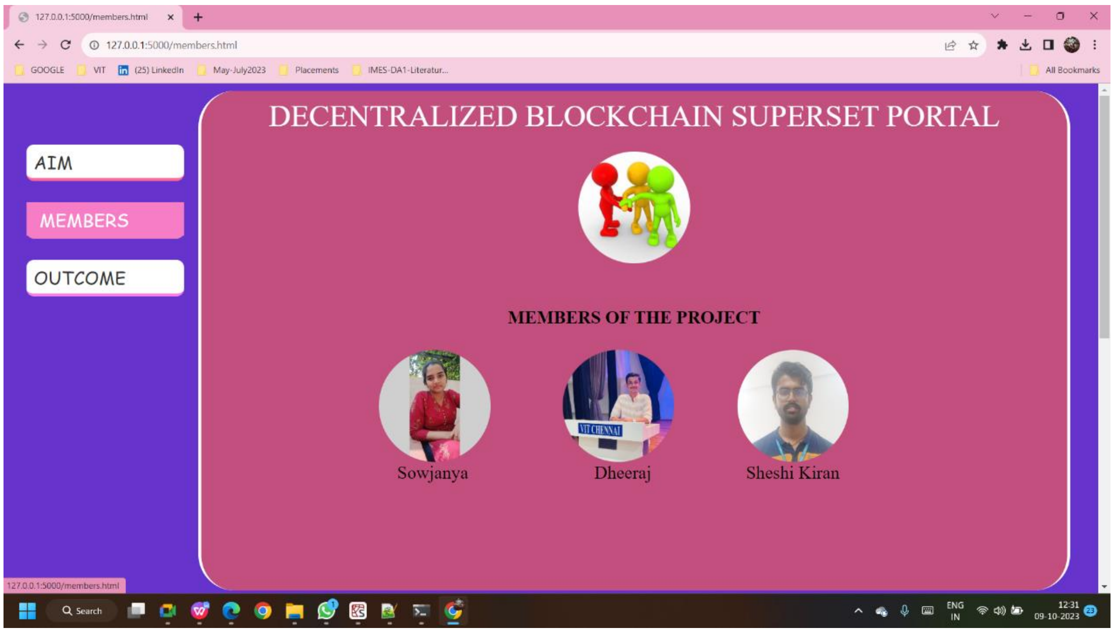
OUTCOME of the project



DECENTRALIZED BLOCKCHAIN SUPERSET PORTAL

AIM OF THE PROJECT

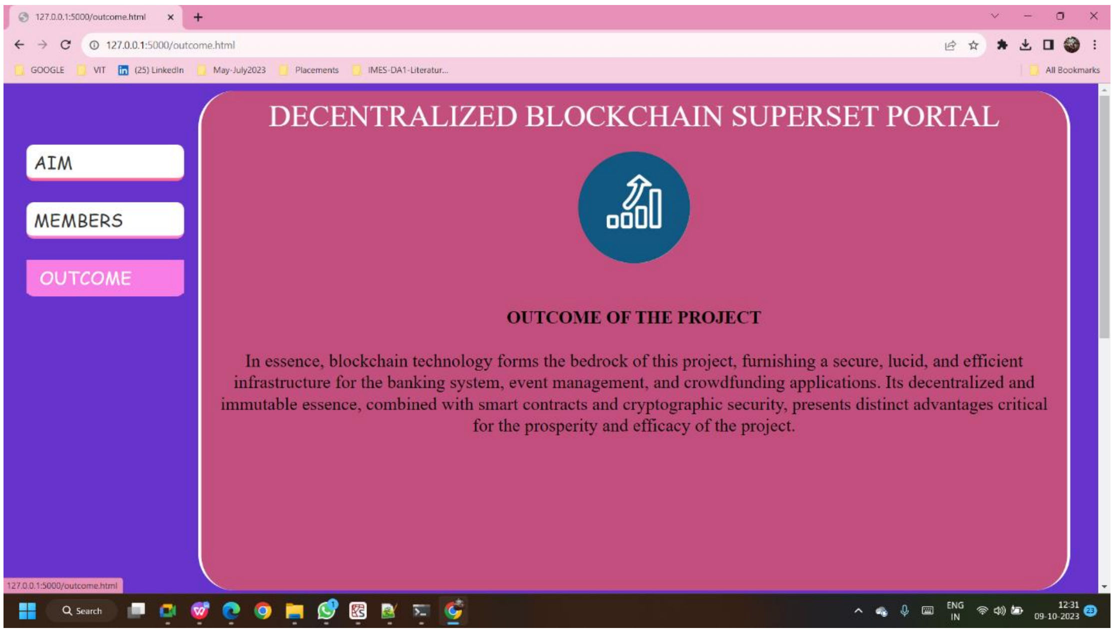
The rise of blockchain technology has sparked an era of inventive advancements, opening doors for decentralized applications (DApps) and causing significant disruptions in various sectors such as finance, event management, and crowdfunding. This project introduces a comprehensive platform that merges these sectors, harnessing the potential of blockchain and smart contracts. MetaMask's capabilities are harnessed for secure authentication and transaction management, while the Remix IDE facilitates efficient smart contract development. By embracing DeFi (Decentralized Finance) principles, users can effortlessly partake in banking operations, event coordination, and crowdfunding endeavours, all within a decentralized and trust less setting. The application of blockchain technology guarantees transparency, immutability, and security, ushering in a fresh era of effective, user-centric, and decentralized solutions for the contemporary world.



DECENTRALIZED BLOCKCHAIN SUPERSET PORTAL

MEMBERS OF THE PROJECT

Sowjanya Dheeraj Sheshi Kiran

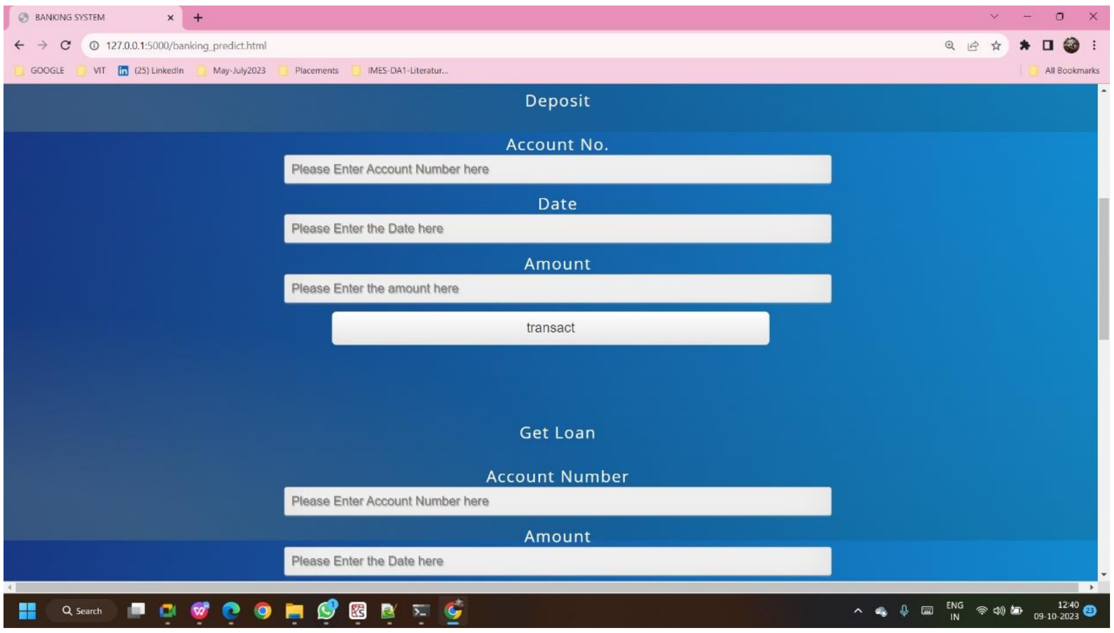


DECENTRALIZED BLOCKCHAIN SUPERSET PORTAL

OUTCOME OF THE PROJECT

In essence, blockchain technology forms the bedrock of this project, furnishing a secure, lucid, and efficient infrastructure for the banking system, event management, and crowdfunding applications. Its decentralized and immutable essence, combined with smart contracts and cryptographic security, presents distinct advantages critical for the prosperity and efficacy of the project.

6) Banking Smart Contract Page with required Functionalities.



Deposit

Account No.

Please Enter Account Number here

Date

Please Enter the Date here

Amount

Please Enter the amount here

transact

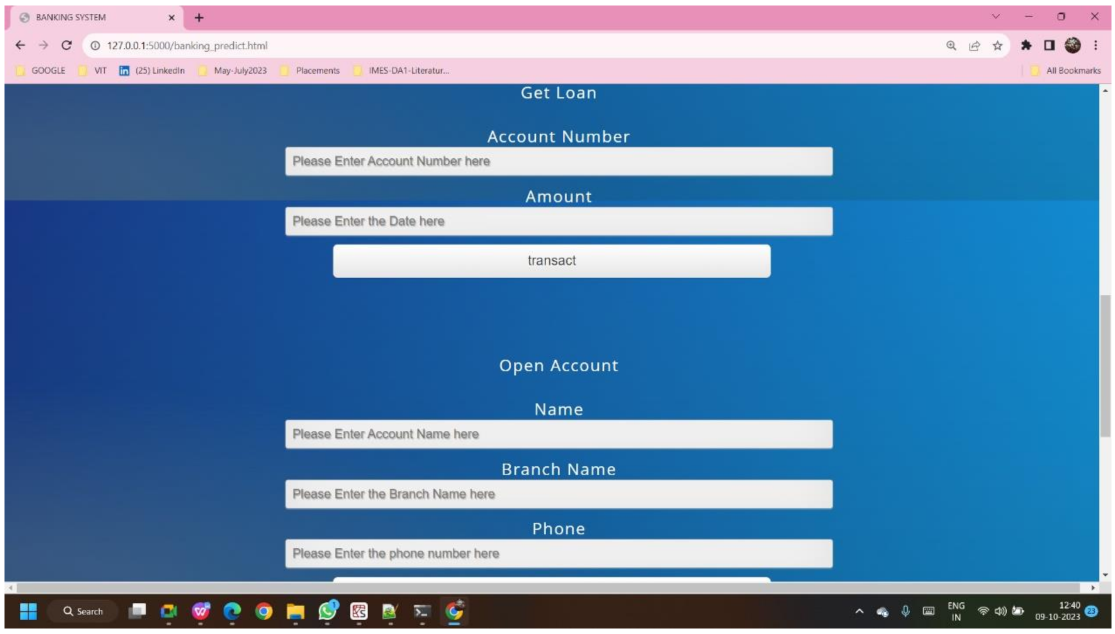
Get Loan

Account Number

Please Enter Account Number here

Amount

Please Enter the Date here



Get Loan

Account Number

Please Enter Account Number here

Amount

Please Enter the Date here

transact

Open Account

Name

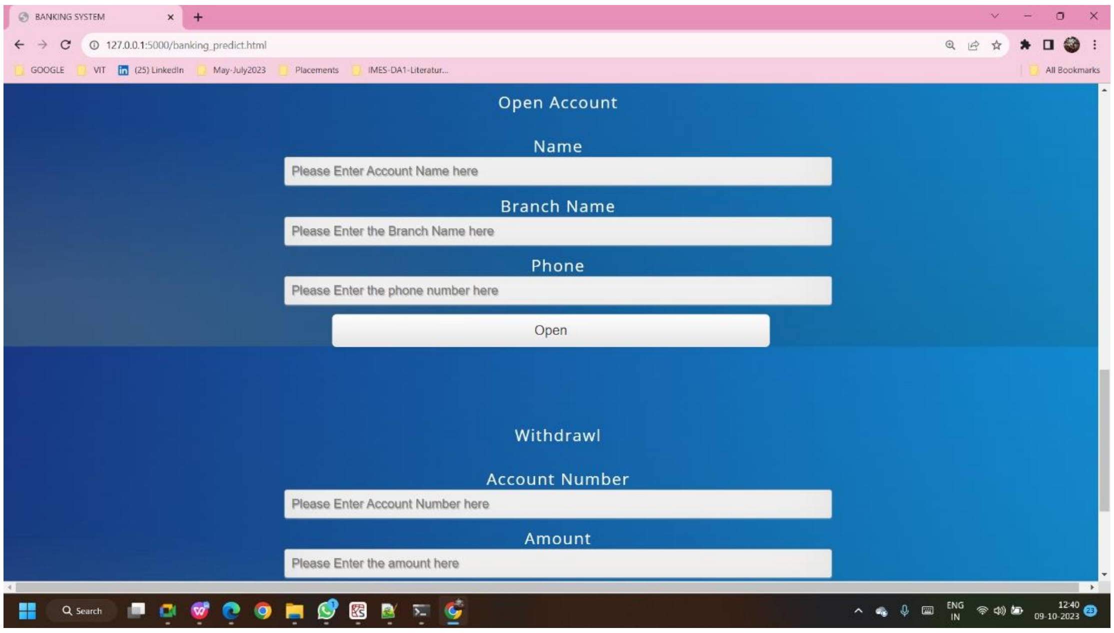
Please Enter Account Name here

Branch Name

Please Enter the Branch Name here

Phone

Please Enter the phone number here



Open Account

Name

Please Enter Account Name here

Branch Name

Please Enter the Branch Name here

Phone

Please Enter the phone number here

Open

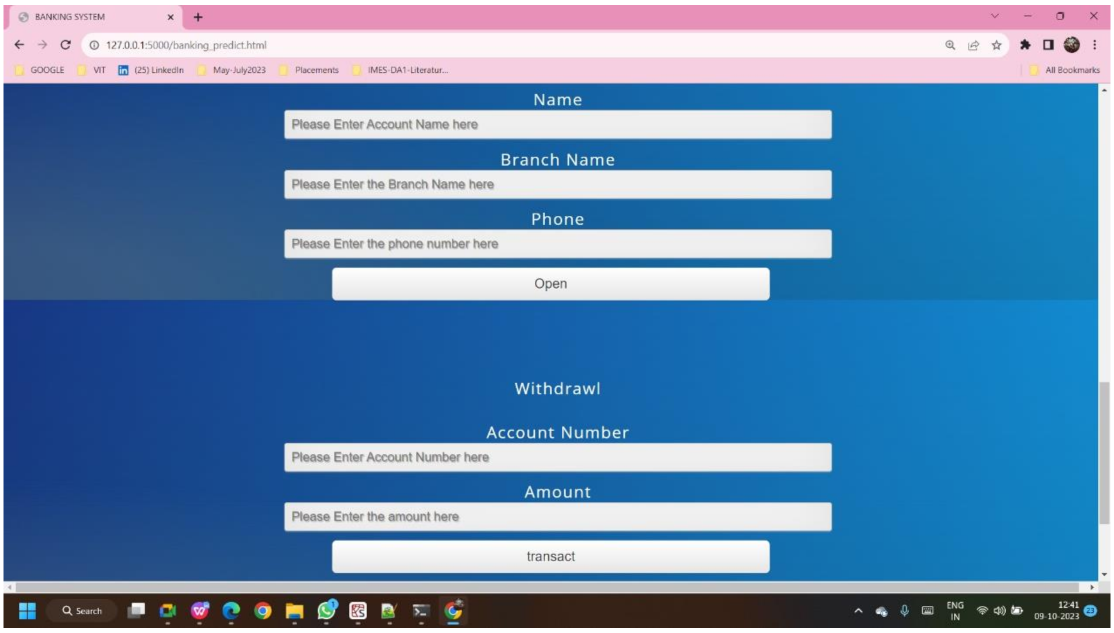
Withdrawl

Account Number

Please Enter Account Number here

Amount

Please Enter the amount here



Name

Please Enter Account Name here

Branch Name

Please Enter the Branch Name here

Phone

Please Enter the phone number here

Open

Withdrawl

Account Number

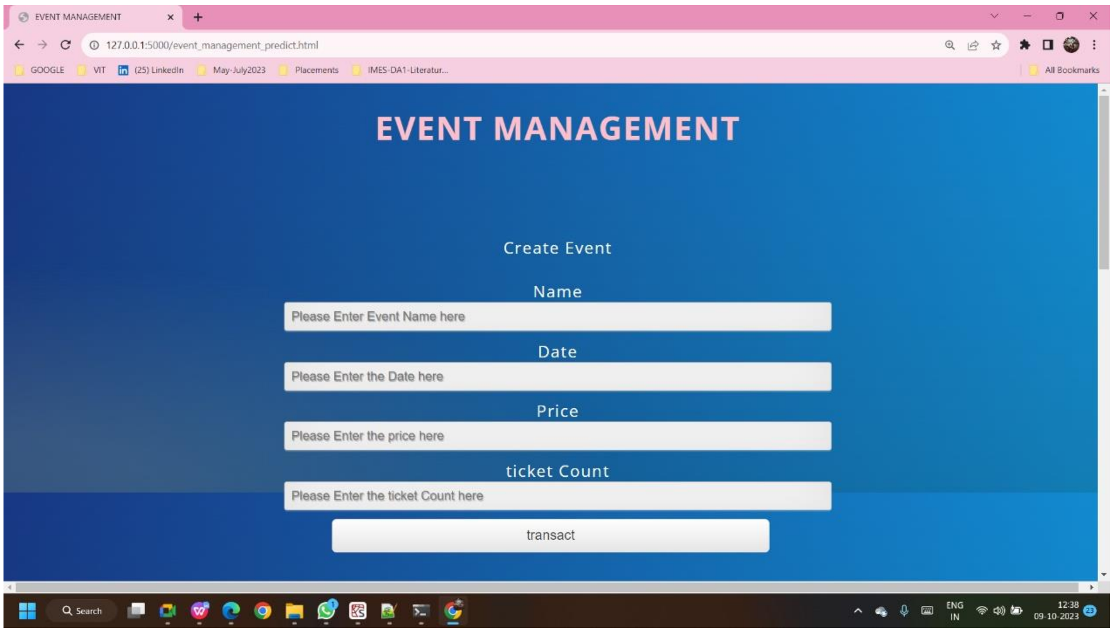
Please Enter Account Number here

Amount

Please Enter the amount here

transact

7) Event Management Smart-Contract with required Parameters.



EVENT MANAGEMENT

127.0.0.1:5000/event_management_predict.html

EVENT MANAGEMENT

Create Event

Name

Please Enter Event Name here

Date

Please Enter the Date here

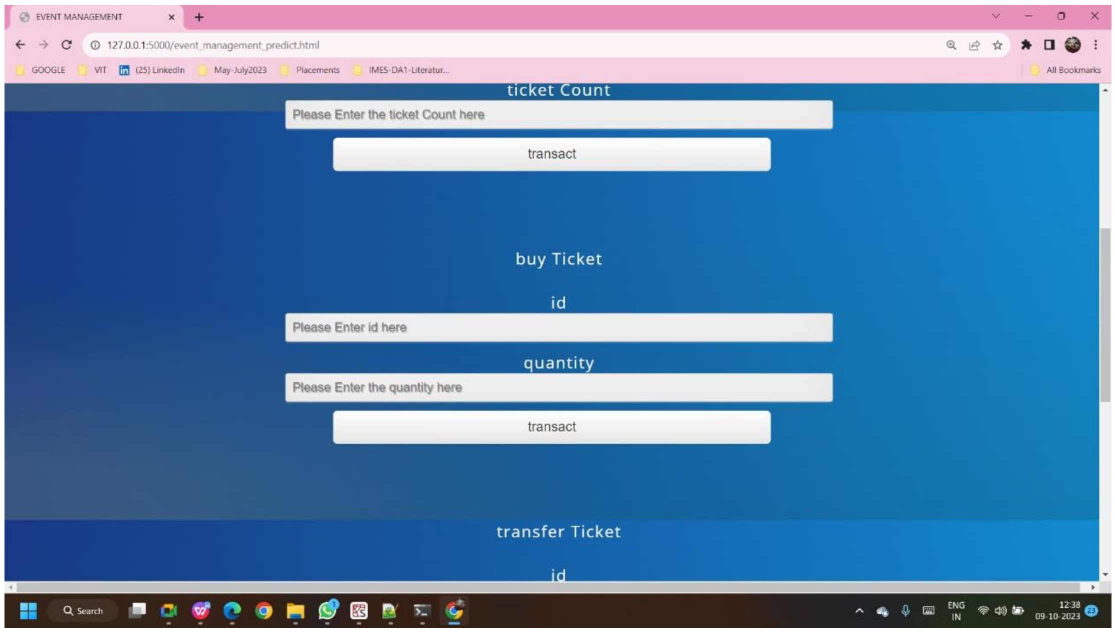
Price

Please Enter the price here

ticket Count

Please Enter the ticket Count here

transact



EVENT MANAGEMENT

127.0.0.1:5000/event_management_predict.html

ticket Count

Please Enter the ticket Count here

transact

buy Ticket

id

Please Enter id here

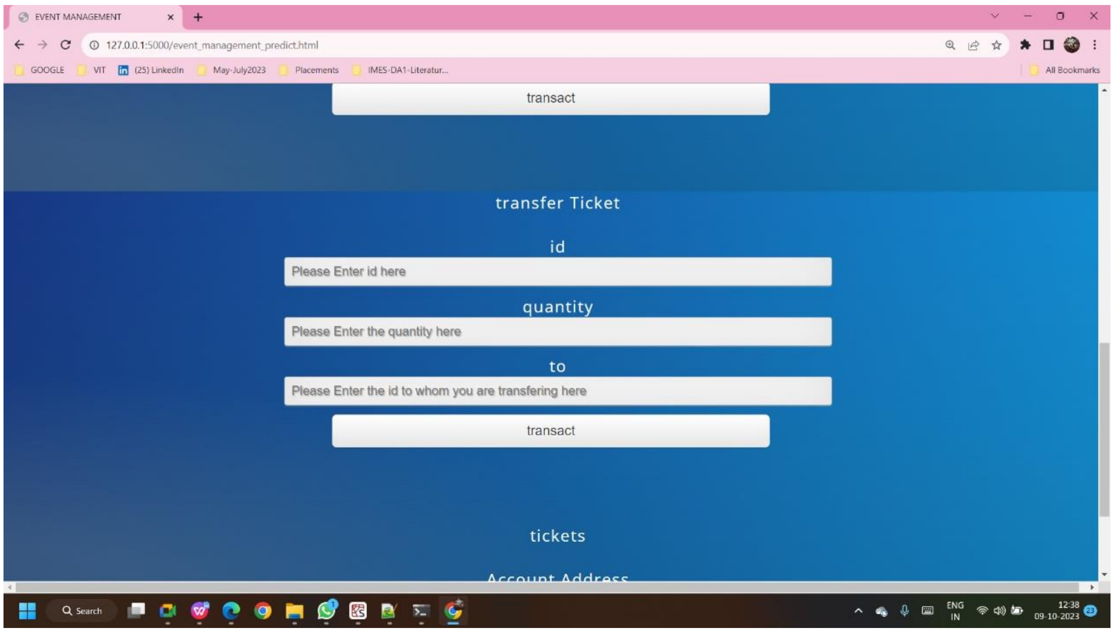
quantity

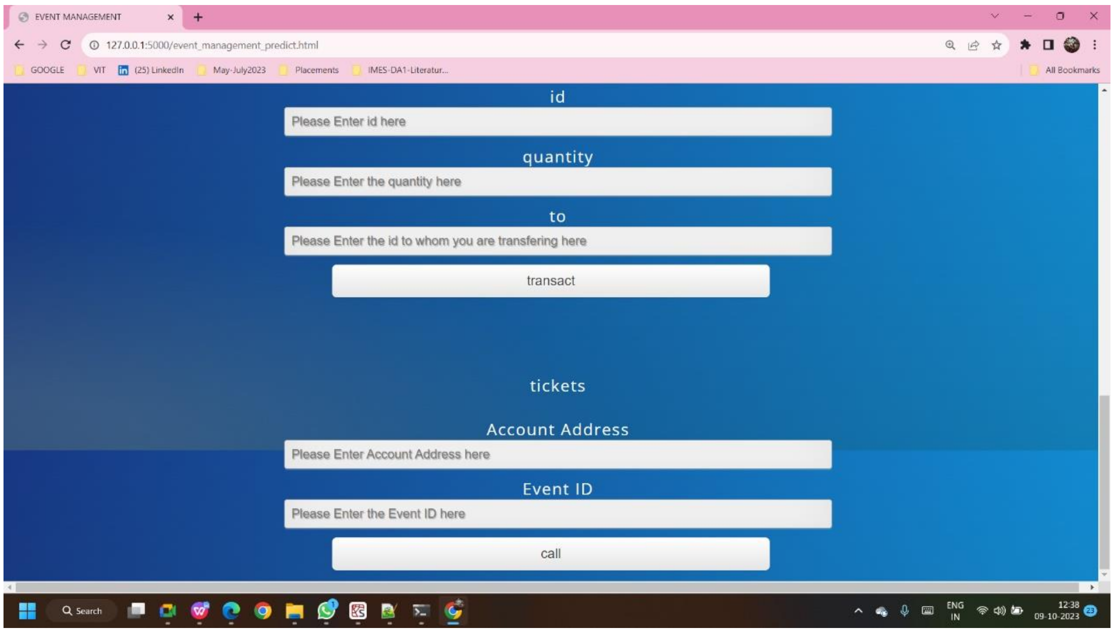
Please Enter the quantity here

transact

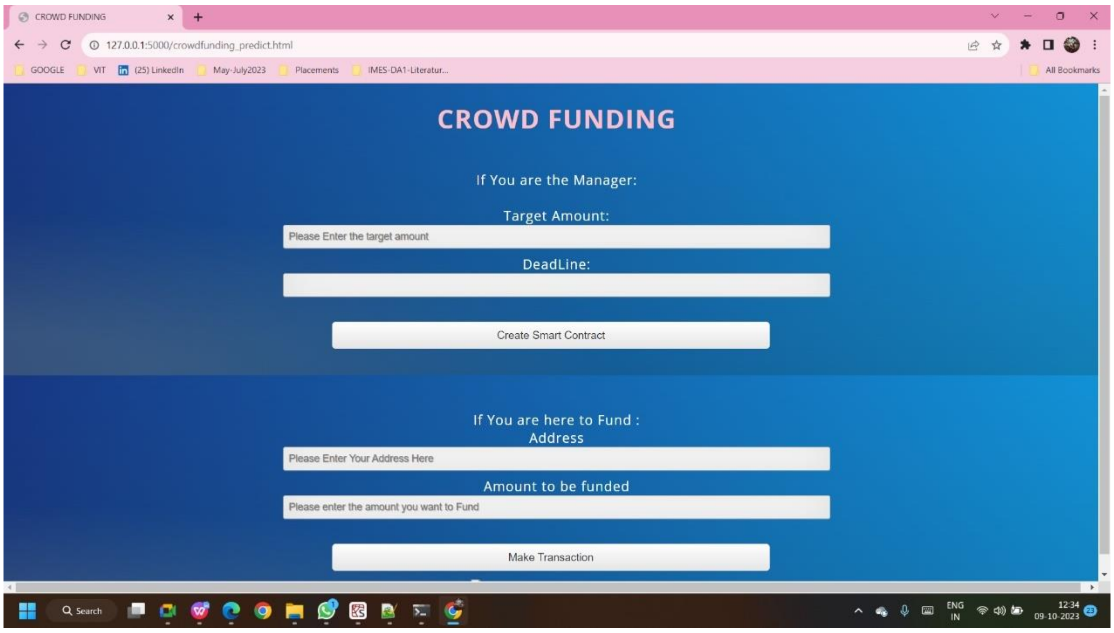
transfer Ticket

id


EVENT MANAGEMENT
127.0.0.1:5000/event_management_predict.html
transact
transfer Ticket
id
Please Enter id here
quantity
Please Enter the quantity here
to
Please Enter the id to whom you are transfering here
transact
tickets
Account Address


EVENT MANAGEMENT
127.0.0.1:5000/event_management_predict.html
id
Please Enter id here
quantity
Please Enter the quantity here
to
Please Enter the id to whom you are transfering here
transact
tickets
Account Address
Please Enter Account Address here
Event ID
Please Enter the Event ID here
call

8) Crowd Funding Smart-Contracts Page with required Parameters.


CROWD FUNDING
127.0.0.1:5000/crowdfunding_predict.html
CROWD FUNDING
If You are the Manager:
Target Amount:
Please Enter the target amount
DeadLine:
Create Smart Contract
If You are here to Fund :
Address
Please Enter Your Address Here
Amount to be funded
Please enter the amount you want to Fund
Make Transaction

RESULTS AND DISCUSSION

The blockchain-based platform, utilizing the given Solidity contracts, vividly demonstrates the immense capabilities and adaptability of blockchain technology across diverse sectors. The platform encompasses three crucial contracts: Banking System, Event Management, and Crowd Funding.

Within the Banking System contract, users can initiate account creation, conduct fund deposits and withdrawals, process cheques, and request loans. The decentralized architecture of blockchain guarantees transparency and security, mitigating fraudulent activities and bolstering trust in financial transactions.

The Event Management contract empowers the seamless creation and administration of events, encompassing key event details like name, date, ticket pricing, and distribution. Employing blockchain for event management ensures the immutability of event data, securely accessible only to authorized participants.

In the realm of fundraising, the Crowd Funding contract efficiently facilitates contributions to projects in a secure manner. Contributors have the added ability to vote on fund allocation, fostering a decentralized and democratic decision-making process.

The integration of these contracts into a blockchain-based platform unmistakably highlights the potential for secure, transparent, and decentralized transactions. The immutability of blockchain's ledger ensures data integrity, averting unauthorized alterations and providing a dependable source of truth. Moreover, the automation of processes through smart contracts streamlines operations and diminishes reliance on intermediaries.

In summary, the blockchain-based platform, underpinned by the provided contracts, exemplifies the transformative influence of blockchain technology on conventional practices within banking, event management, and fundraising. The outcomes underscore heightened security, transparency, and efficiency, underscoring the vast potential of blockchain to propel innovation across a wide array of sectors.

CONCLUSION

The blockchain portal, featuring Banking System, Event Management, and Crowd Funding contracts, demonstrates the pivotal role of blockchain in modern applications. By enhancing transparency, security, and automation, it optimizes financial operations, event management, and fundraising. The decentralized nature and immutability of blockchain instil trust and foster efficient decision-making. This portal showcases blockchain's potential to reshape various domains, heralding a future where trust, efficiency, and reliability is paramount.

REFERENCES

Literature Survey papers:

[1] https://www.ijser.org/researchpaper/Event-Management-Ticket-Booking-using-BlockChain.pdf

[2] https://ieeexplore.ieee.org/document/10140877

[3] https://www.researchgate.net/publication/357355833_Data_immutability_and_event_management_via_blockchain_in_the_Internet_of_things

[4] https://www.researchgate.net/publication/348125863_Artificial_Intelligence_Applications_for_Event_Management_and_Marketing

[5] https://ijrpr.com/uploads/V4ISSUE4/IJRPR12057.pdf

[6] https://www.mdpi.com/2076-3417/11/11/4811

[7] https://www.scirp.org/pdf/jcc_2021091613491949.pdf

[8] https://harvest.usask.ca/bitstream/handle/10388/13343/LIU-THESIS-2021.pdf?sequence=1&isAllowed=y

[9] https://ieeexplore.ieee.org/document/9997939

[10] https://www.ijraset.com/research-paper/real-estate-land-transaction-system-with-blockchain

[11] https://orca.cardiff.ac.uk/id/eprint/126790/7/Blockchain%20Paper%20-Sept_26_2019%20-%20updated.pdf

[12] https://www.irjet.net/archives/V6/i3/IRJET-V6I334.pdf

[13] https://ieeexplore.ieee.org/document/9154115

[14] https://www.researchgate.net/publication/344365086_Decentralized_Banking_Application_using_Block_chain_Technology

[15] https://www.osti.gov/servlets/purl/1475405

[16] “Decentralized Banking: The Future of Finance?" by Robert E. Wright and Richard Sylla (2018)

[17] "Blockchain-Based Decentralized Banking: A Regulatory and Economic Analysis"

[18] "Decentralized Finance (DeFi): An Overview" by Andrei Kouznetsov and Nikita Fadeev (2021)

[19] “Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets" by Fabian Schär (2019)

[20] “Decentralized Finance: Overview and Opportunities" by Ali Akhtar and Muhammad Ahsan Chishti (2021)

[21] https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4330476

[22] https://ieeexplore.ieee.org/document/9640888

[23] https://www.researchgate.net/publication/334142321_Blockchain_based_crowdfunding_systems

[24] https://ijrpr.com/uploads/V3ISSUE11/IJRPR8080.pdf

[25] https://www.dpublication.com/wp-content/uploads/2021/08/F6-8416.pdf

[26] https://scholarworks.calstate.edu/downloads/cf95jb83g

[27] https://ijirt.org/master/publishedpaper/IJIRT158590_PAPER.pdf

[28] https://www.warse.org/IJATCSE/static/pdf/file/ijatcse83932020.pdf

[29] https://www.mdpi.com/2076-3387/13/6/144

[30] https://ijcrt.org/papers/IJCRT2305284.pdf