CODESTREAX: REMOTE CODE EXECUTION ENGINE

PROJECT REPORT

Project Work Phase-I (ECS799)

BACHELOR OF TECHNOLOGY (CSE) ADCA (IBM)

FACULTY OF ENGINEERING & COMPUTING SCIENCES

TEERTHANKER MAHAVEER UNIVERSITY, MORADABAD

Session: Aug - Dec 2023

PROJECT GUIDE:
MR. AMIT SINGH
MR. ALOK SHARMA

SUBMITTED BY:
MANRAJ SINGH (TCA2057022)
AARJAV JAIN (TCA2057001)
LAKSHIKA CHAUDHARY (TCA2057021)

ACKNOWLEDGEMENT

I would like to express my gratitude and appreciation to all those who gave me the possibility to complete this report. Special thanks are due to my supervisor Mr. Amit Singh and Mr. Alok Sharma whose help, stimulating suggestions and encouragement helped me in all time of fabrication process and in writing this report.

I also sincerely thanks for the time spent proof reading and correcting my many mistakes. I also take this opportunity to express my deep sense of gratitude to our honorable Principal "Dr. Rakesh Kumar Dwivedi", TMU, for providing excellent academic climate in the college that made this endeavor possible.

I would also like to acknowledge and Many thanks go to the whole lecturer and supervisors who have given their full effort in guiding the team in achieving the goal as well as their encouragement to maintain our progress in track.

My profound thanks go to especially to my friends for spending their time in helping and giving support whenever I need it in fabricating my project. Finally, I express my gratitude to all the Teaching and Non-Teaching staff of "Computer Science Department," TMU for their timely support and suggestions.

Manraj Singh
Lakshika Chaudhary
Aarjav Jain

Place:

Date:

DECLARATION

We hereby declare that this Project Report titled CodeStreax: Remote Code Execution Engine submitted by us and approved by our project guide, the College of Computing Sciences, and Information Technology (CCSIT), Teerthanker Mahaveer University, Moradabad, is a bona fide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Student Name: Manraj Singh Signature

Student Name: Lakshika Chaudhary Signature

Student Name: Aarjav Jain Signature

Project Guide: (Internal) Mr. Amit Singh Signature

Project Guide: (Internal) Mr. Alok Sharma Signature

Project Guide: (External) Mr. Signature

Table of Contents

1. Project Title

Project Title: Code Streax: Remote Code Execution Engine

2. Domain

Project Domain: Software Development and Code Execution Platform

3. Problem Statement

Background:

The Code Execution Platform project is a response to the increasing demand for a secure and user-friendly coding environment for developers to practice, learn, and improve their programming skills. The field of software development is continuously evolving, and developers often require hands-on experience to hone their coding abilities. Traditional methods of practicing coding may lack the interactivity, security, and feedback necessary for effective skill development.

The background of the Code Streax RCE Engine project is set against the backdrop of an increasingly interconnected digital landscape where software applications play a central role in our daily lives. As technology advances, so do the methods employed by malicious actors to exploit vulnerabilities and compromise the security of these applications. Among the various threats faced by cybersecurity professionals, Remote Code Execution (RCE) vulnerabilities stand out as particularly insidious. RCE allows attackers to execute arbitrary code on a target system remotely, presenting a clear and present danger to the integrity and confidentiality of data.

In recent years, the frequency and sophistication of RCE attacks have risen substantially. This surge is driven by factors such as the proliferation of web-based applications, cloud computing, and the interconnectedness of systems. As organizations and individuals leverage the benefits of digital technologies, they also become more susceptible to cyber threats, including those exploiting RCE vulnerabilities.

Why This Topic is Selected:

The selection of this project topic is motivated by the growing need for a comprehensive and secure coding practice platform. Coding interviews, competitive programming, and

continuous learning have become integral to a developer's journey. However, existing solutions often fall short in providing a seamless and secure coding experience.

The Code Streax RCE Engine as a project topic is rooted in the recognition of the critical nature of RCE vulnerabilities, the inadequacy of current mitigation tools, and the need for a versatile, adaptive, and community-driven solution. As the digital landscape continues to evolve, the Code Streax RCE Engine seeks to stand at the forefront of cybersecurity, contributing to a more secure and resilient software environment.

1. **Critical Nature of RCE Vulnerabilities**

   RCE vulnerabilities represent a critical class of security issues due to their potential to grant unauthorized access and control over a system. Unlike many other types of vulnerabilities, RCE allows attackers not only to exfiltrate data but also to manipulate, delete, or create data within the compromised system. The severity of the consequences stemming from successful RCE exploits makes them a primary focus for cybersecurity professionals.

2. **Rising Incidence and Impact of RCE Attacks**

   The escalating frequency and impact of RCE attacks underscore the urgency of developing effective countermeasures. Organizations across various sectors, including finance, healthcare, and government, are increasingly falling victim to RCE exploits. The financial losses, regulatory scrutiny, and reputational damage resulting from these attacks highlight the need for proactive and sophisticated solutions to mitigate the risks posed by RCE vulnerabilities.

3. **Inadequacy of Current Mitigation Tools**

   Existing tools for RCE detection and prevention often fall short in addressing the evolving tactics employed by attackers. Signature-based detection methods struggle to keep pace with polymorphic threats, and traditional static and dynamic analysis approaches have limitations. The Code Streax RCE Engine is selected to fill this gap by introducing a comprehensive solution that combines advanced code analysis, real-time protection, and machine learning to enhance detection accuracy and effectiveness.

4. **Adaptability to Modern Software Ecosystems**

   The contemporary software landscape is characterized by diversity in programming languages, frameworks, and deployment architectures. Many existing RCE mitigation tools lack the adaptability to seamlessly integrate into this diverse ecosystem. The Code Streax RCE Engine is chosen for its commitment to versatility, being designed to be language and framework agnostic. This adaptability ensures its relevance and efficacy across a wide range of software applications and environments.

5. Community Collaboration and Ethical Considerations

The Code Streax RCE Engine project emphasizes community collaboration and ethical considerations. By providing a platform for developers and security experts to contribute to the vulnerability database, the project aims to create a collaborative and shared defense against emerging RCE threats. The ethical use guidelines integrated into the project underscore a commitment to responsible deployment and adherence to ethical principles in cybersecurity.

Problem to Solve:

The Code Execution Platform project addresses several key problems:

- **Secure Coding Environment:** Ensuring secure code execution to protect against potential security vulnerabilities and resource abuse.
- **User Authentication and Management:** Implementing a robust user authentication system to safeguard user data.
- **Real-time Feedback and Debugging:** Providing developers with immediate feedback on their code and tools for debugging.
- **Structured Challenge Library:** Organizing coding challenges by difficulty and topic, making it easy for users to find relevant practice material.
- **User Progress Tracking:** Allowing users to track their progress, view solved challenges, and update their profiles.
- **Administrator Control:** Empowering administrators with tools to manage user data and platform content efficiently.
- **Ubiquity and Sophistication of RCE Threats**

  The continuous evolution of technology and the interconnectivity of systems have expanded the attack surface for RCE threats. Web applications, cloud services, and networked systems are particularly vulnerable, as attackers exploit vulnerabilities to gain remote access and execute malicious code. Furthermore, the sophistication of attack techniques, including social engineering, evasion tactics, and code injection, underscores the need for a solution that can intelligently detect and prevent RCE exploits.

- **Current Limitations in RCE Mitigation**
    a. **Challenges in Detection**

    Existing RCE mitigation tools often rely on static analysis or dynamic analysis alone, each with its limitations. Static analysis struggles to identify novel threats, while dynamic analysis may introduce performance overhead. The challenge lies in developing a solution that combines the strengths of both approaches for comprehensive and efficient detection.

    b. **False Positives and Negatives**

    Balancing false positives and false negatives is a persistent challenge. False positives, where legitimate code is mistakenly flagged as malicious, can lead to alert fatigue and decreased productivity. Conversely, false negatives, where actual threats go undetected, pose a significant risk to the security of the system. The ideal solution should minimize both types of errors.

    c. **Lack of Adaptability**

    Many existing tools lack adaptability to diverse software environments. With the prevalence of polyglot programming and a variety of deployment architectures, a rigid solution may struggle to provide effective protection across different languages and frameworks. The Code Streax RCE Engine aims to overcome these limitations by offering adaptability and compatibility with a wide array of software ecosystems.

- **Projected Impact of RCE Exploits**
    a. **Data Breaches and Unauthorized Access**

    Successful exploitation of RCE vulnerabilities can lead to data breaches, exposing sensitive information to malicious actors. Unauthorized access to databases, user credentials, and proprietary information can result in severe consequences, including financial losses, reputational damage, and legal ramifications.

    b. **Compromise of Critical Systems**

    In critical infrastructure and enterprise environments, the compromise of systems through RCE exploits can result in the disruption of essential services. Industries such as healthcare, finance, and energy are particularly vulnerable, and the impact extends beyond financial losses to potential threats to public safety and national security.

    c. **Reputational Damage and Loss of Trust**

Beyond immediate technical implications, successful RCE exploits can lead to reputational damage and loss of trust among users and clients. Organizations that fall victim to such attacks may face a decline in customer confidence, impacting their market standing and long-term viability.

Why Code Streax RCE Engine?

The Code Streax RCE Engine is conceived as a solution that transcends the limitations of existing tools by adopting a holistic defense approach. It acknowledges the evolving nature of RCE threats and aims to provide a versatile, adaptable, and proactive mechanism for detecting, preventing, and mitigating the impact of these vulnerabilities.

By integrating advanced code analysis, real-time protection measures, and machine learning, the project seeks to address the identified gaps in current RCE mitigation strategies and contribute to the advancement of secure software development practices. The project's selection is rooted in the belief that a comprehensive defense mechanism is essential to navigate the intricate landscape of cybersecurity and protect against the ever-growing sophistication of RCE exploits.

4. Project Description

The Code Streax RCE Engine is a cutting-edge cybersecurity project designed to address the critical challenges posed by Remote Code Execution (RCE) vulnerabilities in software applications. This project is conceived as a holistic solution to proactively detect, prevent, and mitigate the risks associated with unauthorized code execution. The project encompasses a multifaceted approach that includes advanced algorithms, a comprehensive vulnerability database, and real-time protection mechanisms.

a. Scope of the Work

What will be done:

The Code Execution Platform project aims to develop a web-based coding environment similar to LeetCode, facilitating coding practice and skill improvement for developers. It will include the following key components:

- **User Registration and Authentication:** Users can sign up, log in, and manage their accounts securely using NextAuth.js.
- **Coding Challenges:** A library of coding challenges will be available, categorized by difficulty and topic.
- **Code Editor:** An integrated code editor with syntax highlighting and auto-completion will support multiple programming languages.
- **Code Execution Sandbox:** Code solutions will be executed in isolated Docker containers to ensure security and reliability.
- **Testing and Debugging:** Users can run code in real-time, view outputs, and debug their solutions.
- **Submission and Evaluation:** Users can submit code solutions, which will be evaluated for correctness.
- **User Profiles:** Users can track their progress, view solved challenges, and update their profiles.
- **Admin Panel:** Admins can manage user data and challenge content.

What will not be done: This project will not include advanced machine learning-powered code analysis, social networking features, or real-time communication tools. Additionally, while the platform may support multiple programming languages, it will not include support for all possible languages.

- **Versatile Application:** The Code Streax RCE Engine aims to be a versatile solution applicable across a broad spectrum of software applications. It is

designed to be language and framework agnostic, ensuring compatibility with various programming paradigms. This versatility makes the engine suitable for deployment in diverse environments, from web applications to backend services and more.

- **What Will Be Done:**
  - Development of code analysis algorithms that can comprehend and analyze code written in different programming languages.
  - Ensuring compatibility with various frameworks to accommodate diverse software development environments.
  - Conducting extensive testing to validate the engine's effectiveness across different application types and technology stacks.
- **What Will Not Be Done:**
  - The project does not focus on providing language-specific solutions but aims to create a universal engine applicable to a broad spectrum of languages and frameworks.
  - Customization for specific, niche programming languages may not be prioritized unless there is a significant demand from the user community.

- **Comprehensive Security Coverage:** The scope of the project extends to providing comprehensive coverage against RCE vulnerabilities. This includes both static and dynamic code analysis to identify potential threats throughout the software development life cycle. The project also involves the integration of a continually updated vulnerability database to proactively guard against emerging RCE risks.
  - **What Will Be Done:**
    - Implementation of advanced algorithms for static code analysis, enabling the detection of patterns indicative of RCE vulnerabilities without code execution.
    - Development of dynamic analysis algorithms to monitor code during runtime, identifying anomalous behavior that may signify potential RCE threats.
    - Regular updates to the vulnerability database to incorporate the latest known vulnerabilities and attack patterns.
  - **What Will Not Be Done:**
    - The project may not cover vulnerabilities beyond the scope of RCE, such as other types of injection attacks or data breaches, unless they directly contribute to RCE vulnerability exploitation.
    - The engine may not actively participate in vulnerability disclosure for non-RCE-related issues.

- **Real-time Protection Measures:** Real-time protection mechanisms are a crucial component of the project's scope. The engine aims to employ swift and pre-emptive actions upon identifying potential RCE threats. This real-time response is essential for minimizing the window of vulnerability and preventing the exploitation of identified weaknesses in the code.
  - **What Will Be Done:**
    - Development of mechanisms to respond immediately to identified RCE threats, preventing their exploitation.
    - Integration of real-time protection measures that can adapt to evolving attack vectors and techniques.
    - Testing and validation of the real-time protection module to ensure minimal impact on system performance.
  - **What Will Not Be Done:**
    - The project may not cover other types of security threats that do not fall under the category of RCE vulnerabilities.
    - Extensive intrusion detection or prevention features beyond the scope of RCE may not be a primary focus.

b. Project Modules

The Code Execution Platform project consists of the following modules:

- **User Management Module**

  Description: This module handles user registration, authentication, and user profile management using NextAuth.js.

- **Challenge Library Module**

  Description: Users can access coding challenges categorized by difficulty and topic, with challenge details and descriptions.

- **Code Editor Module**

  Description: An integrated code editor with syntax highlighting and auto-completion supports multiple programming languages.

- **Code Execution Sandbox Module**

Description: Code solutions are executed within isolated Docker containers to ensure security and prevent resource abuse.

- **Testing and Debugging Module**

  Description: Users can run code in real-time, view outputs, and debug their solutions directly within the platform.

- **Submission and Evaluation Module**

  Description: Users can submit code solutions, which are evaluated for correctness, and receive feedback.

- **User Profile Module**

  Description: Users can view their solved challenges, track progress, and customize their profiles.

- **Admin Panel Modul**

  Description: Admins have access to an admin panel for managing user data, challenge content, and platform operations.

- **Database Integration Module**

  Description: This module handles database interactions using PostgreSQL and Prisma ORM, storing user profiles, challenges, and submissions.

- **Code Analysis Module**

  Description: The heart of the Code Streax RCE Engine lies in its Code Analysis module. This module is divided into static and dynamic analysis components. The static analysis involves scrutinizing the source code without execution, identifying patterns, and potential vulnerabilities. Dynamic analysis, on the other hand, monitors the code during runtime, allowing the engine to detect anomalous behavior that might indicate RCE threats.

- **Vulnerability Database**

  Description: The Vulnerability Database module is a critical component that houses a comprehensive repository of known RCE vulnerabilities and attack patterns. Regularly updated, this database serves as a reference point during code analysis, enabling the engine to cross-reference against historical threat intelligence. The integration of this database ensures that the engine remains proactive in identifying both known and emerging RCE risks.

- **Real-time Protection Module**

Description: Upon identifying potential RCE threats, the Real-time Protection Module comes into play. This module is responsible for deploying pre-emptive measures to neutralize the threat before it can be exploited. Techniques such as code sandboxing, access control, and other security measures are implemented in real-time to secure the application and prevent unauthorized code execution.

- **Machine Learning Integration**

Description: The project incorporates machine learning algorithms to enhance its adaptability. These algorithms learn from patterns identified during static and dynamic analysis, improving the engine's ability to recognize novel RCE threats. Machine learning integration contributes to reducing false positives and enhances the engine's predictive capabilities, making it more adept at identifying potential vulnerabilities even before they are added to the vulnerability database.

5. Implementation Methodology

The implementation methodology of the Code Streax RCE Engine is a systematic and strategic approach designed to ensure the development of a robust and effective solution for detecting and preventing Remote Code Execution (RCE) vulnerabilities. The methodology encompasses various stages, each tailored to address specific aspects of the project's goals. Below are key points elaborating on the implementation methodology:

1. **Requirements Gathering:** Conduct interviews with stakeholders, including developers and administrators, to gather detailed requirements.
   Document user stories, feature requests, and technical specifications.

2. **System Design:** Create a system architecture using diagrams (e.g., ER Diagram, Class Diagram, Flowcharts) to visualize components and data flow.
   Define the database schema using Data Models, specifying tables, relationships, and attributes.
   Develop Use Case Diagrams to outline user interactions and system functionalities.

3. **Technology Stack Selection:** Choose Next.js and Node.js for the frontend and backend development. Select Docker for secure code execution. Opt for PostgreSQL as the relational database and Prisma ORM for database interactions. Implement NextAuth.js for user authentication.

4. **Development:** Develop the frontend interface using Next.js, HTML, CSS, and React components. Implement the backend logic using Node.js, creating APIs for user authentication, coding challenges, and user profiles.
   Set up Docker containers for code execution sandboxing.
   Use Prisma ORM to connect to the PostgreSQL database.

5. **Testing:** Conduct unit testing for individual components and functions.
   Perform integration testing to ensure seamless communication between frontend and backend.
   Test the code execution sandbox to verify security and functionality.
   Implement end-to-end testing to assess the entire system's performance.

6. **Deployment:** Deploy the application on a hosting platform like AWS, Heroku, or a custom server. Configure domain and SSL certificates for secure access. Set up continuous integration and continuous deployment (CI/CD) pipelines.

7. **User Acceptance Testing (UAT):** Involve a group of users to perform UAT, validating that the platform meets their expectations and requirements.
   Address any issues or feedback from UAT to fine-tune the application.

8. **Maintenance and Bug Tracking:** Establish a defect log and issue tracking system (e.g., Jira or GitHub Issues) to monitor and manage reported bugs and feature requests.
   Regularly update and maintain the application, applying security patches and improvements.
   Provide user support and feedback mechanisms for ongoing enhancements.

9. **Documentation:** Create user manuals and technical documentation to guide users and developers.
   Document system architecture, API endpoints, and database schemas for future reference.

10. **Scaling and Future Enhancements:** Monitor the platform's performance and scalability to accommodate a growing user base.
    Continuously enhance the platform based on user feedback and emerging technologies.

11. **Diagrams and Visual Aids:** Utilize DFDs (Data Flow Diagrams), ER Diagrams (Entity-Relationship Diagrams), Class Diagrams, and Flowcharts to illustrate system components and data flow.

12. **Testing Methodology:** Implement unit testing for each module/component.
    Perform integration testing to ensure proper communication and data exchange.
    Conduct system testing, focusing on user interaction and the overall user experience.
    Carry out load testing to evaluate system performance under high user loads.
    Perform security testing, including vulnerability assessment and penetration testing.

13. **Defect Log Management:** Establish a defect tracking system (e.g., Jira, Bugzilla) to record, prioritize, and assign issues. Assign severity and priority levels to defects. Regularly review and triage reported defects. Address and resolve defects based on priority, providing timely updates to stakeholders. Maintain a record of resolved issues for future reference.

14. **Research and Analysis**

a. Understanding RCE Threat Landscape

The initial phase involves an in-depth exploration of the Remote Code Execution threat landscape. This includes researching known RCE vulnerabilities, attack vectors, and evolving techniques employed by malicious actors. Understanding the threat landscape is crucial for informing the development of the engine and ensuring it remains aligned with current cybersecurity challenges.

b. Reviewing Existing Solutions

A comprehensive review of existing RCE detection and prevention solutions is conducted to identify their strengths, weaknesses, and gaps. This analysis informs the development team about the state of the art in RCE mitigation, enabling them to incorporate best practices and innovative features into the Code Streax RCE Engine.

15. Algorithm Design

a. Static Code Analysis Algorithms

The design phase focuses on developing advanced algorithms for static code analysis. These algorithms scrutinize source code without execution, identifying patterns and structures that may indicate RCE vulnerabilities. The goal is to create algorithms that are language-agnostic and capable of recognizing diverse coding paradigms.

b. Dynamic Analysis Algorithms

Complementary to static analysis, dynamic analysis algorithms are designed to monitor code during runtime. This involves simulating code execution scenarios and identifying anomalous behavior that could indicate potential RCE threats. The dynamic analysis algorithms contribute to a comprehensive understanding of code behavior and enhance the engine's detection capabilities.

c. Machine Learning Integration

To enhance adaptability, machine learning algorithms are integrated into the engine. These algorithms learn from patterns identified during static and dynamic analysis, improving the engine's ability to recognize novel RCE threats. The machine learning component contributes to reducing false positives and enhances the engine's predictive capabilities.

16. Database Integration

a. Curating the Vulnerability Database

A critical component of the Code Streax RCE Engine is its vulnerability database. This database is curated with information on known RCE vulnerabilities, historical attack patterns, and mitigation strategies. Regular updates to the database are implemented to ensure it remains current and relevant in the face of emerging threats.

b. Integration with Analysis Algorithms

The vulnerability database is seamlessly integrated into the analysis algorithms. During code analysis, the engine cross-references patterns and snippets against the database, allowing it to identify potential RCE threats by comparing them to known vulnerabilities. This integration enhances the engine's proactive capabilities by leveraging historical threat intelligence.

17**. Real-time Protection Mechanisms**

a. Developing Preemptive Measures

Upon identifying a potential RCE threat, the engine employs real-time protection mechanisms. These measures include code sandboxing, access control, and other preemptive actions designed to neutralize the threat before it can be exploited. The development of these mechanisms involves meticulous testing and validation to ensure their effectiveness without disrupting legitimate code execution.

a. Runtime Monitoring

Real-time protection is augmented by runtime monitoring, where the engine continuously observes code behavior during execution. This dynamic approach allows the engine to adapt to evolving threats and respond swiftly to anomalies. The runtime monitoring component is designed to be resource-efficient to minimize any impact on system performance.

18. Iterative Development and Testing

a. Agile Development Practices

The implementation of the Code Streax RCE Engine follows agile development practices, emphasizing iterative development and continuous improvement. This approach allows the development team to respond dynamically to evolving requirements, feedback, and emerging

threats. Regular sprints and feedback loops contribute to the refinement and enhancement of the engine's features.

b. Comprehensive Testing

Testing is an integral part of the implementation methodology. The engine undergoes rigorous testing, including unit testing, integration testing, and system testing. Test scenarios encompass a wide range of code structures, languages, and attack vectors to ensure the engine's reliability, accuracy, and resilience in real-world scenarios.

19. Documentation and User Guidance

a. Comprehensive Documentation

The development process includes the creation of comprehensive documentation covering every aspect of the Code Streax RCE Engine. This documentation serves as a guide for developers, security professionals, and users, providing insights into the engine's architecture, deployment, and best practices for optimal utilization.

b. User-Friendly Interface Design

The development team focuses on creating an intuitive and user-friendly interface for the Code Streax RCE Engine. The interface is designed to present vulnerability insights clearly, enabling users to interpret and respond to the engine's findings efficiently. User feedback is incorporated into the interface design to enhance usability.

20. Ethical Considerations and Responsible Deployment

a. Ethical Use Guidelines

Throughout the development process, ethical considerations guide decision-making. The team establishes clear guidelines for the ethical use of the Code Streax RCE Engine, emphasizing responsible disclosure and transparent communication. The goal is to ensure that the engine is used for ethical purposes and contributes positively to the cybersecurity community.

b. Responsible Deployment Practices

As the Code Streax RCE Engine is deployed, responsible practices are employed to mitigate potential risks. This includes secure deployment configurations, adherence to data protection regulations, and continuous monitoring to identify and address any unintended consequences. The

project team actively engages with users to encourage responsible deployment practices.

21. Community Collaboration Integration

a. Establishing Collaboration Channels

The project embraces community collaboration by establishing channels for developers and security experts to contribute to the vulnerability database. This collaborative model creates a dynamic ecosystem where knowledge is shared, and the engine benefits from the collective intelligence of a global community.

b. Feedback Integration

Community feedback is actively sought and integrated into the development process. The collaborative model ensures that the Code Streax RCE Engine remains responsive to the evolving needs of its user base, fostering a sense of community ownership and continuous improvement.

This implementation methodology ensures a systematic approach to developing, testing, and maintaining the Code Execution Platform while allowing for scalability and future improvements. It emphasizes user involvement, quality assurance, and continuous feedback throughout the project lifecycle.

This is a meticulous and strategic process that encompasses research, algorithm design, database integration, real-time protection mechanisms, iterative development, and ethical considerations.

By following these key steps, the project aims to deliver a versatile, adaptive, and effective solution for identifying and preventing RCE vulnerabilities in software applications. The iterative nature of the development process, coupled with community collaboration, positions the Code Streax RCE Engine as a dynamic and evolving tool at the forefront of cybersecurity.

6. Technologies to be used

a. Software Platform

a) Front-end

1. **Framework:** Next.js
2. **Languages:** HTML, CSS, JavaScript
3. **User Interface:** React components
4. **Code Editor:** Visual Studio Code (Editor)
5. **Web Browser:** Google Chrome, Mozilla Firefox, Microsoft Edge (for testing and compatibility)

b) Back-end

1. **Runtime Environment:** Node.js
2. **Server Framework:** Express.js
3. **Database Management System:** PostgreSQL
4. **Object-Relational Mapping (ORM):** Prisma
5. **Authentication:** NextAuth.js

b. Hardware Platform

RAM: Sufficient RAM to accommodate the application's requirements (minimum 4GB recommended).

Hard Disk: Adequate storage space for code execution containers and database (minimum 20GB recommended).

Operating System: Compatible with Node.js, Docker, and PostgreSQL (e.g., Linux-based distributions like Ubuntu or Windows).

Code Editor: Visual Studio Code or preferred code editor for development.

Web Browser: Google Chrome, Mozilla Firefox, Microsoft Edge (for development and testing).

7. Advantages of this Project

The Code Streax RCE Engine offers a comprehensive set of advantages that make it a formidable solution in the realm of cybersecurity. Its multifaceted approach, coupled with innovative features, positions it as an indispensable tool for identifying and preventing Remote Code Execution (RCE) vulnerabilities. Below are key points elaborating on the advantages of this project:

- **Comprehensive RCE Detection:**
  At the core of the Code Streax RCE Engine is its ability to conduct both static and dynamic code analysis. This dual approach ensures a thorough examination of software source code, enabling the identification of potential RCE vulnerabilities at different stages of the development lifecycle. Static analysis involves scrutinizing the code without execution, identifying patterns and structures that may indicate vulnerabilities. Dynamic analysis, on the other hand, monitors the code during runtime, allowing the engine to detect anomalous behavior that could signify a potential threat.

- **Skill Improvement for Developers:**
  Benefit: Users can practice coding, solve real-world challenges, and enhance their programming skills in a safe and structured environment.

- **Real-time Collaboration:**
  Benefit: Enables multiple developers to work simultaneously on the same codebase, fostering a collaborative and interactive development environment.

- **Shared Editing:**
  Benefit: Facilitates real-time, synchronized code editing, allowing team members to contribute seamlessly to projects.

- **Secure Code Execution:**
  Benefit: The use of Docker containers ensures code execution is isolated and secure, protecting both the platform and its users from potential security risks.

- **User-Friendly Interface:**
  Benefit: The intuitive Next.js-based interface offers an easy-to-navigate platform for writing, testing, and submitting code solutions.

- **Variety of Coding Challenges:**
  Benefit: Users can access a wide range of coding challenges categorized by difficulty and topic, allowing them to expand their knowledge and skills.

- **Versatility:**
  Benefit: Supports a wide array of programming languages, accommodating diverse developer preferences and project requirements.

- **Dynamic Language Integration:**
  Benefit: Incorporates a strategy for continuous integration of new languages based on user demand and emerging trends.

- **Real-Time Code Testing and Debugging:**
  Benefit: Users can test their code in real-time, receive immediate feedback, and debug their solutions, facilitating faster learning and problem-solving.

- **Personalized User Profiles:**
  Benefit: Users can track their progress, view their solved challenges, and customize their profiles, creating a personalized learning experience.

- **Authentication and Data Security:**
  Benefit: The use of NextAuth.js ensures user data is secure, and access is restricted to authorized users only.

- **Fine-Grained Access Control:**

  Benefit: Utilizes access control lists (ACLs) to manage user permissions effectively, securing sensitive operations.

- **Admin Management:**
  Benefit: Admins can efficiently manage user data, challenge content, and overall platform operations through an easy-to-use admin panel.

- **Real-time Mitigation:**
  One of the standout features of the Code Streax RCE Engine is its implementation of real-time protection mechanisms. Upon identifying a potential RCE threat, the engine takes immediate and decisive action to neutralize the threat before it can be exploited. This could involve deploying techniques such as code sandboxing, access control measures, or other pre-emptive actions. Real-time mitigation is crucial for minimizing the impact of RCE vulnerabilities, preventing unauthorized code execution, and safeguarding the integrity of the system.

- **Language and Framework Agnosticism:**

The versatility and adaptability of the Code Streax RCE Engine are fundamental strengths. The engine is designed to be agnostic to various programming languages and frameworks, ensuring that it can seamlessly integrate into diverse software applications. This flexibility is essential in catering to the evolving landscape of software development, where applications often utilize different technologies. The engine's ability to analyze code written in various languages enhances its applicability across a wide range of software environments.

- **Proactive Security Measures:**
  The integration of a constantly updated vulnerability database sets the Code Streax RCE Engine apart as a proactive security solution. This database serves as a repository of known RCE vulnerabilities and attack patterns. By cross-referencing code snippets and patterns against this database, the engine can proactively identify potential threats even before they are exploited. This proactive approach to security ensures that the engine stays ahead of emerging RCE risks, providing a robust defense against the ever-evolving tactics employed by malicious actors.

- **Reduced False Positives:**
  The Code Streax RCE Engine is designed with a focus on accuracy in vulnerability detection. By incorporating advanced algorithms and machine learning capabilities, the engine aims to reduce false positives. False positives can be a significant concern in security solutions, as they may lead to unnecessary alerts and place an additional burden on developers. The engine's commitment to minimizing false positives contributes to its usability and ensures that developers can confidently address legitimate security concerns.

- **Adaptability to Emerging Threats:**
  The proactive and adaptive nature of the Code Streax RCE Engine positions it as a resilient defense against emerging threats. The engine's integration of machine learning algorithms enables it to learn from new patterns and behavior, continually adapting to novel RCE attack vectors. This adaptability ensures that the engine remains effective in the face of constantly evolving cybersecurity challenges, providing organizations with a reliable and future-proof solution.

- **Scalability and Performance**
  The scalability of the Code Streax RCE Engine is crucial for its effective deployment in diverse environments. Whether applied to small-scale applications or large-scale enterprise systems, the engine is designed to scale seamlessly. Its performance is optimized to ensure minimal impact on system resources while maintaining efficient and swift RCE vulnerability detection and

prevention. This scalability and performance optimization contribute to the engine's suitability for a wide range of applications, from startups to enterprise-level deployments.

- **User-Friendly Interface**
  An intuitive and user-friendly interface is a key aspect of the Code Streax RCE Engine's design. The interface provides developers and security teams with clear insights into identified vulnerabilities, their severity levels, and recommended actions. The user-friendly dashboard enhances user experience, making it easier for stakeholders to navigate and interpret the engine's findings. This accessibility is essential for ensuring that the Code Streax RCE Engine is a practical and usable tool for both security professionals and developers.

- **Provisions for Community Collaboration**
  The Code Streax RCE Engine's integration with community collaboration features fosters a sense of shared responsibility within the cybersecurity community. By providing a platform for developers and security experts to contribute to the vulnerability database, the engine benefits from a collective pool of knowledge. This collaborative approach not only enriches the engine's understanding of RCE threats but also creates a network of experts actively engaged in fortifying the engine against emerging challenges. Community collaboration ensures that the Code Streax RCE Engine remains a dynamic and evolving solution.

- **Ethical Considerations and Responsible Use:**
  Benefit: Ethical considerations are embedded in the Code Streax RCE Engine's development and deployment. The project emphasizes responsible use and ethical standards to ensure that the engine serves its intended purpose without being misused for malicious intent. A commitment to transparency, responsible disclosure, and adherence to ethical principles is integral to maintaining the trust of users, organizations, and the broader cybersecurity community.

8. Future Scope and further enhancement of the Project

The Code Streax RCE Engine, while already a powerful tool in the realm of cybersecurity, holds immense potential for future enhancements and broader applications. The following outlines key areas for future development, ensuring the continuous evolution of the project to address emerging challenges and technologies.

- **Code Collaboration and Pair Programming:** Enhance the platform to support collaborative coding, enabling multiple users to work on the same code simultaneously, which is beneficial for pair programming sessions and collaborative coding challenges.

- **Advanced Code Analysis and Feedback:** Implement code analysis tools that provide in-depth feedback on code quality, performance, and adherence to best practices.
  Offer suggestions for code optimization and improvements.

- **User-Generated Content and Challenges:** Allow users to create and share their coding challenges and exercises, fostering a sense of community engagement and knowledge sharing.

- **Leaderboards and Gamification:** Introduce leaderboards and a gamification system to encourage healthy competition among users, providing recognition and rewards for top performers.

- **Code Sharing and Social Integration:** Enable users to share their code solutions on social media platforms and coding communities. Implement social login options to simplify registration and enhance user engagement.

- **Integration with Learning Resources:** Integrate educational resources such as tutorials, documentation, and video lectures related to coding challenges, providing a comprehensive learning experience.

- **Machine Learning-Powered Recommendations:** Implement machine learning algorithms to recommend coding challenges and learning paths tailored to individual user interests and skill levels.

- **Support for More Programming Languages:** Expand the list of supported programming languages, accommodating a wider range of coding preferences and challenges.

- **Real-Time Collaboration Tools:** Integrate real-time communication tools like chat or video conferencing for users engaged in collaborative coding sessions.

- **Mobile Applications:** Develop mobile applications (iOS and Android) for users to access the platform on the go, ensuring a seamless mobile experience.

- **Offline Coding:** Implement an offline mode, allowing users to download coding challenges and work on them without an internet connection.

- **Accessibility and Internationalization:** Enhance accessibility features to ensure the platform is usable by individuals with disabilities. Support multiple languages for a global user base.

- **Analytics and Data Insights:** Implement analytics tools to gather user data and provide insights into user behavior, challenge popularity, and platform usage trends.

- **Security Audits and Penetration Testing:** Conduct regular security audits and penetration testing to identify and mitigate vulnerabilities in the platform.

- **Monetization Options:** Explore monetization strategies such as premium subscription plans, advertising, or offering advanced features to generate revenue.

- **APIs and Integration with Other Platforms:** Develop APIs to allow integration with other educational or coding platforms, enabling seamless data sharing and interaction.

- **Machine Learning Integration:** Integrating machine learning (ML) into the Code Streax RCE Engine presents an exciting avenue for improving its detection capabilities.
  - By leveraging ML algorithms, the engine can learn from patterns and behaviors identified during its analysis of code structures. This adaptive learning mechanism allows the engine to recognize previously unseen RCE threats and adapt to evolving attack techniques. The introduction of ML can significantly enhance the engine's predictive capabilities, making it more adept at identifying potential vulnerabilities even before they are added to the vulnerability database.
  - Moreover, machine learning can contribute to reducing false positives by refining the engine's understanding of legitimate code structures and behaviors. This refinement is crucial for minimizing the impact on developers and system performance, ensuring that the engine's alerts are accurate and actionable.

- **Cloud Integration:** As software applications increasingly migrate to cloud environments, integrating the Code Streax RCE Engine with cloud security services becomes a logical progression.

- Cloud integration offers several advantages, including scalability, centralized management, and the ability to provide real-time protection across distributed systems.
- By seamlessly integrating with cloud platforms, the engine can extend its reach and effectiveness, offering a unified solution for RCE detection and prevention in cloud-native applications. This enhancement ensures that the engine remains applicable and efficient in modern, dynamic computing environments. Additionally, leveraging cloud services can facilitate automatic updates, allowing the engine to stay current with the latest threat intelligence and security measures.

- **Community Collaboration:** The Code Streax RCE Engine's potential for community collaboration is a pivotal aspect that can be further developed to create a dynamic and continually updated defense mechanism against RCE vulnerabilities.
  - Establishing a collaborative platform or an open-source model invites contributions from a diverse community of developers and security experts.
  - In this collaborative framework, community members can contribute to expanding the vulnerability database by sharing their insights, experiences, and newly discovered RCE patterns. This collective effort not only enriches the engine's knowledge base but also fosters a sense of shared responsibility within the cybersecurity community. A community-driven approach ensures that the Code Streax RCE Engine remains resilient and adaptive, benefiting from the collective expertise of a global network of security enthusiasts.

- **Integration with DevSecOps Practices:** As organizations increasingly embrace DevSecOps practices, integrating the Code Streax RCE Engine into the continuous integration and continuous deployment (CI/CD) pipelines becomes a logical progression.
  - Embedding the engine into the DevSecOps workflow enables developers to identify and address RCE vulnerabilities early in the development process.
  - Automated security checks within the CI/CD pipeline can help prevent the introduction of vulnerabilities into the codebase, fostering a proactive security culture. This integration aligns with the industry's shift towards shifting security left in the development lifecycle, ensuring that security considerations are an integral part of the software development process.

- **Enhanced User Interface and Reporting:** Improving the user interface and reporting capabilities of the Code Streax RCE Engine is essential for enhancing user experience and facilitating efficient threat management.
  - A user-friendly interface that provides clear insights into identified vulnerabilities, their severity, and recommended actions empowers developers and security teams to respond effectively.

- Advanced reporting features, including trend analysis, historical data, and mitigation success rates, can aid in risk assessment and decision-making. Customizable dashboards and real-time alerts contribute to a proactive security posture, enabling organizations to stay ahead of potential threats.

- **Regulatory Compliance Features:** With an increasing focus on regulatory compliance in the cybersecurity landscape, future enhancements could include features tailored to assist organizations in meeting industry-specific security standards.
  - Integrating compliance checks within the Code Streax RCE Engine ensures that applications align with relevant regulations and standards, such as GDPR, HIPAA, or PCI DSS. This enhancement not only fortifies security but also streamlines the compliance process for organizations operating in regulated industries.

- **Internationalization and Localization:** To cater to a global user base, incorporating internationalization and localization features into the Code Streax RCE Engine can enhance its accessibility.
  - This includes providing language support, regional customization, and accommodating diverse coding practices. Ensuring that the engine is accessible to a broad spectrum of users contributes to its widespread adoption and effectiveness in various cultural and linguistic contexts.

- **Cross-Platform Compatibility:** Ensuring cross-platform compatibility is crucial for the Code Streax RCE Engine to accommodate diverse software environments.
  - Enhancements in this area involve optimizing the engine for different operating systems, development frameworks, and deployment environments. This ensures that the engine seamlessly integrates into various software ecosystems, making it a versatile and universally applicable solution.

These enhancements aim to further enrich the Code Execution Platform, making it more engaging, educational, and user-friendly while catering to a diverse user base with varying needs and preferences.

9. Team Details

Project Name & ID	Course Name	Student ID	Student Name	Role	Signature
CodeStreax	B.Tech CSE IBM	TCA2057022	Manraj Singh	Developer	
		TCA2057001	Aarjav Jain	Developer	
		TCA2057021	Lakshika Chaudhary	Developer	

10. Conclusion

In conclusion, the Code Execution Platform project represents an innovative approach to creating a versatile coding environment for developers. It offers a comprehensive solution for coding practice, skill enhancement, and code evaluation, ensuring both security and usability.

Key innovations in our approach include the integration of Docker containers for secure code execution, the utilization of NextAuth.js for user authentication, and the implementation of Prisma ORM for database interactions. These technologies provide a robust and scalable foundation for the platform.

Some of the main achievements of this project include:

- **Secure Code Execution:** The use of Docker containers ensures the secure execution of code, preventing security vulnerabilities and resource abuse.
- **User-Centric Design:** The user-friendly Next.js-based interface, coupled with real-time code testing and debugging, offers an intuitive and engaging experience for developers.
- **Comprehensive User Management:** NextAuth.js simplifies user registration and authentication, providing a secure environment for user data.
- **Scalable and Maintainable:** The project's modular structure and use of industry-standard technologies make it highly scalable and easy to maintain.
- **Database Integration:** PostgreSQL and Prisma ORM enable efficient data storage and retrieval, enhancing user profiles and challenge management.
- **Admin Control:** The admin panel empowers administrators to manage user data and platform content seamlessly.

One of the standout features of this system is its secure, real-time code execution sandbox, which offers developers a safe environment to practice, experiment, and learn. This, combined with an extensive library of coding challenges and user-friendly tools, sets this platform apart from others in the field.

- **Significance of Code Streax RCE Engine**

In conclusion, the Code Streax RCE Engine stands as a pioneering solution in the field of cybersecurity, addressing the critical and persistent issue of Remote Code Execution vulnerabilities. The significance of this project lies in its multifaceted approach towards fortifying software applications against malicious exploits. In an era where cyber threats are becoming increasingly sophisticated, the Code Streax RCE Engine offers a proactive and dynamic defense mechanism, underscoring the project's relevance in the contemporary digital landscape.

- **Holistic Defense Strategy**

One of the primary strengths of the Code Streax RCE Engine is its adoption of a holistic defense strategy. By incorporating both static and dynamic code analysis, the engine ensures a comprehensive examination of software source code. This approach allows for the identification of potential vulnerabilities at different stages of the software development life cycle, reducing the likelihood of oversight. The real-time protection mechanisms further elevate the engine's efficacy, providing an immediate response to detected threats and preventing their exploitation.

- **Versatility and Adaptability**

The versatility and adaptability of the Code Streax RCE Engine emerge as key factors contributing to its effectiveness. In recognizing the diverse landscape of software development, the engine is designed to be agnostic to programming languages and frameworks. This adaptability ensures that the engine can be seamlessly integrated into various applications, regardless of their technological stack. This flexibility positions the Code Streax RCE Engine as a universal tool for developers and organizations seeking a robust defense against RCE vulnerabilities across different software environments.

- **Collaboration and Community Involvement**

A noteworthy aspect of the Code Streax RCE Engine is its potential to foster collaboration within the cybersecurity community. By establishing a platform for community contributions to the vulnerability database, the project encourages a collective effort to stay ahead of emerging threats. This collaborative approach not only enhances the accuracy and relevance of the vulnerability database but also creates a network of experts actively engaged in fortifying the engine against the ever-evolving tactics employed by malicious actors.

- **Continuous Improvement and Future Prospects**

The Code Streax RCE Engine does not represent the culmination of efforts but rather serves as a foundation for continuous improvement and future innovation. As the cybersecurity landscape evolves, the project can be enhanced in various ways to stay at the forefront of defense mechanisms against RCE vulnerabilities. The envisioned integration of machine learning algorithms and cloud services presents exciting possibilities for elevating the engine's predictive capabilities and scalability. Furthermore, the establishment of community collaboration channels opens avenues for a collective, agile response to emerging threats.

- **Contribution to Secure Software Development**

In the broader context of secure software development, the Code Streax RCE Engine contributes significantly to the establishment of robust cybersecurity practices. By providing a tool that can identify, prevent, and adapt to RCE vulnerabilities, the project aligns with the industry's growing emphasis on proactive security measures. This is particularly crucial as organizations increasingly recognize the cost-effectiveness and reputational benefits of pre-emptive security, minimizing the potential impact of cyber incidents on both business operations and user trust.

- **Ethical Considerations**

In the development and deployment of the Code Streax RCE Engine, ethical considerations are paramount. As with any security tool, ethical use and responsible disclosure play a pivotal role in ensuring that the engine serves its intended purpose without being misused for malicious intent. It is imperative for the developers, users, and stakeholders to adhere to ethical standards, promote transparency, and prioritize the responsible utilization of the engine to uphold the principles of digital security.

- **Final Thoughts**

In conclusion, the Code Streax RCE Engine not only addresses a critical problem in cybersecurity but also represents a commitment to advancing the field. Its holistic defense strategy, adaptability, and collaborative features position it as a cutting-edge solution with the potential to shape the future of RCE vulnerability mitigation. As the project moves forward, maintaining a focus on ethical considerations and community involvement will be integral to ensuring its positive impact on the broader landscape of secure software development. The Code Streax RCE Engine is not merely a project; it is a testament to the continuous pursuit of excellence and resilience in the face of evolving cyber threats.

11. References

https://nextjs.org/docs

https://docs.docker.com/

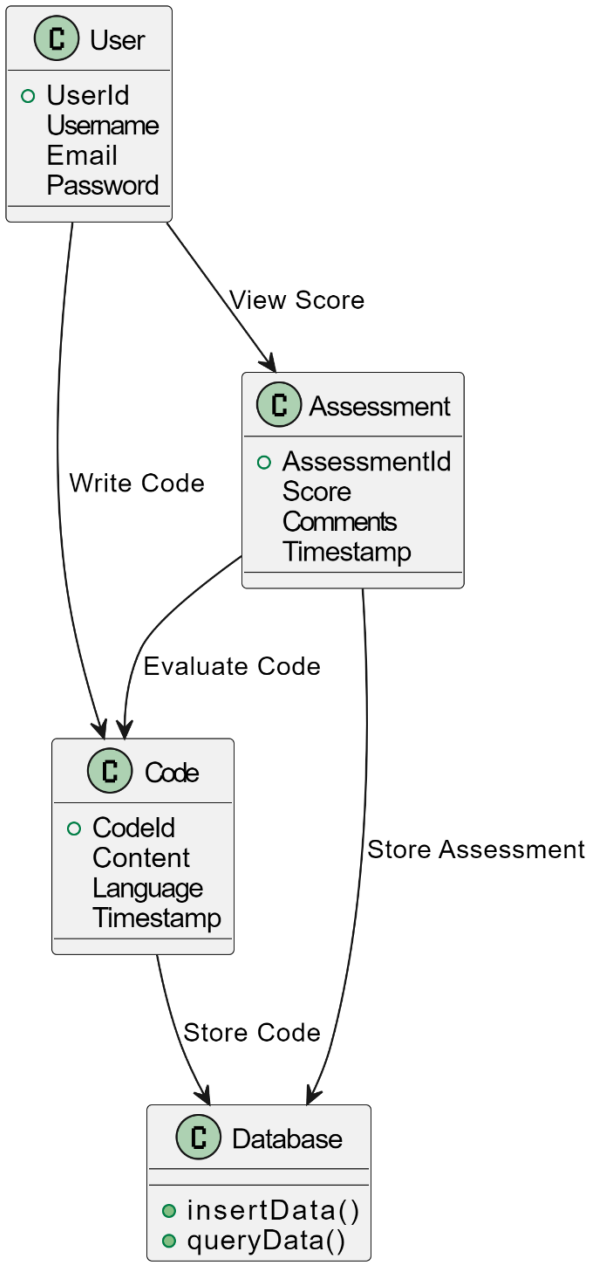
https://www.prisma.io/docs/

https://next-auth.js.org/getting-started/introduction

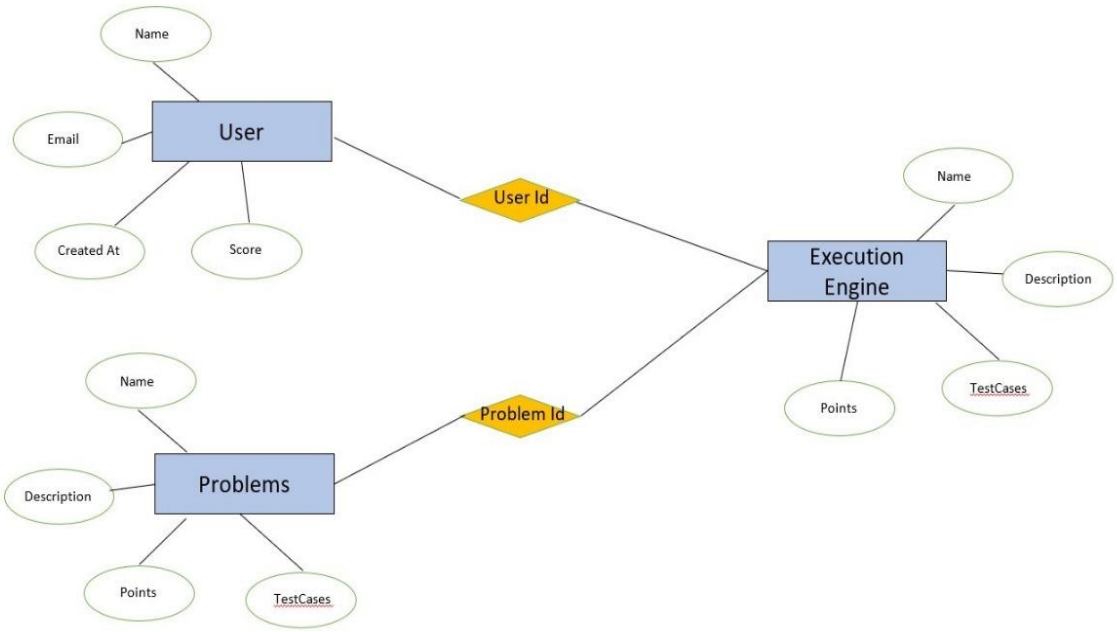
https://www.postgresql.org/docs/

https://code.visualstudio.com/docs

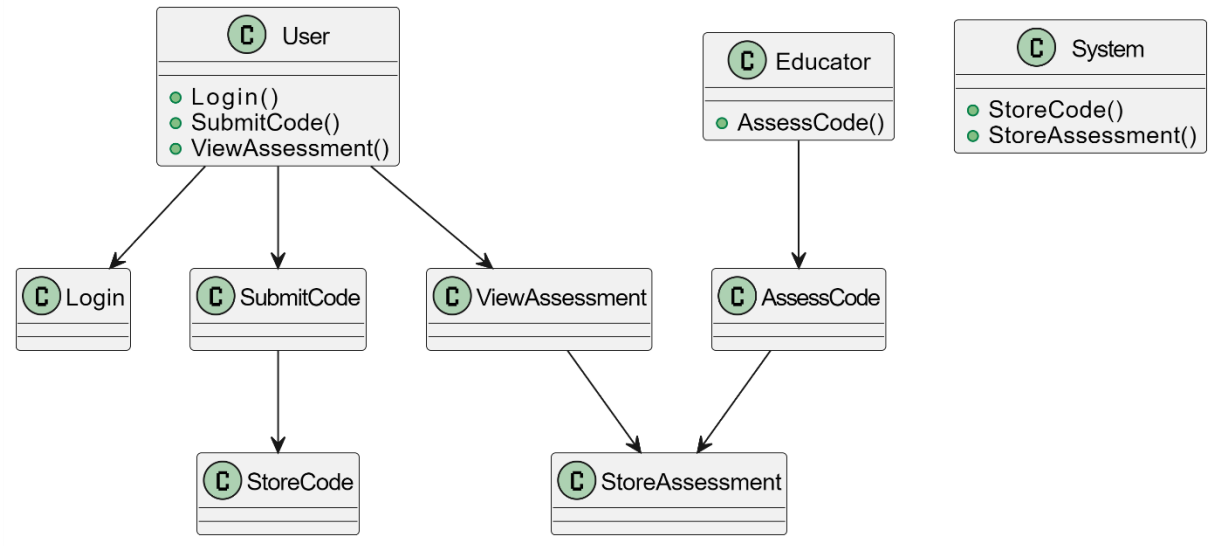
Annexure A
Data Flow Diagram (DFD)



Annexure B
Entity-Relationship Diagram (ERD)



Annexure C
Use-Case Diagram (UCD)



Annexure D
Data Dictionary (DD)
(Mandatory)

User Table (USR)

	A	B	C
1	Fields	Data Type	Description
2	UserId	Text	Unique identifier for a user
3	Username	Text	User's chosen username
4	Email	Text	User's email address
5	Password	Text	User's hashed password

Code Submission Table (CodeSubmission)

	A	B	C
1	Fields	Data Type	Description
2	CodeId	Text	Unique identifier for a code submission
3	Content	Text	Code content submitted by the user
4	Language	Text	Programming language of the code
5	Timestamp	DateTime	Date and time of the code submission

Assessment Table (Assessment)

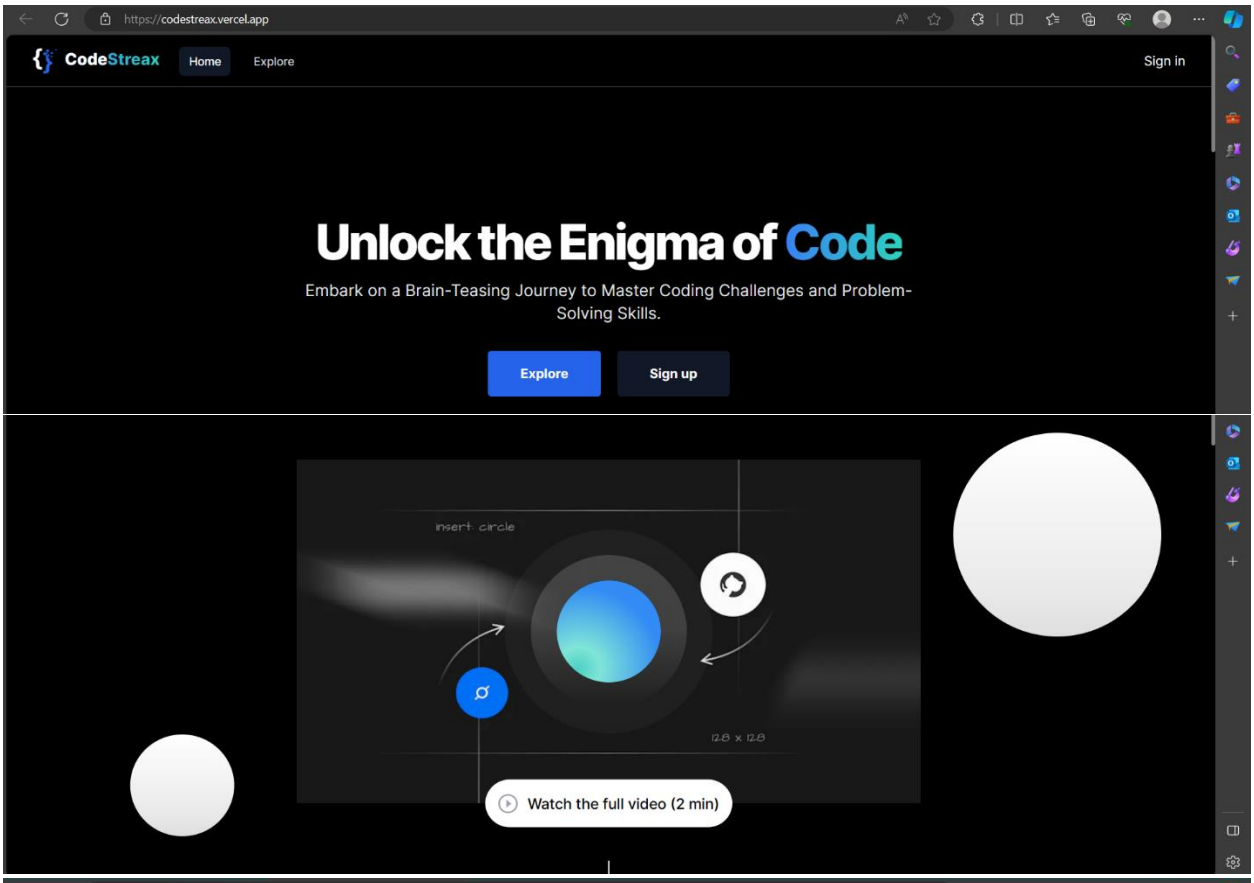
Fields	Data Type	Description
AssessmentId	Text	Unique identifier for an assessment
CodeId	Text	Foreign key referencing the associated code submission
Score	Number	Numeric score assigned to a code submission
Comments	Text	Comments provided by an educator during assessment
Timestamp	DateTime	Date and time of the assessment

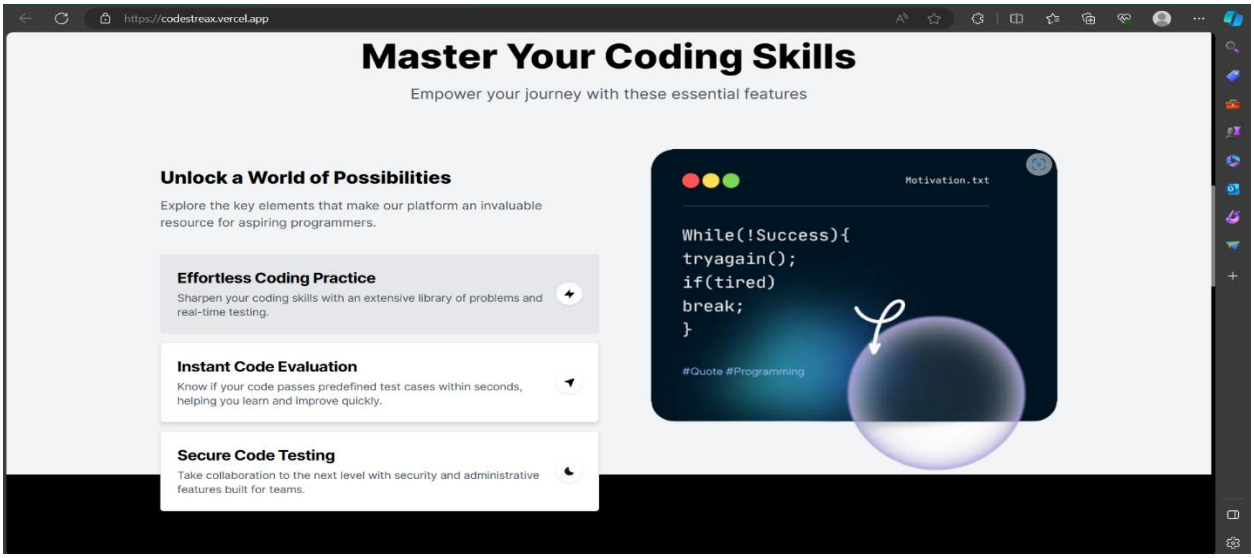
System Configuration Table (SystemConfig)

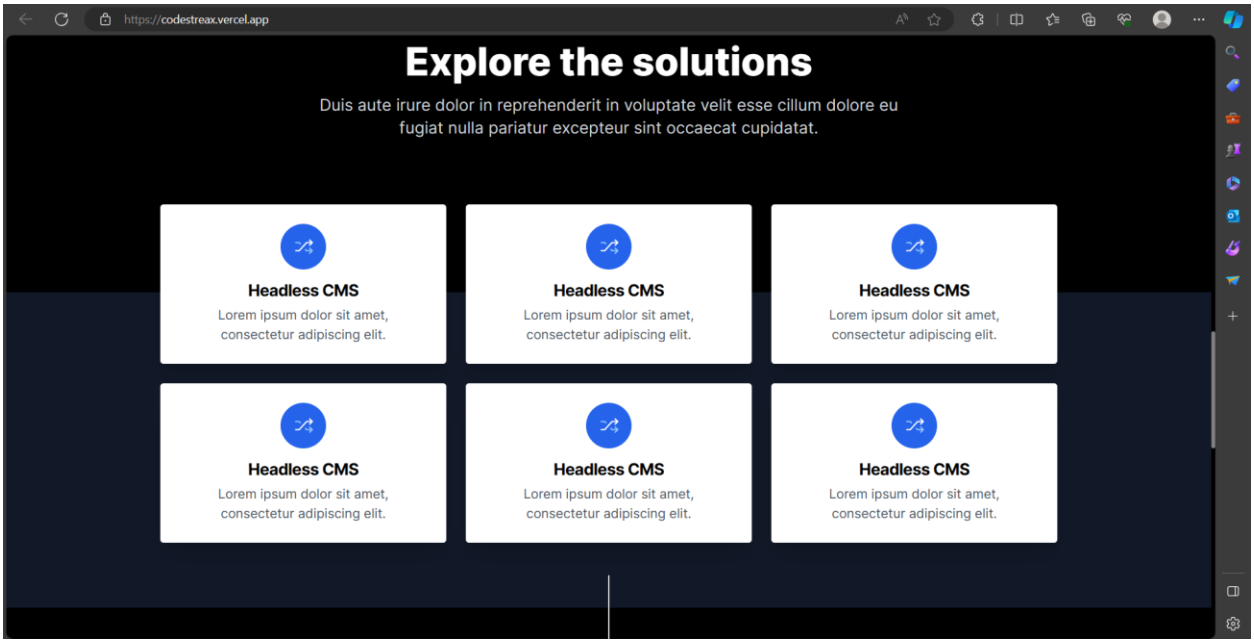
Fields	Data Type	Description
DATABASE_URL	Text	Configuration variable for the database connection URL
AWS_ACCESS_KEY_ID	Text	Access key for AWS services
AWS_SECRET_ACCESS_KEY	Text	Secret key for AWS services

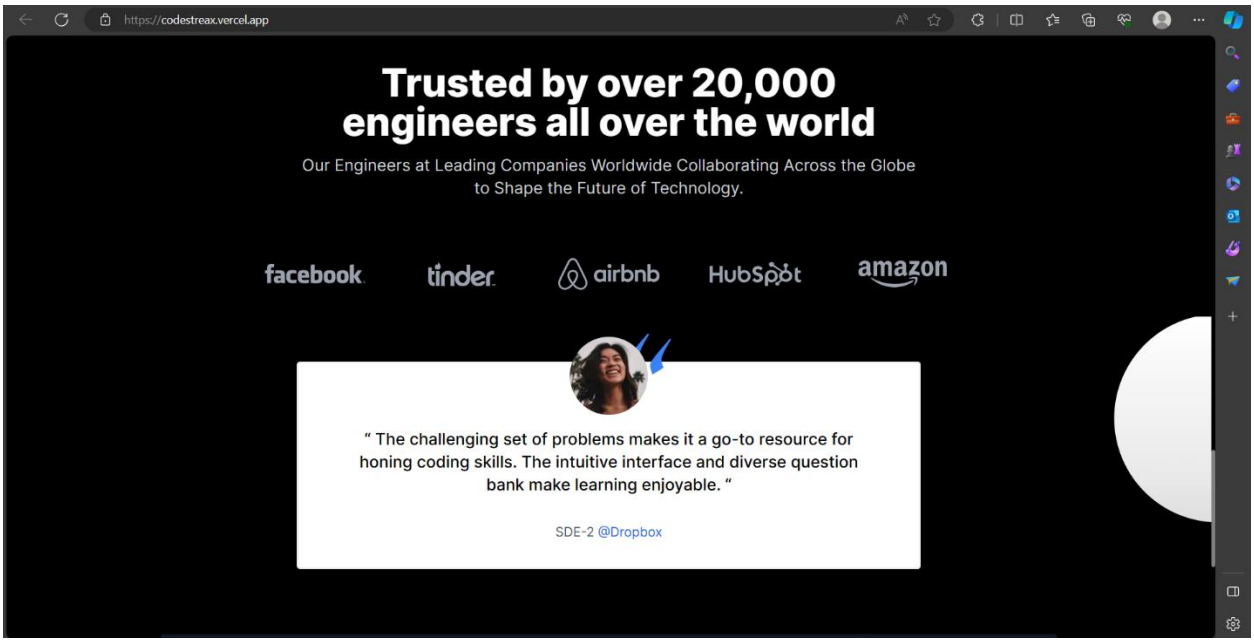
Annexure E
Screen Shots

Home Page:

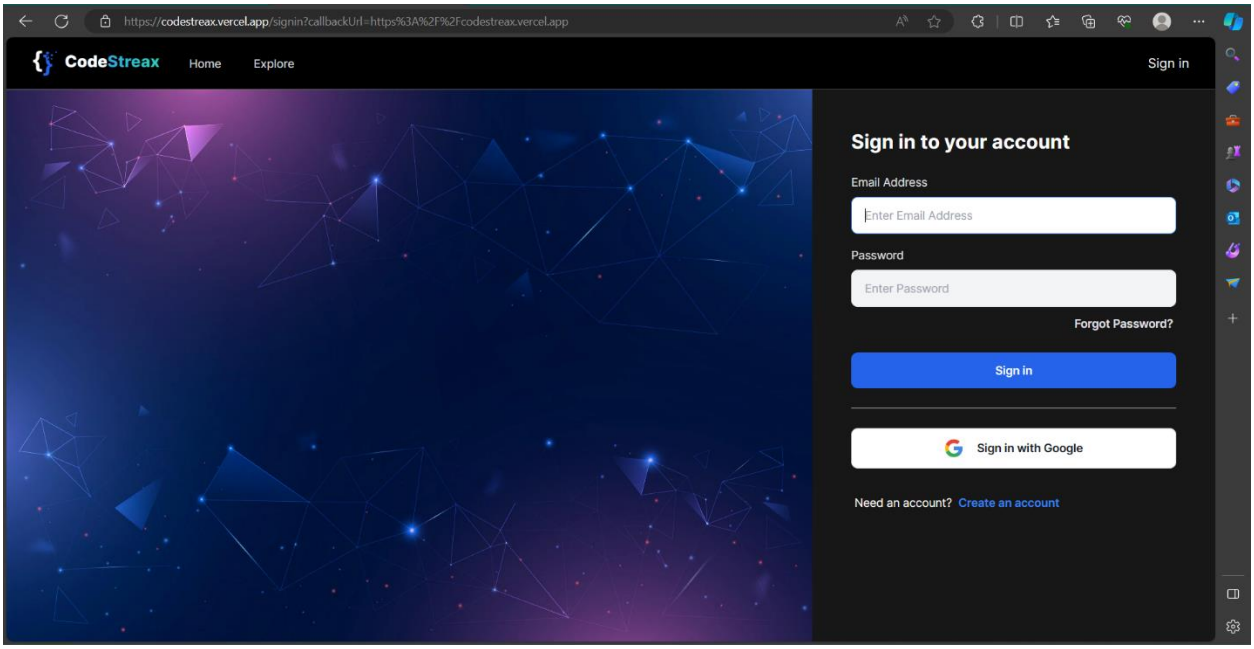





https://codestreax.vercel.app
Explore the solutions
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur excepteur sint occaecat cupidatat.
Headless CMS
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Headless CMS
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Headless CMS
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Headless CMS
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Headless CMS
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Headless CMS
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

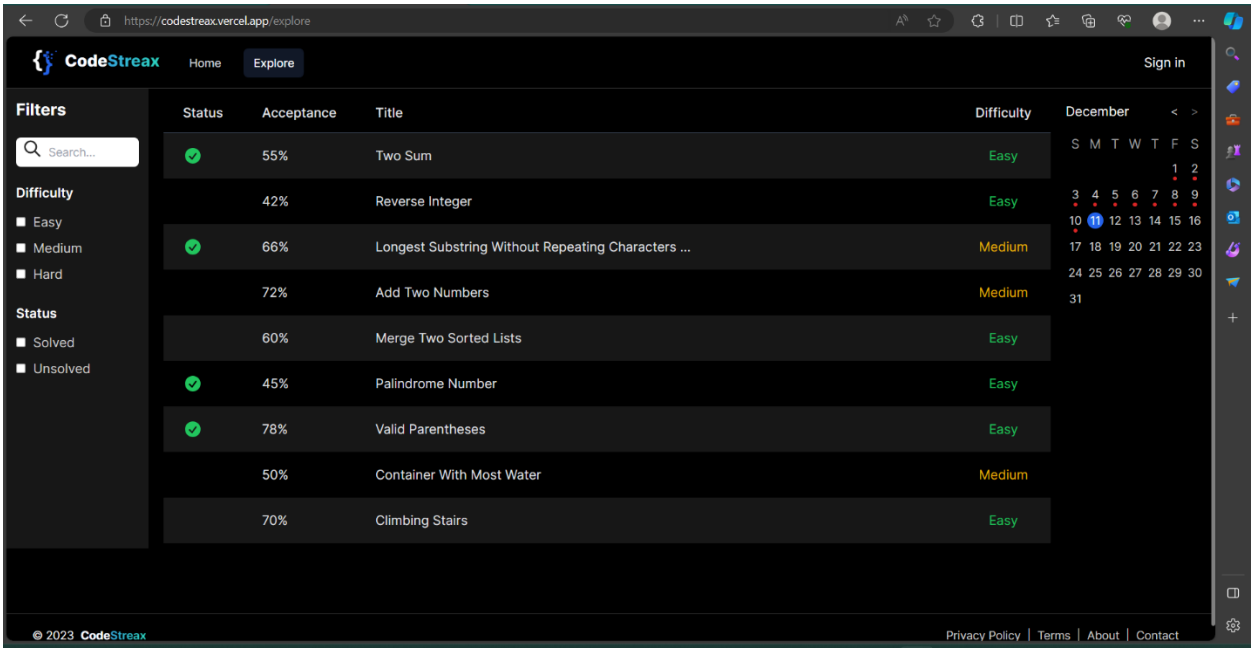

https://codestreax.vercel.app
Trusted by over 20,000 engineers all over the world
Our Engineers at Leading Companies Worldwide Collaborating Across the Globe to Shape the Future of Technology.
facebook
tinder
airbnb
HubSpot
amazon
" The challenging set of problems makes it a go-to resource for honing coding skills. The intuitive interface and diverse question bank make learning enjoyable. "
SDE-2 @Dropbox

Sign-In Page:

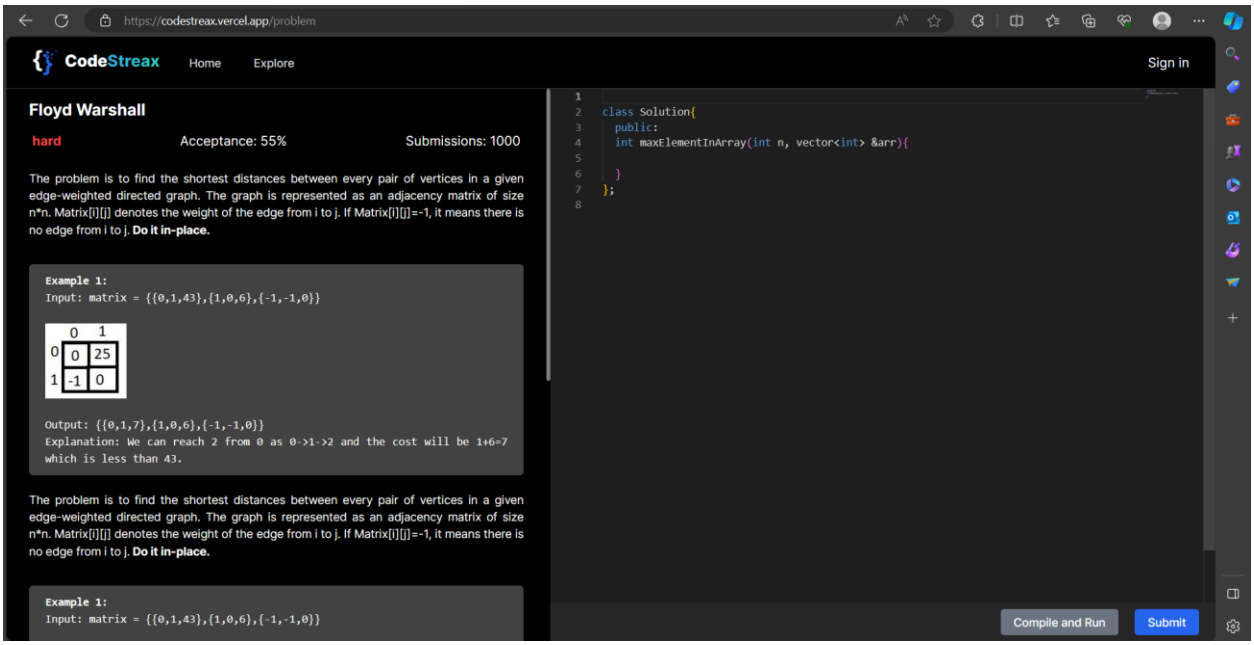


Explore Page:

Problem Statements list:



Problem Solving:



Code Submission: