A SYNOPSIS ON

SaaS – BASED ONLINE IDE

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

ROHIT TIWARI – URN: 2261494

ANISH KUMAR – URN: 2261094

ARNAV SINGH – URN: 2261113

HIMANSHU SINGH – URN: 2261273

Under the Guidance of

MR. ANUBHAV BEWERWAL

ASSISTANT PROFESSOR

Project Team ID: 118

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the Synopsis entitled **"SaaS – BASED ONLINE IDE"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering of the Graphic Era Hill University, Bhimtal campus and shall be carried out by the undersigned under the supervision of **Mr. Anubhav Bewerwal, Assistant Professor**, Department of Computer Science & Engineering, Graphic Era Hill University, Bhimtal.

• ROHIT TIWARI	2261494
• ANISH KUMAR	2261094
• ARNAV SINGH	2261113
• HIMANSHU SINGH	2261273

The above mentioned students shall be working under the supervision of the undersigned on the **"SaaS – BASED ONLINE IDE".**

Mr. Anubhav Bewerwal **Dr .Ankur Bisht(HOD)**

<u>Internal Evaluation (By DPRC Committee)</u>

Status of the Synopsis: Accepted / Rejected

Any Comments:

Name of Committee Members : **Signature With Date**

1.

2.

TABLE OF CONTENTS

CHAPTER – 1

Abstract:

The “SaaS-Based Online IDE with OS Concepts” is a modern cloud-based development environment designed to provide seamless coding experiences through the web. This platform allows users to write, execute, and manage code across multiple programming languages without the need to install any software on their local machines. Integrated with modern technologies such as Docker, Convex, Clerk, and hosted on Vercel, this IDE mimics functionalities of real operating systems by incorporating OS concepts like process and memory management, file handling, and user authentication.

The IDE supports features like snippet saving and sharing, real-time output display, execution history, and user profiles. To simulate isolated environments for code execution, Docker containers are leveraged—representing the concept of process isolation and memory management. Clerk handles user roles and permissions, replicating OS-level access control. API requests act like system calls, forming a strong bridge between user actions and server responses.

With the growing demand for remote and platform-independent development tools, this SaaS-based IDE offers a unique combination of operating system principles and cloud-native scalability. The system is ideal for students, developers, and educators looking for an efficient, secure, and user-friendly coding environment.

Introduction

In the modern era of cloud computing and remote development, traditional desktop-based IDEs are giving way to online IDEs that offer platform-independent access, team collaboration, and cloud-based execution environments. This project presents a SaaS (Software as a Service) based Online IDE that not only offers these modern conveniences but also incorporates fundamental Operating System (OS) concepts to deliver a robust and scalable solution.

The proposed system provides an all-in-one development environment with essential modules such as code editor, output panel, snippet sharing, execution history, user management, and a pricing/subscription model. What sets this IDE apart is its deliberate integration of OS functionalities like process and memory management using Docker, access control using Clerk, and system-level interactions through API system calls.

By utilizing Docker containers, each code execution is treated as a separate process—ensuring resource isolation and security. Memory allocation and limits replicate the real-world behavior of OS-level memory handling. Clerk ensures user-based access control, just like user roles in operating systems. File systems are simulated through databases where snippets and execution records are stored and retrieved.

Overall, this project not only addresses the need for an advanced cloud-based IDE but also acts as a practical implementation of how OS concepts work in modern software infrastructures. It bridges the academic knowledge of operating systems with real-world application development.

CHAPTER – 2

Objective:

The primary objective of this project is to develop a SaaS-based Online IDE that offers a complete coding experience within a web browser, integrating core Operating System principles such as process and memory management, user access control, and file handling. Specific goals include:

- Providing a browser-based code editor supporting multiple programming languages.
- Allowing users to execute code in isolated environments using Docker.
- Implementing user authentication, roles, and permissions via Clerk.
- Managing and storing code snippets and execution history.
- Offering subscription-based access models using Lemon Squeezy.
- Demonstrating how OS-level functionalities can be mirrored in a SaaS setup.

System Requirements:

Software Requirements:

- Operating System: Any (for developers), Ubuntu/Debian preferred on server
- Web Browser: Chrome, Firefox, Edge (latest versions)
- Development Environment: Visual Studio Code (optional)
- Node.js (v20+)
- Docker (latest version)
- Next.js 15
- TypeScript
- Convex (Backend logic and Database)
- Clerk (Authentication)
- Lemon Squeezy (Payment Integration)
- Vercel (Hosting)

Hardware Requirements:

- Server: Minimum 4 vCPUs, 8 GB RAM (for Docker execution)
- Client: Any device with internet access and modern browser
- Storage: At least 50GB for logs, code snippets, and user data

CHAPTER – 3

Technology Stack Used:

The proposed SaaS-based Online IDE system is developed using a modern full-stack architecture to ensure scalability, security, and seamless user experience. The following technologies are utilized in its development:

Component	Technology	Purpose
Frontend Framework –	Next.js 15	Used for building user interfaces with server-side rendering and routing.
Programming Language	TypeScript	Adds type safety and scalability to JavaScript-based development.
Backend & Database	Convex	Provides serverless functions and real-time data storage.
Authentication	Clerk	Manages user authentication, sessions, and access roles.
Code Execution Engine	Docker	Executes user code in isolated containers, simulating process and memory management.
Styling	Tailwind CSS, ShadCN UI	Used for modern and responsive UI design.
Hosting	Vercel	Continuous integration and deployment of the web application.
Payment Integration	Lemon Squeezy	Manages subscription plans and payment processing.
Version Control	Git & GitHub	Source code management and collaboration.

Modules of the Project:

1. Authentication Module
This module is responsible for secure user registration, login, and session handling. It incorporates role-based access control, utilizing **Clerk** to manage authentication and authorization processes effectively.

2. Home Page Module
The home page serves as the central hub for navigation. It provides users with access to key functionalities, language selection options, and a user-friendly interface that enhances overall accessibility.

3. Code Editor Module
This module delivers a feature-rich code editing environment. It supports syntax highlighting, auto-completion, and customizable themes based on popular Visual Studio Code (VSCode) aesthetics, offering an intuitive and modern coding experience.

4. Execution Module
The execution engine runs user-submitted code securely within Docker containers. This ensures process-level isolation, simulates an operating system environment, and manages memory efficiently to protect system integrity.

5. Output Module
Outputs from the execution process, including standard output and error messages, are displayed in real time. This module ensures immediate feedback for users, enhancing the interactivity and responsiveness of the platform.

6. Snippet Management Module
Users can save, retrieve, organize, and share their code snippets. This module mimics a file management system, allowing for structured storage and convenient access to frequently used code blocks.

7. User Profile Module
This component tracks user activity, including execution history, saved snippets, and personalized settings. It serves as a centralized space for users to manage their development preferences and records.

8. Pricing and Subscription Module
A tiered pricing system is implemented through **Lemon Squeezy**, offering both free and premium plans. Webhook integration ensures real-time subscription status updates, dynamically managing access to premium features.

CHAPTER – 4

OS Concepts Integration:

The following OS principles and concepts are utilized or simulated in the system architecture:

1. Process Management

- **Implementation**: Code execution is handled in isolated environments using Docker containers.
- **OS Relevance**: This mimics how modern operating systems manage processes in isolated memory spaces and allocate CPU resources accordingly.

2. Memory Management

- **Implementation**: Each code execution instance is allocated limited memory within its container.
- **OS Relevance**: This simulates how an OS allocates and restricts memory to running processes to avoid conflicts or overconsumption.

3. Multi-Programming & Concurrency

- **Implementation**: Multiple users can run programs simultaneously using serverless architecture.
- **OS Relevance**: This mimics concurrent execution where multiple processes are scheduled and run without interfering with each other.

4. User Management and Access Control

- **Implementation**: Clerk is used to manage users, roles, and sessions.
- **OS Relevance**: Simulates user authentication and access control as handled by operating systems.

5. File System Management

- **Implementation**: Users can save, update, and share code snippets.
- **OS Relevance**: Simulates file creation, storage, and sharing — key components of any file system.

6. Security and Isolation

- **Implementation**: Code runs inside secure containers, preventing interference and external access.
- **OS Relevance**: Aligns with OS-level process isolation and sandboxing for secure execution.

CHAPTER – 5

<u>Advantages</u>:

The Online SaaS IDE project offers several advantages in terms of both user experience and technical implementation. Key benefits include:

1. OS Concept Simulation

- Successfully integrates fundamental OS concepts like process management, memory allocation, and user authentication in a practical, web-based application.

2. Cross-Platform Accessibility

- As a SaaS application, it runs on any device with internet access, eliminating the need for local IDE installations or system-specific configurations.

3. Multi-language Code Execution

- Supports execution of programs in multiple programming languages, enhancing its versatility and usability for a wider audience.

4. User Isolation & Security

- Each code execution happens in an isolated container, ensuring user security and preventing cross-interference between users.

5. Code Sharing and Persistence

- Enables users to save, manage, and share code snippets, resembling a virtual file system environment.

6. Scalable and Modular Architecture

- Built using modern technologies like Next.js, Convex, and Clerk, ensuring the platform is scalable, maintainable, and performance-efficient.

Limitations:

Despite its strengths, the system also has a few limitations:

1. Not a Full Operating System

- While it implements OS principles, it does not replace or replicate a complete operating system kernel or architecture.

2. Limited Execution Power

- Code execution is sandboxed and limited in memory/time to ensure security and cost-efficiency, which restricts complex or resource-heavy programs.

3. Dependency on Internet

- Being an online platform, the system is dependent on a stable internet connection for functionality.

4. Language Support is Fixed

- The system currently supports a limited set of programming languages, which may restrict some users.

Conclusion:

The Online SaaS IDE project effectively bridges the gap between modern web-based software solutions and fundamental operating system concepts. Though not a full-fledged OS, the project successfully incorporates several OS principles, such as process management, memory allocation, user access control, and file handling, through the use of containerization, authentication services, and secure execution environments.

The system not only provides an intuitive and user-friendly interface for writing, executing, and sharing code in multiple programming languages but also demonstrates how OS functionalities can be replicated within a web application. This approach not only satisfies academic requirements but also showcases practical understanding and implementation of theoretical concepts.

In conclusion, the project stands as a valuable academic and technical contribution, demonstrating a real-world application of OS principles in a scalable and innovative SaaS environment.

REFERENCES

1. **Next.js Official Documentation**
   https://nextjs.org/docs

2. **Convex Developer Documentation**
   https://docs.convex.dev

3. **Clerk Authentication Documentation**
   https://clerk.dev/docs

4. **Docker: What is a Container?**
   https://www.docker.com/resources/what-container

5. **Operating System Concepts – Abraham Silberschatz, Peter B. Galvin, Greg Gagne**
   10th Edition, Wiley, 2018.

6. **MDN Web Docs – JavaScript and Web APIs**
   https://developer.mozilla.org

7. **GitHub Repository – Project Source Code**
   https://github.com/burakorkmez/online-compiler

8. **Udemy Course: Build and Deploy a Modern SaaS App**
   Author: Burak Orkmez, Udemy.com

9. **React.js Official Documentation**
   https://reactjs.org/docs/getting-started.html

10. **Node.js Official Documentation**
    https://nodejs.org/en/docs