CodeCampus - All In One Student Platform

Chapter 1 - Introduction

1.1 Background

In today's rapidly evolving digital landscape, the need for efficient, collaborative, and accessible development tools has become paramount. The software development industry has witnessed a significant shift towards remote work and distributed teams, necessitating platforms that can seamlessly integrate various aspects of the development process. This project aims to address these needs by introducing a comprehensive suite of tools designed to enhance productivity, foster collaboration, and streamline the learning process for developers of all skill levels.

The background of this project is rooted in the challenges faced by modern software developers, educators, and aspiring programmers. Traditional development environments often lack the flexibility and collaborative features required for today's distributed workforce. Moreover, the learning curve for new developers can be steep, with limited access to interactive, real-time coding experiences. Additionally, the job market for software developers is increasingly competitive, requiring candidates to not only possess strong coding skills but also to present their abilities effectively through well-crafted resumes.

The concept of online collaborative coding environments has gained traction in recent years, with platforms like GitHub Codespaces and Repl.it leading the way. However, there remains a gap in the market for a solution that combines a robust collaborative coding environment with educational resources and career development tools. This project seeks to fill that gap by offering a unified platform that caters to the diverse needs of the software development community.

The idea for this project was born out of the recognition that while there are numerous tools available for each aspect of a developer's journey – from learning to coding to job hunting – there is a lack of integration between these tools. This fragmentation often leads to inefficiencies and a disjointed experience for users. By bringing together a collaborative code editor, a LeetCode-style problem-solving platform, and a resume builder, this project aims to create a seamless ecosystem that supports developers throughout their career progression.

The collaborative code editor component of the project draws inspiration from the increasing need for real-time collaboration in software development. With the rise of remote work, especially accelerated by global events such as the COVID-19 pandemic, the ability to work on code simultaneously with team members across different locations has become crucial. This feature is designed to mimic the experience of pair programming in a virtual environment, allowing developers to share knowledge, troubleshoot issues, and innovate together regardless of their physical location.

The LeetCode clone aspect of the project addresses the growing emphasis on algorithmic problem-solving skills in technical interviews. Many leading tech companies use coding challenges as a key component of their hiring process, making platforms like LeetCode essential for job seekers in the tech industry. By incorporating a similar feature, this project aims to provide users with a familiar environment to hone their problem-solving skills and prepare for technical interviews.

Lastly, the resume builder component acknowledges the importance of effective self-presentation in the job market. A well-crafted resume can be the difference between landing an interview and being overlooked. However, many developers struggle to articulate their skills and experiences effectively on paper. By including a resume builder tailored specifically for tech professionals, this project aims to help users create compelling resumes that showcase their abilities and achievements in the best possible light.

The background of this project is thus a response to the evolving needs of the software development industry, aiming to create a comprehensive platform that supports developers from the learning phase through to job acquisition and beyond. By combining these three key components – collaborative coding, algorithmic problem-solving, and resume building – the project seeks to offer a unique and valuable resource for the global developer community.

1.2 Objectives

The primary objectives of this project are multifaceted, addressing various aspects of the software development lifecycle and career progression. These objectives have been carefully crafted to ensure that the platform provides comprehensive support for developers at different stages of their journey, from learning and skill development to job preparation and professional collaboration.

1. Facilitate Real-Time Collaboration: A key objective is to create a robust collaborative coding environment that allows multiple users to work on the same project simultaneously. This feature aims to replicate the benefits of in-person pair programming in a virtual setting, enabling developers to share knowledge, solve problems collectively, and improve code quality through real-time peer review.

2. Enhance Learning and Skill Development: By incorporating a LeetCode-style problem-solving platform, the project aims to provide a structured learning environment for developers to improve their algorithmic thinking and coding skills. The objective is to offer a wide range of data structure and algorithm problems, categorized by difficulty and topic, to cater to learners at various skill levels.

3. Streamline Project Setup and Management: The virtual space feature is designed to simplify the process of setting up development environments. The objective is to provide users with pre-configured environments for popular frameworks like React and Node.js, reducing the time and effort required to start new projects or onboard new team members.

4. Improve Code Accessibility and Sharing: By allowing users to easily share their virtual spaces, the platform aims to facilitate code reviews, mentoring sessions, and collaborative problem-solving. This objective addresses the need for seamless knowledge transfer within development teams and the wider developer community.

5. Enhance Interview Preparation: Through the LeetCode clone component, the project aims to help users prepare effectively for technical interviews. The objective is to provide a realistic coding environment that mimics interview conditions, allowing users to practice under time constraints and receive immediate feedback on their solutions.

6. Accelerate Career Development: The resume builder feature is aimed at helping developers create professional, tailored resumes that effectively showcase their skills and experiences. The objective is to provide templates and guidance specific to the tech industry, helping users stand out in competitive job markets.

7. Promote Continuous Learning: By integrating coding challenges with the collaborative environment, the project aims to encourage continuous learning and skill improvement. The objective is to create a platform where users can easily switch between working on personal projects and solving coding problems, fostering a habit of regular practice and learning.

8. Improve Code Quality and Best Practices: Through features like real-time syntax highlighting, error checking, and integrated terminal access, the platform aims to promote best coding practices and help users write cleaner, more efficient code.

9. Enhance Productivity and Workflow Efficiency: By combining multiple tools into a single platform, the project aims to streamline the developer workflow, reducing the need to switch between different applications for coding, learning, and career development.

10. Provide a Comprehensive Development Environment: The objective is to offer a full-featured development environment that includes file management, code editing, preview capabilities, and terminal access, all within a web browser. This aims to reduce setup time and provide a consistent experience across different devices and operating systems.

1.3 Purpose

The overarching purpose of this project is to create an integrated, all-encompassing platform that addresses the diverse needs of software developers throughout their career journey. This platform aims to serve as a one-stop solution for coding, learning, collaboration, and career development, effectively bridging the gaps between various aspects of a developer's professional life.

At its core, the purpose of this project is to empower developers by providing them with the tools and resources they need to excel in their craft, collaborate effectively with peers, and advance their careers. By combining a collaborative code editor, a LeetCode-style problem-solving platform, and a resume builder, the project seeks to create a seamless ecosystem that supports developers at every stage of their professional development.

One of the primary purposes of the collaborative code editor is to foster a culture of teamwork and knowledge sharing within the developer community. In an increasingly globalized and remote work environment, the ability to collaborate effectively on code is paramount. This feature aims to break down geographical barriers, allowing developers from different parts of the world to work together in real-time, share ideas, and learn from each other. It serves to recreate the dynamic and interactive nature of in-person pair programming in a virtual setting, promoting code quality, rapid problem-solving, and continuous learning.

The virtual space concept within the collaborative editor serves a crucial purpose in simplifying the development process. By providing pre-configured environments for popular frameworks like React and Node.js, it aims to eliminate the often time-consuming and frustrating process of setting up development environments. This not only increases productivity but also lowers the barrier to entry for newcomers to web development, allowing them to focus on coding rather than configuration.

The LeetCode clone component of the project serves multiple important purposes. Firstly, it aims to provide a structured learning environment where developers can systematically improve their problem-solving skills and deepen their understanding of data structures and algorithms. This is crucial not only for personal growth but also for success in technical interviews. Secondly, it serves as a platform for continuous practice and improvement, helping developers stay sharp and up-to-date with their coding skills. Lastly, it aims to build confidence in developers by providing immediate feedback and allowing them to track their progress over time.

The resume builder feature serves the critical purpose of helping developers present their skills and experiences effectively to potential employers. In the competitive tech job market, a well-crafted resume can be the difference between landing an interview and being overlooked. By providing tailored templates and guidance specific to the tech industry, this feature aims to help developers create resumes that truly showcase their abilities and achievements, thereby improving their chances of career advancement.

Another key purpose of this project is to create a sense of community among developers. By providing a platform where users can share their projects, collaborate on code, and discuss problem-solving approaches, the project aims to foster a supportive environment where knowledge sharing and mutual growth are encouraged. This sense of community can be particularly valuable for newcomers to the field, providing them with mentorship opportunities and a support network as they navigate their early career stages.

The project also serves the purpose of promoting best practices in software development. Through features like real-time syntax highlighting, error checking, and integrated testing in the LeetCode clone, it aims to help developers write cleaner, more efficient, and more maintainable code. This not only benefits individual developers but also contributes to raising the overall standard of code quality in the industry.

Furthermore, the project aims to address the growing need for lifelong learning in the rapidly evolving field of software development. By integrating learning resources (in the form of coding challenges) with practical development tools, it encourages a culture of continuous improvement and adaptation to new technologies and methodologies.

From an educational standpoint, the platform serves the purpose of providing a realistic and interactive learning environment for students and aspiring developers. It bridges the gap between theoretical knowledge and practical application, allowing users to immediately apply what they've learned in a real coding environment.

Lastly, the project serves the broader purpose of democratizing access to high-quality development tools and resources. By providing these features in a web-based platform, it aims to make advanced development capabilities accessible to users regardless of their hardware limitations or geographical location, as long as they have internet access.

In summary, the purpose of this project is to create a comprehensive, accessible, and user-friendly platform that empowers developers to learn, collaborate, and advance their careers more effectively. By addressing multiple aspects of a developer's journey within a single ecosystem, it aims to streamline the development process, foster community engagement, and ultimately contribute to the growth and success of individual developers and the software development industry as a whole.

1.4 Scope

The scope of this project encompasses a wide range of functionalities and features designed to create a comprehensive platform for software developers. It aims to cover various aspects of the development lifecycle, from learning and practice to collaboration and career advancement. The project is ambitious in its breadth, but also clearly defined in its boundaries to ensure feasibility and effective implementation.

Collaborative Code Editor:
The core of the project is the collaborative code editor, which forms the foundation of the virtual space concept. The scope of this component includes:

1. Real-time collaboration features allowing multiple users to edit code simultaneously.
2. Support for creating and managing virtual spaces, each representing a distinct project or coding environment.
3. Integration of a file structure system within each virtual space for organizing project files and directories.

4. A code editor with syntax highlighting, auto-completion, and error detection for supported languages.
5. A browser-like view for previewing output, particularly useful for web development projects.
6. An integrated terminal for executing commands, installing dependencies, and managing the development environment.
7. Initial support for React and Node.js projects, with the potential for future expansion to other frameworks and languages.
8. Sharing capabilities, allowing users to invite collaborators via email and set appropriate access levels.
9. Basic version control features to track changes and manage code versions.

LeetCode Clone:
The LeetCode-style problem-solving platform is another key component of the project. Its scope includes:

1. A curated collection of data structure and algorithm problems, categorized by difficulty and topic.
2. A split-screen interface with problem description on one side and code editor on the other.
3. An execution environment to run user-submitted code against predefined test cases.
4. Automated evaluation of solutions, providing immediate feedback on correctness and efficiency.
5. Progress tracking features to allow users to monitor their improvement over time.
6. A system for users to save their solutions and revisit them later.

Resume Builder:
The resume builder component, while more limited in technical complexity, is crucial for the career development aspect of the platform. Its scope includes:

1. A selection of resume templates tailored for tech industry professionals.
2. An intuitive interface for users to input their personal information, skills, experiences, and achievements.
3. Real-time preview of the resume as users build it.
4. Export options in various formats (e.g., PDF, Word) for the completed resume.
5. Basic customization options for fonts, colors, and layout.
6. Tips and guidance on effective resume writing for tech positions.

Integration and Platform Features:

To tie these components together into a cohesive platform, the scope also includes:

1. A unified user account system for accessing all features of the platform.
2. A dashboard interface for users to navigate between different components (code editor, problem-solving, resume builder).
3. Basic social features to foster community interaction, such as the ability to view public projects or solutions.
4. A notification system to alert users of collaboration invites, comments, or other relevant activities.

Security and Data Management:
Ensuring the security and privacy of user data is critical. The scope includes:

1. Secure user authentication and authorization systems.
2. Encryption of sensitive data, both in transit and at rest.
3. Regular backups of user projects and data.
4. Compliance with relevant data protection regulations.

Performance and Scalability:
To ensure a smooth user experience, the scope includes:

1. Optimization of the collaborative editing features to handle multiple simultaneous users.
2. Efficient data synchronization mechanisms to maintain consistency across collaborating users.
3. Scalable architecture to handle growing numbers of users and projects.

User Support and Documentation:
To aid users in effectively utilizing the platform, the scope includes:

1. Comprehensive documentation covering all features and functionalities.
2. In-app tutorials or walkthroughs for new users.
3. A help center or FAQ section to address common questions and issues.

While ambitious, the scope of this project is designed to create a cohesive and valuable platform for developers. It's important to note that while the initial release may not include all features at their full extent, the architecture will be designed to allow for future expansions and improvements.

Out of Scope:

To maintain focus and feasibility, certain elements are considered out of scope for the initial project:

1. Advanced project management features (these may be considered for future iterations).
2. Integration with external version control systems like GitHub (though this could be a valuable future addition).
3. Mobile app development (the initial focus is on a web-based platform).
4. Advanced AI-powered code generation or problem-solving assistance.
5. Extensive learning resources beyond coding problems (e.g., video tutorials, courses).

By clearly defining what is in and out of scope, the project aims to deliver a robust, useful platform that meets the core needs of developers while leaving room for future growth and expansion.

Chapter 2: Survey of Technologies

1. Frontend Technologies

Next.js – A React-based framework that helps in building fast and SEO-friendly websites. It supports server-side rendering and static site generation, making it perfect for performance optimization.

Tailwind CSS – A utility-first CSS framework that helps in quickly designing modern and responsive UI components without writing much custom CSS.

2. Backend Technologies

Node.js – A runtime environment that allows JavaScript to be used on the server side. It is lightweight and efficient for handling multiple requests at once.

Express.js – A web framework for Node.js that simplifies backend development by handling routes, middleware, and API requests efficiently.

3. Real-time Collaboration

WebSockets (Socket.io) – A technology that enables real-time, bi-directional communication between the server and clients, allowing multiple users to collaborate in the code editor simultaneously.

4. AI Integration

Google Gemini AI (3.5 Turbo AI model) – A powerful AI model used for intelligent suggestions, resume improvement, code analysis, and DSA problem-solving hints.

DigitalOcean/AWS – Cloud providers that can host backend services, databases, and real-time collaboration servers.

5. Version Control & Collaboration

Git & GitHub – Essential for managing code, tracking changes, and collaborating with multiple developers.

Chapter 3: System Analysis

2.1 Existing System

In the current technological landscape, there are various standalone solutions that address specific aspects of coding, problem-solving, and professional development. However, these solutions often lack integration and fail to provide a comprehensive platform for users to enhance their skills across multiple domains.

For collaborative coding, existing systems like GitHub and GitLab offer version control and basic collaboration features. However, they lack real-time editing capabilities and integrated development environments (IDEs) within the browser. This limitation often requires users to set up local development environments, which can be time-consuming and may lead to inconsistencies across team members' setups.

In the realm of coding practice and algorithm challenges, platforms like LeetCode and HackerRank exist. These platforms provide a wide array of problems but often lack a seamless integration with real-world project development environments. Users typically solve problems in isolation, without the context of how these algorithms might be applied in larger software projects.

For resume building, there are numerous online tools and templates available. However, these tools often operate in isolation from the user's actual coding experiences and achievements. This disconnect can result in resumes that fail to accurately represent a candidate's true capabilities and project involvement.

The existing systems, while valuable in their respective domains, fall short in providing a unified experience that combines collaborative coding, algorithmic problem-solving, and professional development. Users are required to juggle multiple platforms, leading to a fragmented learning and development experience. This fragmentation can hinder the efficient acquisition and application of skills, as well as the ability to showcase one's capabilities comprehensively.

Moreover, the current systems often lack personalization and adaptive learning features. They typically offer a one-size-fits-all approach, which may not cater to the diverse learning styles and skill levels of users. This can result in suboptimal learning experiences and slower skill progression for many individuals.

Another limitation of existing systems is the absence of a seamless transition between learning and application. While users can practice coding on one platform and build projects on another, there's often a disconnect between these activities. This gap can make it challenging for learners to see how the algorithms and data structures they're studying apply to real-world software development scenarios.

Additionally, current platforms often lack robust feedback mechanisms. While some may offer basic test case validation, they frequently fall short in providing detailed, context-aware feedback that can help users improve their coding style, efficiency, and best practices adherence.

In terms of collaboration, existing systems often lack fine-grained access control and real-time communication features within the coding environment. This can lead to challenges in managing larger projects with multiple contributors, especially in educational or training contexts where different levels of access may be required for different users.

2.2 Proposed System

The proposed system aims to address the limitations of existing solutions by providing an integrated platform that combines collaborative coding, algorithm practice, and professional development tools. This comprehensive system is designed to create a seamless experience for users, enabling them to develop their skills, work on projects, and showcase their abilities all within a single, cohesive environment.

At the core of the proposed system is the Collaborative Code Editor module. This feature-rich environment allows users to create virtual spaces where they can work on React or Node.js projects. The virtual space is more than just a code repository; it's a fully-fledged development environment accessible through a web browser. This eliminates the need for complex local setups and ensures consistency across all team members.

The Collaborative Code Editor includes a robust file structure system, allowing users to organize their projects efficiently. The integrated code editor supports syntax highlighting, auto-completion, and other modern IDE features, making the coding experience smooth and productive. A unique aspect of this module is the browser-like view that renders the output in real-time, enabling developers to see the results of their code immediately. This feature is particularly valuable for front-end development, where visual feedback is crucial.

To further enhance the development experience, the system includes a terminal interface within the virtual space. This terminal allows users to execute commands, install dependencies, and perform other command-line operations directly in the browser. This feature ensures that all project dependencies and configurations are managed within the shared environment, eliminating inconsistencies that often arise when developers use different local setups.

The collaboration aspect of the Code Editor is a standout feature. Users can invite team members by adding their email addresses, granting them access to the shared project space. Real-time collaboration is supported, allowing multiple users to work on the same codebase simultaneously. This feature is enhanced with presence awareness, showing which team members are currently active and which files they are working on. To facilitate communication, the system includes an integrated chat feature, allowing developers to discuss changes, share ideas, and solve problems without leaving the coding environment.

The second major component of the proposed system is the LeetCode clone, which serves as a comprehensive platform for practicing data structures and algorithms. This module presents users with a curated list of problems, carefully categorized by topic and difficulty level. The interface is designed to mimic real-world coding scenarios, with a split-screen layout that displays the problem description, examples, and constraints on one side, and a fully-functional code editor on the other.

What sets this LeetCode clone apart is its integration with the Collaborative Code Editor. Users can seamlessly transition from solving algorithm problems to applying those solutions in their projects. This integration helps bridge the gap between theoretical knowledge and practical application, a common challenge in software development education.

The problem-solving environment is equipped with a robust execution engine that runs submitted code against a comprehensive set of test cases. These test cases are designed to cover edge cases and performance considerations, ensuring that solutions are not only correct but also efficient. Upon submission, users receive detailed feedback, including runtime analysis, memory usage statistics, and suggestions for optimization where applicable.

To enhance the learning experience, the system incorporates an intelligent recommendation engine. This engine analyzes a user's performance and problem-solving patterns to suggest relevant problems that will help improve specific skills or address weaknesses. Additionally, the platform includes a discussion forum for each problem, allowing users to share insights, discuss alternative solutions, and learn from peers.

The third component of the proposed system is the Resume Builder. This module is designed to seamlessly integrate with the user's activities on the platform, automatically populating the resume with relevant projects, solved problems, and acquired skills. The Resume Builder offers a variety of templates and customization options, allowing users to create professional, tailored resumes that accurately reflect their capabilities.

What makes this Resume Builder unique is its dynamic nature. As users complete projects in the Collaborative Code Editor or solve problems in the LeetCode clone, their resume is automatically updated to reflect these achievements. This ensures that the resume always presents an up-to-date and accurate representation of the user's skills and experiences.

The Resume Builder also includes an AI-powered suggestion system that provides recommendations for improving the resume's content and structure based on industry standards and best practices. It can highlight key skills that are in high demand in the job market, suggest ways to better describe project contributions, and even recommend additional skills to acquire based on the user's career goals.

By integrating these three modules - Collaborative Code Editor, LeetCode clone, and Resume Builder - the proposed system creates a comprehensive ecosystem for coding education, practice, and professional development. This integrated approach addresses the fragmentation issues present in existing systems and provides a more holistic platform for users to grow their skills, collaborate on projects, and showcase their abilities to potential employers.

2.3 Requirement Analysis

The requirement analysis for the proposed system encompasses a wide range of functional and non-functional requirements necessary to deliver a comprehensive, user-friendly, and efficient platform. This analysis is crucial for ensuring that the system meets the needs of its users while maintaining high standards of performance, security, and scalability.

Functional Requirements:

1. User Authentication and Authorization:

   - The system must support secure user registration and login processes.
   - Implementation of role-based access control for different types of users (e.g., administrators, project owners, collaborators).
   - Integration with OAuth providers for simplified login options.

2. Collaborative Code Editor:
   - Creation and management of virtual spaces for projects.
   - Support for React and Node.js project development environments.
   - Real-time collaborative editing with conflict resolution mechanisms.
   - File structure management system with create, read, update, and delete (CRUD) operations.
   - Integrated terminal for command execution and dependency management.
   - Live preview functionality for immediate visual feedback of code changes.

3. Project Sharing and Collaboration:
   - Ability to invite users to projects via email.
   - Real-time presence indicators showing active users and their current files.

4. LeetCode Clone:
   - Comprehensive problem database with categorization by topic and difficulty.
   - Split-screen interface with problem description and code editor.
   - Code execution engine supporting multiple programming languages.
   - Robust test case system for validating submitted solutions.
   - Progress tracking system to monitor user advancement through problems.

5. Resume Builder:
   - Dynamic resume generation based on user activities within the platform.
   - Customizable resume templates with various design options.
   - Automatic updates to reflect completed projects and solved problems.
   - AI-powered content suggestions for resume improvement.
   - Export functionality in multiple formats (PDF, DOCX, HTML).

Non-Functional Requirements:

1. Performance:
   - The system should support real-time collaboration with minimal latency (< 100ms).
   - Code execution for the LeetCode clone should complete within a reasonable timeframe (< 5 seconds for most problems).

- The platform should be able to handle a high number of concurrent users (at least 10,000) without significant performance degradation.

2. Scalability:
   - The architecture should support horizontal scaling to accommodate growing user bases.
   - Database design should optimize for read and write operations at scale.
   - Implement caching mechanisms to reduce database load and improve response times.

4. Reliability:
   - The system should have an uptime of at least 99.9%.
   - Implement robust error handling and logging mechanisms.
   - Regular backups of user data and project information.

5. Usability:
   - The user interface should be intuitive and responsive across different devices and screen sizes.
   - Accessibility features should be implemented to support users with disabilities.
   - The system should provide clear documentation and help resources for users.

6. Compatibility:
   - The platform should be compatible with major web browsers (Chrome, Firefox, Safari, Edge).
   - The code editor should support syntax highlighting for multiple programming languages.

7. Data Management:
   - Efficient storage and retrieval of large codebases and project structures.
   - Implement data retention policies in compliance with relevant regulations.

8. Integration:
   - API endpoints for potential future integrations with external tools and services.
   - Support for popular version control systems (e.g., Git) integration.

9. Monitoring and Analytics:
   - Implement comprehensive logging and monitoring solutions.
   - Analytics tools to track user engagement, problem-solving patterns, and system usage.

10. Compliance:
   - Ensure compliance with data protection regulations (e.g., GDPR, CCPA) as applicable.
   - Implement necessary measures for intellectual property protection.

This comprehensive requirement analysis provides a solid foundation for the development of the proposed system. It addresses the core functionalities needed for each module while also considering crucial non-functional aspects that will ensure the platform's success in terms of performance, security, and user satisfaction. As the project progresses, these requirements may be further refined and expanded based on user feedback and evolving technological capabilities.

2.4 Hardware Requirements

The hardware requirements for the proposed system are crucial to ensure smooth operation, scalability, and optimal performance.
Client-Side Requirements:

While the system is primarily web-based, it's important to consider the minimum hardware requirements for end-users to ensure a smooth experience:

1. Computer/Laptop:
   - Modern multi-core processor (Intel i5/AMD Ryzen 5 or better).
   - Minimum 8GB RAM, recommended 16GB for optimal performance.
   - Stable internet connection with at least 10 Mbps download and 5 Mbps upload speeds.

2. Mobile Devices:
   - Recent smartphone or tablet models (within the last 3-4 years).
   - Minimum 3GB RAM for smooth operation of the mobile version.

The hardware requirements outlined above are designed to support a robust, scalable, and high-performance system capable of handling the diverse needs of the collaborative coding platform, algorithm practice environment, and resume builder. These specifications provide a solid foundation for the system's infrastructure, ensuring it can deliver a seamless user experience while maintaining the capacity for future growth and expansion.

As the user base grows and feature set expands, it's crucial to regularly review and update the hardware infrastructure to meet evolving demands. This may involve upgrading existing hardware, adding new servers, or transitioning certain components to cloud-based solutions for improved scalability and cost-effectiveness. Regular performance monitoring and capacity planning will be essential to maintain the system's efficiency and reliability over time.

2.5 Software Requirements

The software requirements for the proposed system are crucial for ensuring the platform's functionality, performance, and user experience. Given the complex nature of the system, which integrates collaborative coding, algorithm practice, and resume building, a carefully selected stack of software components is necessary. Here's a comprehensive breakdown of the software requirements:

Operating System:

1. Server-Side:
   - Linux-based OS (e.g., Ubuntu Server, CentOS) for its stability, security, and performance in server environments.
   - Consideration for containerization using Docker for easier deployment and scaling.

2. Client-Side:
   - The system should be accessible from any modern operating system (Windows, macOS, Linux) via web browsers.

Web Server:

1. Nginx or Apache:
   - For serving static content and acting as a reverse proxy.
   - Chosen for their performance, reliability, and extensive community support.

2. Node.js:
   - For running the server-side

Chapter 4: System Design

3.1 Module Division

The system is divided into three primary modules, each serving a distinct purpose while integrating seamlessly to provide a comprehensive platform for coding, learning, and professional development. The module division is as follows:

1. Collaborative Code Editor Module:
This module forms the core of the development environment, allowing users to create, manage, and collaborate on coding projects in real-time. Key components include:

a) Virtual Space Management:
- Creation and configuration of virtual spaces for projects
- Project template selection (React or Node.js)

b) Collaboration Tools:
- Real-time code synchronization across multiple users
- User invitation system via email
- Access control and permission management
- Presence awareness indicators

c) Development Environment:
- File structure viewer and manager
- Live preview/output window for immediate visual feedback
- Integrated terminal for command execution and dependency management

e) Project Sharing and Export:
- Functionality to share projects or specific files
- Export options for local development or deployment

2. LeetCode Clone Module:
This module provides a platform for users to practice and improve their algorithmic problem-solving skills. Components include:

a) Problem Database:
- Categorized collection of data structure and algorithm problems
- Difficulty level classification

- Tags for topics and concepts covered

b) User Interface:
- Split-screen layout with problem description and code editor
- Language selection for coding
- Input/Output examples display

c) Code Execution Engine:
- Sandboxed environment for secure code execution
- Test case runner with performance metrics

3. Resume Builder Module:
This module enables users to create professional resumes that showcase their coding skills and projects. Components include:

a) Resume Editor:
- Template selection and customization
- Section management (education, experience, skills, projects)
- Real-time preview of the resume

c) AI-Powered Suggestions:
- Content recommendations for effective resume writing
- Keyword optimization for ATS compatibility
- Style and formatting suggestions

d) Export and Sharing:
- Multiple format export options (PDF)
- Direct sharing to professional networking platforms

Integration Layer:
While not a separate module, an integration layer ensures seamless communication and data flow between the three primary modules. This layer includes:

- Unified user authentication and authorization system
- Centralized user profile management
- Cross-module activity tracking and analytics
- Shared notification system
- Consistent UI/UX elements across modules

By dividing the system into these modules, we ensure a clear separation of concerns while allowing for integrated functionality. Each module can be developed, tested, and scaled independently, yet they work together to provide a cohesive user experience. The modular approach also facilitates future expansions and integrations, making the system adaptable to evolving user needs and technological advancements.

3.2 Data Dictionary

A comprehensive data dictionary is crucial for understanding the structure and relationships of data within our system. Here's a detailed breakdown for each module:

1. Collaborative Code Editor Module:

a) User Table:
- UserID (Primary Key): Unique identifier for each user
- Email: User's email address (used for authentication and notifications)
- Password: Hashed password for security

b) VirtualSpace Table:
- SpaceID (Primary Key): Unique identifier for each virtual space
- OwnerID (Foreign Key): References UserID of the space creator
- Name: Name of the virtual space
- CreationDate: Timestamp of space creation
- ProjectType: Enum (React, Node.js)
- StoragePath: Path to project files in the storage system
- IsPublic: Boolean indicating if the space is publicly accessible

c) SharedAccess Table:
- AccessID (Primary Key): Unique identifier for each access grant
- SpaceID (Foreign Key): References SpaceID of the shared space
- UserID (Foreign Key): References UserID of the user granted access
- InvitationDate: Timestamp of when access was granted

2. LeetCode Clone Module:

a) Question Table:
- QuestionID (Primary Key): Unique identifier for each question
- Title: Title of the problem
- Description: Detailed problem statement
- Difficulty: Enum (Easy, Medium, Hard)

- Category: Primary category of the problem (e.g., Arrays, Trees)
- Tags: Array of tags associated with the problem

b) TestCase Table:
- TestCaseID (Primary Key): Unique identifier for each test case
- QuestionID (Foreign Key): References QuestionID of the associated question
- Input: Input data for the test case
- ExpectedOutput: Expected output for the given input
- IsHidden: Boolean indicating if this is a hidden test case

3. Resume Builder Module:

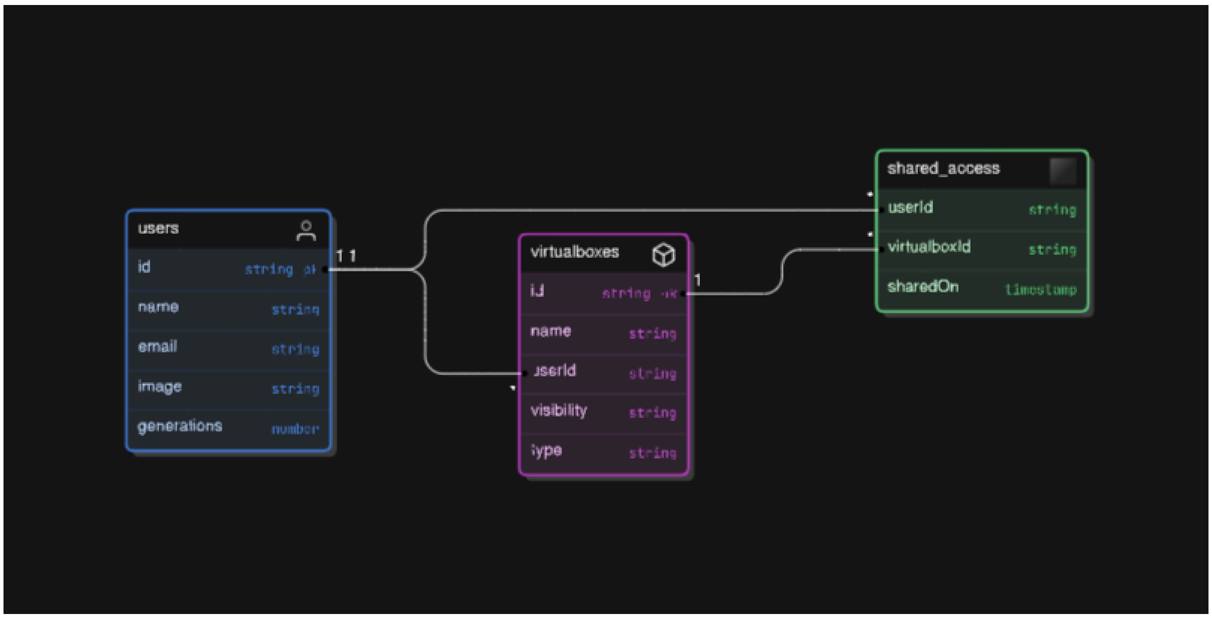
a) Resume Table:
- ResumeID (Primary Key): Unique identifier for each resume
- UserID (Foreign Key): References UserID of the resume owner
- Title: Title of the resume
- CreationDate: Date the resume was created
- LastModified: Date of last modification
- TemplateID (Foreign Key): References TemplateID of the used template
- Content: JSON object containing structured resume data
- IsPublic: Boolean indicating if the resume is publicly viewable

This data dictionary provides a comprehensive overview of the main entities and their attributes within our system. It serves as a foundation for database design and ensures consistency across different parts of the application. The relationships between these entities (e.g., one-to-many, many-to-many) would be implemented through appropriate foreign key constraints and junction tables where necessary.

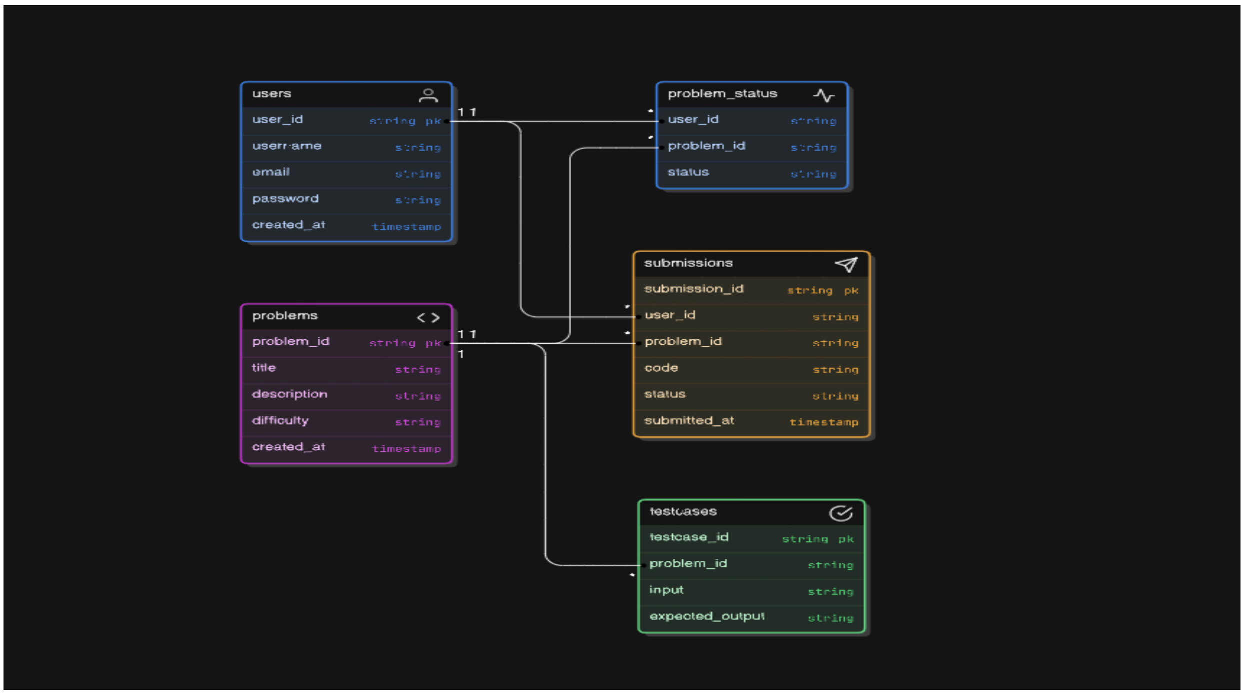
3.3 ER Diagrams

Entity-Relationship (ER) diagrams are crucial for visualizing the structure and relationships of data within our system. For each module, we'll create a detailed ER diagram that illustrates the entities, their attributes, and the relationships between them.

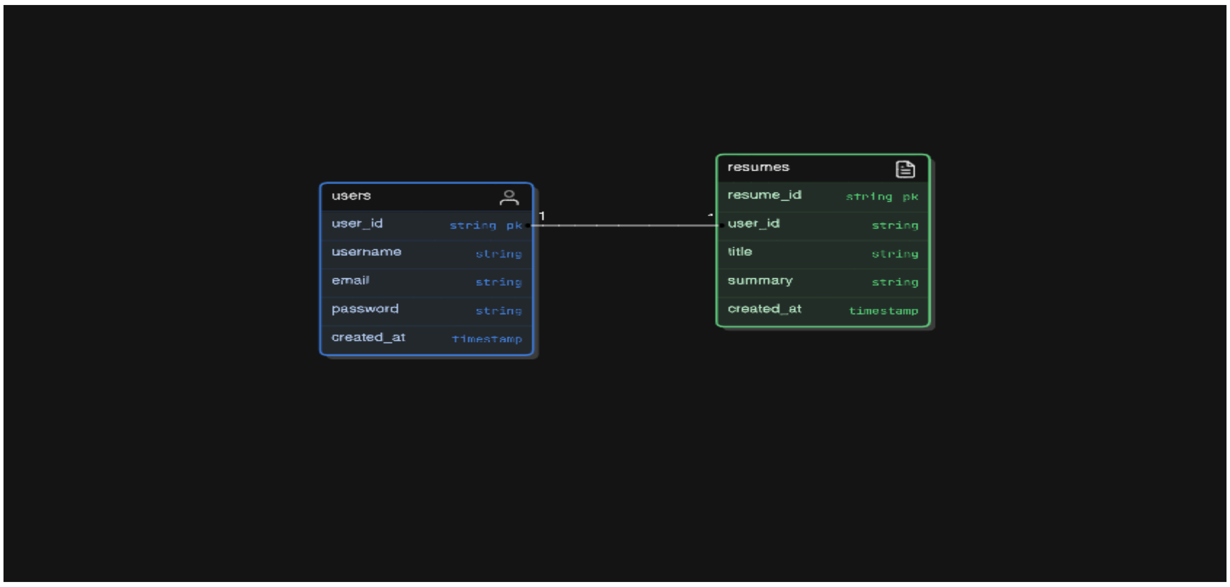
1. Collaborative Code Editor Module ER Diagram:



2. LeetCode Clone Module ER Diagram:



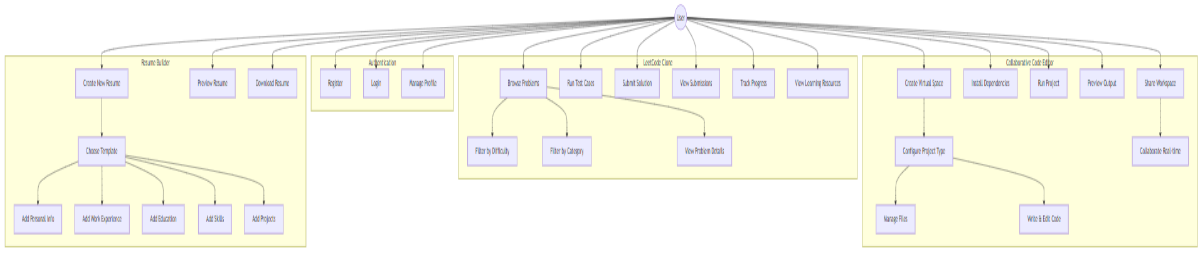
3. Resume Builder Module ER Diagram:



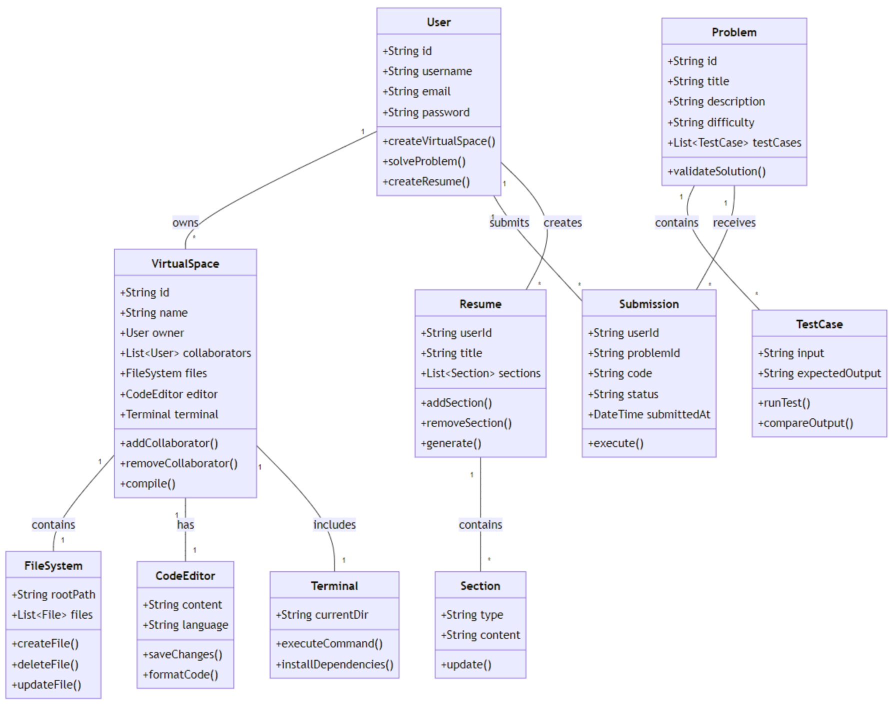
3.4 DFD/UML Diagrams

Data Flow Diagrams (DFD) and Unified Modeling Language (UML) diagrams are essential tools for visualizing the system's processes, data flows, and interactions. We'll create high-level diagrams for each module and an overall system diagram to illustrate how the components interact.

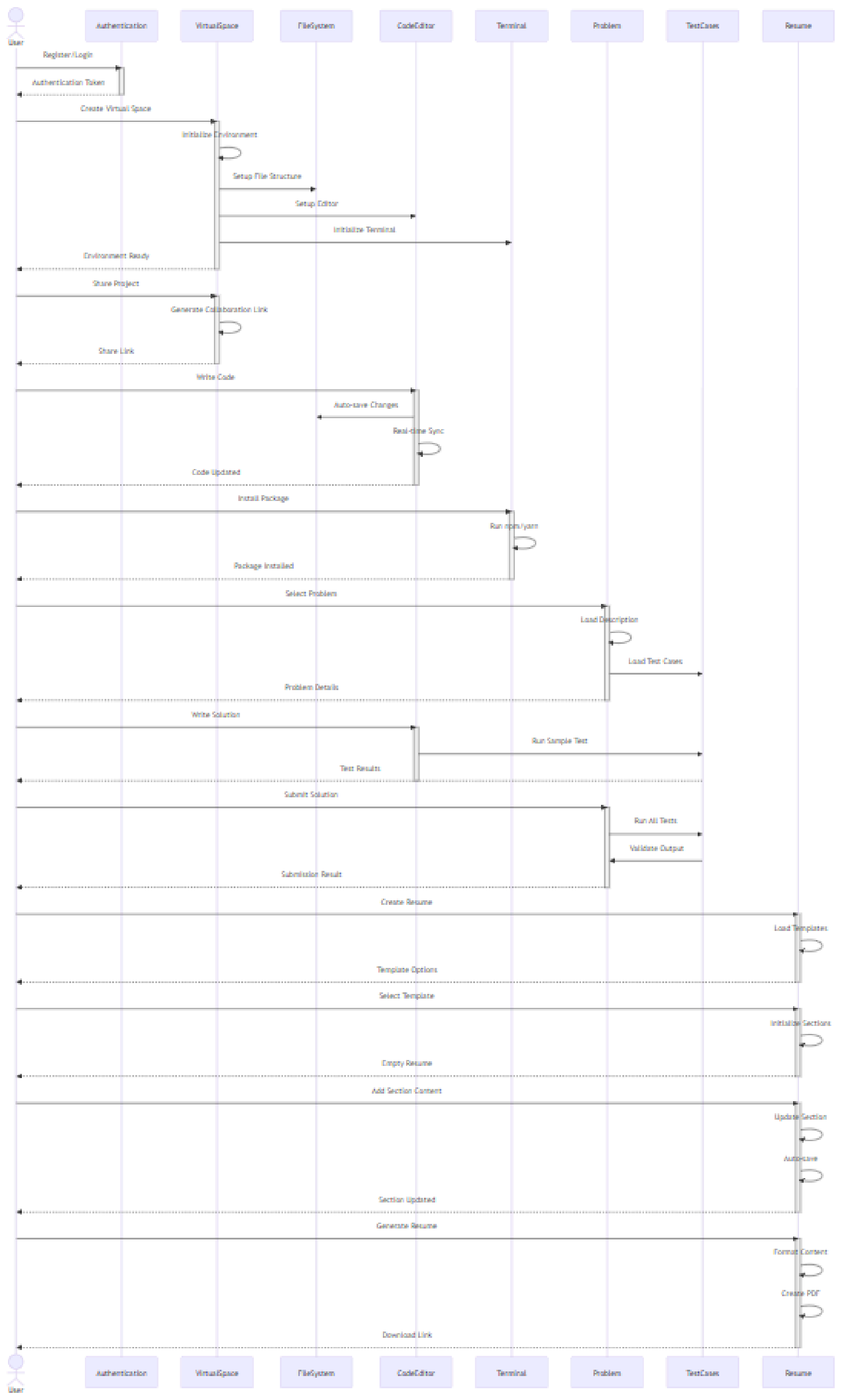
1.1. Collaborative Code Editor Module Class Diagram:



1.2. Collaborative Code Editor Module Class Diagram:



1.3 Collaborative Code Editor Module State Diagram



Chapter 5 - Implementation & Testing Approach

This chapter provides insights into the implementation and testing of the three core modules of the project

Implementation

1. Resume Builder

- Developed an ATS-friendly resume generator with customizable templates.
- Implemented a user-friendly interface for seamless resume creation and PDF export.

2. Collaborative Code Editor

- Built a real-time code editor using WebSockets for live collaboration.
- Integrated React & Node.js preview panel for instant output.
- Enabled multiple users to join and work on the same workspace.

3. DSA Problem-Solving Platform

- Designed an interactive platform with a rich set of DSA questions.
- Implemented a code execution engine to evaluate solutions.
- Provided performance insights and progress tracking for users.

4. Tech Stack & Infrastructure

- **Frontend:** Next.js, TailwindCSS for a responsive and fast UI.
- **Backend:** Node.js, Express.js with TypeScript for scalable API handling.
- **AI Integration:** Google Gemini AI for smart code suggestions.
- **Deployment & Security:** Cloudflare for DNS & security enhancements.

5.1 Coding Approach

```
import React, { useState, useEffect } from 'react';
import AceEditor from 'react-ace';
import 'ace-builds/src-noconflict/mode-javascript';
import 'ace-builds/src-noconflict/theme-github';

const CodeEditor = ({ initialCode, onCodeChange, onSave }) => {
  const [code, setCode] = useState(initialCode);

  useEffect(() => {
    setCode(initialCode);
  }, [initialCode]);

  const handleChange = (newValue) => {
    setCode(newValue);
    onCodeChange(newValue);
  };

  return (
    <div>
      <AceEditor
        mode="javascript"
        theme="github"
        name="code-editor"
        value={code}
        onChange={handleChange}
        editorProps={{ $blockScrolling: true }}
        setOptions={{
          useWorker: false,
        }}
      />
      <button onClick={() => onSave(code)}>Save</button>
    </div>
  );
};

export default CodeEditor;
```

5.2 Testing Approach

The testing approach for this project encompasses both unit testing and integration testing to ensure the reliability and functionality of the components.

5.3 Unit Testing

```
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import CodeEditor from './CodeEditor';

describe('CodeEditor Component', () => {
  it('should update code state on change', () => {
    const onCodeChange = jest.fn();
    const { getByRole } = render(<CodeEditor initialCode="initial code"
onCodeChange={onCodeChange} onSave={() => {}} />);

    const editor = getByRole('textbox');
    fireEvent.change(editor, { target: { value: 'new code' } });

    expect(onCodeChange).toHaveBeenCalledWith('new code');
  });

  it('should trigger save function on button click', () => {
    const onSave = jest.fn();
    const { getByText } = render(<CodeEditor initialCode="initial code"
onCodeChange={() => {}} onSave={onSave} />);

    const saveButton = getByText('Save');
    fireEvent.click(saveButton);

    expect(onSave).toHaveBeenCalledWith('initial code');
  });
});
```

5.4 Integration Testing

```
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import CodeEditor from './CodeEditor';
import Terminal from './Terminal';

describe('Collaborative Code Editor Integration', () => {
  it('should execute command and update code', () => {
    const onCommand = jest.fn().mockImplementation((command) => {
      if (command === 'npm install') {
        // Simulate successful command execution
        return 'Dependencies installed';
      }
    });

    const { getByRole, getByText } = render(
      <div>
        <CodeEditor initialCode="initial code" onCodeChange={() => {}} onSave={()
=> {}} />
        <Terminal onCommand={onCommand} />
      </div>
    );

    const commandInput = getByRole('textbox');
    const executeButton = getByText('Execute');

    fireEvent.change(commandInput, { target: { value: 'npm install' } });
    fireEvent.click(executeButton);

    expect(onCommand).toHaveBeenCalledWith('npm install');
  });
});
```

5.4 Code Details

1. backend/server/index.ts

```
import express, { Express } from 'express'
import { createServer } from 'http'
import { Server } from 'socket.io'
import { z } from 'zod'
import getVirtualboxFiles from './getVirtualboxFiles'
import {
  createFile,
  deleteFile,
  generateCode,
  getFolder,
  getProjectSize,
  renameFile,
  saveFile,
} from './utils'
import path from 'path'
import fs from 'fs'
import { IDisposable, IPty, spawn } from 'node-pty'
import os from 'os'
import { Virtualbox } from './types'
import {
  MAX_BODY_SIZE,
  createFileRL,
  createFolderRL,
  deleteFileRL,
  renameFileRL,
  saveFileRL,
} from './ratelimit'

const app: Express = express()

const port = process.env.PORT || 4000
const httpServer = createServer(app)

let inactivityTimeout: NodeJS.Timeout | null = null
let isOwnerConnected = false

const terminals: {
  [id: string]: {
    terminal: IPty
    onData: IDisposable
    onExit: IDisposable
  }
} = {}
const dirName = path.join(__dirname, '..')

const io = new Server(httpServer, {
  cors: {
    origin: '*',
  },
})

const handshakeSchema = z.object({
  userId: z.string(),
  virtualboxId: z.string(),
  // type: z.enum(['node', 'react']),
  EIO: z.string(),
  transport: z.string(),
  t: z.string(),
})

io.use(async (socket, next) => {
  const q = socket.handshake.query
  // console.log('middleware')
  // console.log(q)

  const parseQuery = handshakeSchema.safeParse(q)
  if (!parseQuery.success) {
    console.error('Validation error:', parseQuery.error)
// Log the error details

    next(new Error('Invalid request'))
    return
  }

  const { virtualboxId, userId } = parseQuery.data

  const dbUser = await fetch(

`https://database.vigneshpamu2002.workers.dev/api/user?id
=${userId}`
  )
  const dbUserJSON = await dbUser.json()

  if (!dbUserJSON) {
    next(new Error('DB error'))
    return
  }

  console.log(virtualboxId, 'Check value')

  if (!Array.isArray(dbUserJSON.virtualbox)) {
    console.error('dbUserJSON.virtualbox is not defined
or not an array')
    return
  }

  const virtualbox = dbUserJSON?.virtualbox?.find(
    (v: Virtualbox) => v.id === virtualboxId
  )

  if (!virtualbox) {
    next(new Error('Invalid credentials'))
    return
  }

  const sharedVirtualboxes =
dbUserJSON.usersToVirtualboxes.find(
    (utv: any) => utv.virtualboxId === virtualboxId
  )

  if (!virtualbox && !sharedVirtualboxes) {
    next(new Error('Invalid credentials'))
    return
  }

  socket.data = {
    id: virtualboxId,
    userId,
    isOwner: virtualbox !== undefined,
  }

  next()
})

io.on('connection', async (socket) => {
  if (inactivityTimeout) clearTimeout(inactivityTimeout)

  const data = socket.data as {
    userId: string
    id: string
    isOwner: boolean
  }

  if (data.isOwner) {
    isOwnerConnected = true
  } else if (!isOwnerConnected) {
    console.log('the virtual box owner not connected')
    socket.emit('disableAccess', 'The virtualbox owner is
not connected.')
    return
  }

  const virtualboxFiles = await
getVirtualboxFiles(data.id)
  virtualboxFiles.fileData.forEach((file) => {
    const filePath = path.join(dirName, file.id)
    fs.mkdirSync(path.dirname(filePath), { recursive:
true })
    fs.writeFile(filePath, file.data, function (err) {
      if (err) throw err
    })
  })

  socket.emit('loaded', virtualboxFiles.files)

  socket.on('getFile', (fileId: string, callback) => {
    const file = virtualboxFiles.fileData.find((f) =>
f.id === fileId)
    if (!file) return

    callback(file.data)
  })

  // socket.on('saveFile', async (fileId: string, body:
string) => {
  //   try {
  //     console.log('Yaha tak toh pauch raha hai ')
  //     await saveFileRL.consume(data.userId, 1)

  //     if (Buffer.byteLength(body, 'utf-8') >
MAX_BODY_SIZE) {
  //       console.log('Yes this is happening')
  //       socket.emit(
  //          'rateLimit',
  //          'Rate limited: file size too large. Please
reduce the file size.'
  //        )
  //        return
  //      }

  //      const file = virtualboxFiles.fileData.find((f)
=> f.id === fileId)
  //      if (!file) return

  //      file.data = body

  //      fs.writeFile(path.join(dirName, file.id), body,
function (err) {
  //        console.log('Body is getting appended',
JSON.stringify(body))
  //        if (err) throw err
  //      })

  //      console.log('Saving the file')

  //      await saveFile(fileId, body)
  //    } catch (e) {
  //      io.emit('rateLimit', 'Rate limited: file saving.
Please slow down.')
  //    }
  // })

  socket.on('saveFile', async (fileId: string, body:
string) => {
    try {
      console.log('Yaha tak toh pauch raha hai ')

      const file = virtualboxFiles.fileData.find((f) =>
f.id === fileId)
```

```
      if (!file) return

      file.data = body

      fs.writeFile(path.join(dirName, file.id), body,
function (err) {
        console.log('Body is getting appended',
JSON.stringify(body))
        if (err) throw err
      })

      console.log('Saving the file')

      await saveFile(fileId, body)
    } catch (e) {
      io.emit('error', 'Error saving file. Please try
again.')
    }
  })

  socket.on('createFile', async (name: string, callback)
=> {
    try {
      const size: number = await getProjectSize(data.id)
      if (size > 200 * 1024 * 1024) {
        io.emit(
          'rateLimit',
          'Rate Limited: project size exceeded. Please
delete some files.'
        )
        callback({ success: false })
      }
      await createFileRL.consume(data.userId, 1)
      const id = `projects/${data.id}/${name}`

      fs.writeFile(path.join(dirName, id), '', function
(err) {
        if (err) throw err
      })

      virtualboxFiles.files.push({
        id,
        name,
        type: 'file',
      })

      virtualboxFiles.fileData.push({
        id,
        data: '',
      })

      await createFile(id)
      callback({ success: true })
    } catch (e) {
      io.emit('rateLimit', 'Rate limited: file saving.
Please slow down.')
    }
  })

  socket.on('moveFile', async (fileId: string, folderId:
string, callback) => {
    const file = virtualboxFiles.fileData.find((f) =>
f.id === fileId)

    if (!file) return

    const parts = fileId.split('/')
    const newFileId = folderId + '/' + parts.pop()

    fs.rename(
      path.join(dirName, fileId),
      path.join(dirName, newFileId),
      function (err) {
        if (err) throw err
      }
    )

    file.id = newFileId

    await renameFile(fileId, newFileId, file.data)
    const newFiles = await getVirtualboxFiles(data.id)

    callback(newFiles.files)
  })

  socket.on('getFolder', async (folderId: string,
callback) => {
    const files = await getFolder(folderId)
    callback(files)
  })

  socket.on('deleteFolder', async (folderId: string,
callback) => {
    const files = await getFolder(folderId)

    await Promise.all(
      files.map(async (file) => {
        fs.unlink(path.join(dirName, file), function
(err) {
          if (err) throw err
        })

        virtualboxFiles.fileData =
virtualboxFiles.fileData.filter(
          (f) => f.id !== file
        )

        await deleteFile(file)
      })
    )

    const newFiles = await getVirtualboxFiles(data.id)

    callback(newFiles.files)
  })

  socket.on('renameFolder', async (folderId: string,
callback) => {})

  socket.on('createFolder', async (name: string,
callback) => {
    try {
      await createFolderRL.consume(data.userId, 1)

      const id = `projects/${data.id}/${name}`

      fs.mkdir(path.join(dirName, id), { recursive: true
}, function (err) {
        if (err) throw err
      })

      callback()
    } catch (e) {
      io.emit('rateLimit', 'Rate limited: folder
creation. Please slow down')
    }
  })

  socket.on('deleteFile', async (fileId: string,
callback) => {
    try {
      await deleteFileRL.consume(data.userId, 1)
      const file = virtualboxFiles.fileData.find((f) =>
f.id === fileId)
```

```
      if (!file) return

      fs.unlink(path.join(dirName, fileId), function
(err) {
        if (err) throw err
      })

      virtualboxFiles.fileData =
virtualboxFiles.fileData.filter(
        (f) => f.id !== fileId
      )

      await deleteFile(fileId)

      const newFiles = await getVirtualboxFiles(data.id)
      callback(newFiles.files)
    } catch (e) {
      io.emit('rateLimit', 'Rate limited: file saving.
Please slow down.')
    }
  })

  socket.on('resizeTerminal', (dimensions: { cols:
number; rows: number }) => {
    Object.values(terminals).forEach((t) => {
      t.terminal.resize(dimensions.cols, dimensions.rows)
    })
  })

  socket.on('renameFile', async (fileId: string, newName:
string) => {
    try {
      await renameFileRL.consume(data.userId, 1)
      const file = virtualboxFiles.fileData.find((f) =>
f.id === fileId)
```

```
      if (!file) return

      file.id = newName
      const parts = fileId.split('/')
      const newFileId =
        parts.slice(0, parts.length - 1).join('/') + '/'
+ newName

      fs.rename(
        path.join(dirName, fileId),
        path.join(dirName, newFileId),
        function (err) {
          if (err) throw err
        }
      )
      await renameFile(fileId, newFileId, file.data)
    } catch (e) {
      io.emit('rateLimit', 'Rate limited: file saving.
Please slow down.')
    }
  })

  socket.on('createTerminal', (id: string, callback) => {
    if (terminals[id] || Object.keys(terminals).length >=
4) {
      return
    }

    console.log('creating terminal (' + id + ')')
    const pty = spawn(os.platform() === 'win32' ?
'cmd.exe' : 'bash', [], {
      name: 'xterm',
      cols: 100,
      cwd: path.join(dirName, 'projects', data.id),
    })
    const onData = pty.onData((data) => {
      io.emit('terminalResponse', {
        id,
        data,
      })
    })

    const onExit = pty.onExit((code) =>
console.log('exit:(', code))
    pty.write('clear\r')
    terminals[id] = {
      terminal: pty,
      onData,
      onExit,
    }

    callback()
  })

  socket.on('closeTerminal', (id: string, callback) => {
    if (!terminals[id]) {
      console.log(
        'tried to close, but term does not exists.
terminals',
        terminals
      )
      return
    }

    terminals[id].onData.dispose()
    terminals[id].onExit.dispose()

    delete terminals[id]

    callback()
  })
  socket.on('terminalData', (id: string, data: string) =>
{
    console.log(`Received data for terminal ${id}:
${data}`)
    if (!terminals[id]) {
      return
    }

    try {
      terminals[id].terminal.write(data)
    } catch (e) {
      console.log('Error writing to terminal', e)
    }
  })

  socket.on(
    'generateCode',
    async (
      fileName: string,
      code: string,
      line: number,
      instructions: string,
      callback
    ) => {
      const fetchPromise = fetch(

`https://database.vigneshpamu2002.workers.dev/api/virtual
box/generate`,
        {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify({
            userId: data.userId,
          }),
        }
      )

      const generateCodePromise = generateCode({
        fileName,
        code,
        line,
        instructions,
      })

      const [fetchResponse, generateCodeResponse] = await
Promise.all([
        fetchPromise,
        generateCodePromise,
      ])
      const json = await generateCodeResponse.json()
      callback(json)

      console.log(json, 'check json')
      callback(json)
    }
  )

  socket.on('disconnect', async () => {
    if (data.isOwner) {
      Object.entries(terminals).forEach((t) => {
        const { terminal, onData, onExit } = t[1]
        if (os.platform() !== 'win32') terminal.kill()
        onData.dispose()
        onExit.dispose()
        delete terminals[t[0]]
      })

      console.log('The owner disconnected')
      socket.broadcast.emit('ownerDisconnected')
    } else {
      console.log('A shared user disconnected.')
      socket.broadcast.emit(
        'disableAccess',
        'The virtualbox owner has disconnected.'
      )
    }

    const sockets = await io.fetchSockets()
    if (inactivityTimeout) {
      clearTimeout(inactivityTimeout)
    }
    if (sockets.length === 0) {
      inactivityTimeout = setTimeout(() => {
        io.fetchSockets().then((sockets) => {
          if (sockets.length === 0) {
            console.log('No users have been connected for
15 seconds')
          }
        })
      }, 15000)
    }
  })
})

httpServer.listen(port, () => {
  console.log(`Server running on port ${port}`)
})
```

2. backend/server/utils.ts

```
// import {
//   DeleteServiceCommand,
//   DescribeServicesCommand,
//   ECSClient,
//   StopTaskCommand,
// } from '@aws-sdk/client-ecs'
import { R2Files } from './types'
import { error } from 'console'

// const client = new ECSClient({
//   region: 'us-east-1',
//   credentials: {
//     accessKeyId: '',
//     secretAccessKey: '',
//   },
// })

// export const testDescribe = async () => {
//   const command = new DescribeServicesCommand({
//     cluster: 'virtualboxccce',
//     services: ['virtualboxccce'],
//   })

//   const response = await client.send(command)
//   console.log('describing:', response)
//   return response
// }

// export const stopServer = async (service: string) => {
//   const command = new DeleteServiceCommand({
//     cluster: 'virtualboxccce',
//     service,
//     force: true,
//   })

//   try {
//     const response = await client.send(command)
//     console.log('Stopped server:', response)
//   } catch (error) {
//     console.error('Error stopping server: ', error)
//   }
// }

export const renameFile = async (
  fileId: string,
  newFileId: string,
  data: string
) => {
  const res = await fetch(

`https://storage.vigneshpamu2002.workers.dev/api/rename`,
    {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ fileId, newFileId, data }),
    }
  )

  console.log(res)

  return res.ok
}

export const saveFile = async (fileId: string, data:
string) => {
  console.log('This is also running')
  const res = await fetch(

`https://storage.vigneshpamu2002.workers.dev/api/save`,
    {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ fileId, data }),
    }
  )

  console.log('Received the response')

  return res.ok
}

export const createFile = async (fileId: string) => {
  const res = await
fetch(`https://storage.vigneshpamu2002.workers.dev/api`,
{
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ fileId }),
  })

  return res.ok
}

export const deleteFile = async (fileId: string) => {
  const res = await
fetch(`https://storage.vigneshpamu2002.workers.dev/api`,
{
    method: 'DELETE',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ fileId }),
  })
  return res.ok
}

export const generateCode = async ({
  fileName,
  code,
  line,
  instructions,
}: {
  fileName: string
  code: string
  line: number
  instructions: string
}) => {
  return await fetch(

'https://api.cloudflare.com/client/v4/accounts/dfe11e43c5
ce7c71a9b7ce503ac1e8d5/ai/run/@cf/meta/llama-3-8b-instruc
t',
    {
      method: 'POST',
      headers: {
        Authorization: 'Bearer
pFVOnQSHBEjvxnxuY456dtXixW-gMWsRpKlkmJ7v',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        messages: [
          {
            role: 'system',
            content:
              'You are an expert coding assistant who
reads from an existing code file, and suggests code to
add to the file. You may be given instructions on what to
generate, which you should follow. You should generate
code that is correct, efficient, and follows best
practices. You should also generate code that is clear
and easy to read.',
          },
          {
            role: 'user',
            content: `The file is called ${fileName}.`,
          },
          {
            role: 'user',
            content: `Here are my instructions on what to
generate: ${instructions}.`,
          },
          {
            role: 'user',
            content: `Suggest me code to insert at line
${line} in my file. Give only the code, and NOTHING else.
DO NOT include backticks in your response. My code file
content is as follows

            ${code}`,
          },
        ],
      }),
    }
  )
}

export const getProjectSize = async (id: string) => {
  const res = await fetch(

`https://storage.vigneshpamu2002.workers.dev/api/size?vir
tualboxId=${id}`
  )

  return (await res.json()).size
}
```

```
export const getFolder = async (folderId: string) => {
  const res = await fetch(

`https://storage.vigneshpamu2002.workers.dev/api?folderId
=${folderId}`
  )

  const data: R2Files = await res.json()

  return data.objects.map((obj) => obj.key)
}
```

3. frontend/app/(auth)/sign-in

```
import { SignIn } from '@clerk/nextjs'
import { dark } from '@clerk/themes'

export default function Page() {
  return (
    <SignIn
      appearance={{
        baseTheme: dark,
        elements: {
          footerActionLink: {
            color: '#fff',
          },
        },
      }}
    />
  )
}
```

4. frontend/app/(auth)/sign-up

```
import { SignUp } from '@clerk/nextjs'
import { dark } from '@clerk/themes'

export default function Page() {
  return (
    <SignUp
      appearance={{
        baseTheme: dark,
        elements: {
          footerActionLink: {
            color: '#A3A3A3',
          },
        },
      }}
    />
  )
}
```

5. frontend/app/code/[id]/page.tsx

```
import Navbar from '@/components/editor/navbar'
import { Room } from '@/components/editor/live/room'
import { TFile, TFolder } from
'@/components/editor/sidebar/types'
import { R2Files, User, UsersToVirtualboxes, Virtualbox }
from '@/lib/types'
import { currentUser } from '@clerk/nextjs/server'
import dynamic from 'next/dynamic'
import { notFound, redirect } from 'next/navigation'
```

```
const CodeEditor = dynamic(() =>
import('@/components/editor'), {
  ssr: false,
})

const getUserData = async (id: string) => {
  const userRes = await fetch(

`https://database.vigneshpamu2002.workers.dev/api/user?id
=${id}`
  )
  const userData = (await userRes.json()) as User
  return userData
}

const getVirtualboxData = async (id: string) => {
  const virtualboxRes = await fetch(

`https://database.vigneshpamu2002.workers.dev/api/virtual
box?id=${id}`
  )
  const virtualboxData: Virtualbox = await
virtualboxRes.json()
  return virtualboxData
}

const getSharedUsers = async (usersToVirtualboxes:
UsersToVirtualboxes[]) => {
  const shared = await Promise.all(
    usersToVirtualboxes?.map(async (user) => {
      const userRes = await fetch(

`https://database.vigneshpamu2002.workers.dev/api/user?id
=${user.userId}`
      )
      const userData: User = await userRes.json()
      return { id: userData.id, name: userData.name }
    })
  )
  return shared
}

export default async function CodePage({ params }: {
params: { id: string } }) {
  const user = await currentUser()
  const virtualboxId = params.id
  if (!user) {
    redirect('/')
  }

  const userData = await getUserData(user.id)
  const virtualboxData = await
getVirtualboxData(virtualboxId)
  const shared = await
getSharedUsers(virtualboxData.usersToVirtualboxes ?? [])
  const isOwner = virtualboxData.userId === user.id

  const isSharedUser = shared.some((utv) => utv.id ===
user.id)

  if (!isOwner && !isSharedUser) {
    return notFound()
  }

  return (
    <div className="flex w-screen flex-col h-screen
bg-background">
      <Room id={virtualboxId}>
        <Navbar
          userData={userData}
          virtualboxData={virtualboxData}
          shared={shared}
        />
        <div className="w-screen flex grow ">
          <CodeEditor
            userData={userData}
            virtualboxData={virtualboxData}
            isSharedUser={isSharedUser}
          />
        </div>
      </Room>
    </div>
  )
}
```

6. frontend/app/dashboard/page.tsx

```
import Dashboard from '@/components/dashboard'
import Navbar from '@/components/dashboard/navbar'
import { User, Virtualbox } from '@/lib/types'
import { currentUser } from '@clerk/nextjs/server'
import { redirect } from 'next/navigation'

export default async function DashboardPage() {
  const user = await currentUser()

  if (!user) {
    redirect('/')
  }

  console.log(user.id)

  const userRes = await fetch(

`https://database.vigneshpamu2002.workers.dev/api/user?id
=${user.id}`
  )
  const userData = (await userRes.json()) as User
  console.log(userData)

  const sharedRes = await fetch(

`https://database.vigneshpamu2002.workers.dev/api/virtual
box/share?id=${user.id}`
  )

  const shared = (await sharedRes.json()) as {
    id: string
    name: string
    type: 'react' | 'node'
    author: {
      id: string
      name: string
      email: string
      image: any
    }
    sharedOn: Date
  }[]

  console.log('shared: ', shared)

  return (
    <div>
      <Navbar userData={userData} />
      <Dashboard virtualboxes={userData.virtualbox}
shared={shared} />
    </div>
  )
}
```

7. frontend/app/leetcode/page.tsx

```
'use client'
import ProblemsTable from
'@/leetcode_components/ProblemsTable/ProblemsTable'
import Topbar from '@/leetcode_components/Topbar/Topbar'
import useHasMounted from
'../../../_leetcode_things/hooks/useHasMounted'

import { useState } from 'react'

export default function Home() {
  const [loadingProblems, setLoadingProblems] =
useState(true)
  const hasMounted = useHasMounted()

  if (!hasMounted) return null

  return (
    <>
      <main className="bg-dark-layer-2 min-h-screen">
        <Topbar />
        <h1
          className="text-2xl text-center text-gray-700
dark:text-gray-400 font-medium
                        uppercase mt-10 mb-5"
        >
          &ldquo; QUALITY OVER QUANTITY &rdquo; 👇
        </h1>
        <div className="relative overflow-x-auto mx-auto
px-6 pb-10">
          {loadingProblems && (
            <div className="max-w-[1200px] mx-auto
sm:w-7/12 w-full animate-pulse">
              {[...Array(10)].map((_, idx) => (
                <LoadingSkeleton key={idx} />
              ))}
            </div>
          )}
          <table className="text-sm text-left 
text-gray-500 dark:text-gray-400 sm:w-7/12 w-full 
max-w-[1200px] mx-auto">
            {!loadingProblems && (
              <thead className="text-xs text-gray-700 
uppercase dark:text-gray-400 border-b ">
                <tr>
                  <th scope="col" className="px-1 py-3 
w-0 font-medium">
                    Status
                  </th>
                  <th scope="col" className="px-6 py-3 
w-0 font-medium">
                    Title
                  </th>
                  <th scope="col" className="px-6 py-3 
w-0 font-medium">
                    Difficulty
                  </th>

                  <th scope="col" className="px-6 py-3 
w-0 font-medium">
                    Category
                  </th>
                  <th scope="col" className="px-6 py-3 
w-0 font-medium">
                    Solution
                  </th>
                </tr>
              </thead>
            )}
            <ProblemsTable 
setLoadingProblems={setLoadingProblems} />
          </table>
```

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

const LoadingSkeleton = () => {
  return (
    <div className="flex items-center space-x-12 mt-4
px-6">
      <div className="w-6 h-6 shrink-0 rounded-full
bg-dark-layer-1"></div>
      <div className="h-4 sm:w-52  w-32  rounded-full
bg-dark-layer-1"></div>
      <div className="h-4 sm:w-52  w-32 rounded-full
bg-dark-layer-1"></div>
      <div className="h-4 sm:w-52 w-32 rounded-full
bg-dark-layer-1"></div>
      <span className="sr-only">Loading...</span>
    </div>
  )
}
```

8. frontend/app/my-resume/[resumeId]/view/page.tsx

```
'use client'
import Header from '@/resume_components/custom/Header'
import { Button } from
'@/resume_components/ui_two/button'
import { ResumeInfoContext } from
'@/context/ResumeInfoContext'
import React, { useEffect, useRef, useState } from
'react'
import { RWebShare } from 'react-web-share'
import ResumePreview from
```

```
'../../../resume-dashboard/resume/components/ResumePrevie
w'
import { useParams, useRouter } from 'next/navigation'
function ViewResume() {
  const [resumeInfo, setResumeInfo] = useState()
  const params = useParams()
  const { resumeId } = params

  useEffect(() => {
    GetResumeInfo()
  }, [])
  const GetResumeInfo = async () => {
    try {
      const resp = await
fetch(`/api/get-resume-by-id?resumeId=${resumeId}`)
      const data = await resp.json()

      if (resp.ok) {
        setResumeInfo(data.data.data)
      }
    } catch (err) {
    } finally {
    }
  }
  const divRef = useRef()

  const HandleDownload = () => {
    // Get the print and no-print areas
    const printArea =
document.getElementById('print-area')
    const noPrintArea =
document.getElementById('no-print')

    // Save the current body margin and padding values
for restoration later
    const originalBodyMargin = document.body.style.margin
    const originalBodyPadding =
document.body.style.padding
    const originalNoPrintDisplay = noPrintArea ?
noPrintArea.style.display : ''

    // Hide other areas that should not be printed
    if (noPrintArea) noPrintArea.style.display = 'none'

    // Temporarily apply print-specific styles
    document.body.style.margin = '0'
    document.body.style.padding = '0'

    // Remove margins and padding for print-area and all
other elements
    printArea.style.margin = '0'
    printArea.style.padding = '0'

    // Apply page-specific print styles
    const printStyles = document.createElement('style')
    printStyles.innerHTML = `
    @media print {
      body, html {
        margin: 0;
        padding: 0;
        width: 100%;
        height: 100%;
        overflow: hidden;
      }

      #print-area {
        margin: 0 !important;
        padding: 0 !important;
        width: 100%;
        height:auto;
        page-break-inside: avoid;
      }
      * {
      }

      @page {
        margin: -40px -40px 0px -40px !important;

        // padding: -15px !important;
      }
    }
  `
    document.head.appendChild(printStyles)

    // Trigger the print dialog
    window.print()

    // Restore the original styles after printing
    document.body.style.margin = originalBodyMargin
    document.body.style.padding = originalBodyPadding
    if (noPrintArea) noPrintArea.style.display =
originalNoPrintDisplay

    // Clean up the added print styles
    document.head.removeChild(printStyles)
  }

  // const HandleDownload = async () => {
  //   const element = divRef.current

  //   // Capture the content of the div as a canvas
  //   const canvas = await html2canvas(element, {
  //     scale: 2, // Higher scale for better resolution
  //   })
  //   const imgData = canvas.toDataURL('image/png')

  //   // Create a PDF with A4 dimensions
  //   const pdf = new jsPDF({
  //     orientation: 'portrait',
  //     unit: 'mm',
  //     format: 'a4', // A4 size
  //   })

  //   const pdfWidth = 210 // A4 width in mm
  //   const pdfHeight = 297 // A4 height in mm

  //   const imgWidth = pdfWidth
  //   const imgHeight = (canvas.height * pdfWidth) /
canvas.width

  //   // Add image to the PDF
  //   pdf.addImage(imgData, 'PDF', 0, 0, imgWidth,
imgHeight)

  //   // Save the PDF with a specific name
  //   pdf.save('resume.pdf')
  // }

  return (
    <ResumeInfoContext.Provider value={{ resumeInfo,
setResumeInfo }}>
      <div id="no-print">
        {/* <Header /> */}

        <div className="my-10 mx-10 md:mx-20 lg:mx-36">
          <h2 className="text-center text-2xl
font-medium">
            Congrats! Your Ultimate AI generates Resume
is ready !{' '}
          </h2>
          <p className="text-center text-gray-400">
            Now you are ready to download your resume and
you can share unique
            resume url with your friends and family{' '}
          </p>
          <div className="flex justify-between px-44
my-10">
            <Button
onClick={HandleDownload}>Download</Button>

            {/* <RWebShare
              data={{
                text: 'Hello Everyone, This is my resume
please open url to see it',
                url:
                  process.env.VITE_BASE_URL +
                  '/my-resume/' +
                  resumeId +
                  '/view',
                title:
                  resumeInfo?.firstName +
                  ' ' +
                  resumeInfo?.lastName +
                  ' resume',
              }}
              onClick={() => console.log('shared
successfully!')}
            >
              {' '}
              <Button>Share</Button>
            </RWebShare> */}
          </div>
        </div>
      </div>
      <div className="my-10 mx-10 md:mx-20 lg:mx-36">
        <div
```

```
          ref={divRef}
          style={{
            // width: '750px', // Match A4 width
            // height: '297mm', // Match A4 height
            // padding: '20px',
            // border: '1px solid black',
            // margin: '0px auto',
            background: 'white',
          }}
          id="print-area"
        >
          <ResumePreview />
        </div>
      </div>
    </ResumeInfoContext.Provider>
  )
}

export default ViewResume
```

9. frontend/app/problems/page.tsx

```
// @ts-nocheck
import Topbar from '@/leetcode_components/Topbar/Topbar'
import Workspace from
'@/leetcode_components/Workspace/Workspace'
import { problems } from '@/utils/problems'
import { Problem } from '@/utils/types/problem'

export async function generateStaticParams() {
  const paths = Object.keys(problems).map((key) => ({
    pid: key,
  }))
  return paths.map(({ pid }) => ({
    pid,
  }))
}

export async function generateMetadata({
  params,
}: {
  params: { pid: string }
}) {
  const problem = problems[params.pid]
  if (problem) {
    return {
      title: problem.title,
      description: problem.description,
    }
  }
  return {}
}

export default async function ProblemPage({
  params,
}: {
  params: { pid: string }
}) {
  const problem = problems[params.pid]

  if (!problem) {
    return {
      notFound: true,
    }
  }

  problem.handlerFunction =
problem.handlerFunction.toString()

  return (
    <div>
      <Topbar problemPage />
      <Workspace problem={problem} />
    </div>
  )
}
```

10. frontend/app/resume/page.tsx

```
'use client'
import React, { useEffect, useState } from 'react'
import AddResume from './components/AddResume'
import { useUser } from '@clerk/nextjs'
import ResumeCardItem from './components/ResumeCardItem'
import axios from 'axios'

function Dashboard() {
  const { user } = useUser()
  const [resumeList, setResumeList] = useState([])

  useEffect(() => {
    user && GetResumesList()
  }, [user])

  const GetResumesList = async () => {
    try {
      const response = await fetch(
        `/api/get-resume?userEmail=${
          user?.primaryEmailAddress?.emailAddress
            ? user?.primaryEmailAddress?.emailAddress
            : 'vigneshpamu2002@gmail.com'
        }`
      )
      const data = await response.json()
      console.log(data.data.data)
      setResumeList(data.data.data)
    } catch (error) {
      console.error('Error fetching resumes:',
error.message)
    }
  }

  return (
    <div className="p-10 md:px-20 lg:px-32">
      <h2 className="font-bold text-3xl">My Resume</h2>
      <p>Start Creating AI resume to your next Job
role</p>
      <div
        className="grid grid-cols-2
      md:grid-cols-3 lg:grid-cols-5 gap-5
      mt-10
      "
      >
        <AddResume />
        {resumeList?.length > 0
          ? resumeList.map((resume, index) => (
              <ResumeCardItem
                resume={resume}
                key={index}
                refreshData={GetResumesList}
              />
            ))
          : [1, 2, 3, 4].map((item, index) => (
              <div
                key={index}
                className="h-[280px] rounded-lg
bg-slate-200 animate-pulse"
              ></div>
            ))}
      </div>
    </div>
  )
}

export default Dashboard
```

11. frontend/app/resume/resume/[resumeId]/edit/page.tsx

```
'use client'
import React, { useEffect, useState } from 'react'
import { useRouter } from 'next/navigation' // Use
Next.js useRouter for dynamic routes
import ResumePreview from
'../../components/ResumePreview'
import { ResumeInfoContext } from
'../../../../../../context/ResumeInfoContext'
import FormSection from '../../components/FormSection'
import { useParams } from 'next/navigation'

function EditResume() {
  const router = useRouter()
  const params = useParams()

  const { resumeId } = params // Access the dynamic route
parameter
  const [resumeInfo, setResumeInfo] = useState(null)
  const [loading, setLoading] = useState(true) // Add
loading state
  const [error, setError] = useState(null) // Add error
state

  useEffect(() => {
    if (resumeId) {
      GetResumeInfo() // Fetch data only when resumeId is
available
    }
  }, [resumeId])

  const GetResumeInfo = async () => {
    try {
      setLoading(true)
      setError(null)
      const resp = await
fetch(`/api/get-resume-by-id?resumeId=${resumeId}`)
      const data = await resp.json()

      if (resp.ok) {
        setResumeInfo(data.data.data)
      } else {
        setError(data.error || 'Failed to load resume
data.')
      }
    } catch (err) {
      setError('Failed to load resume data. Please try
again later.')
    } finally {
      setLoading(false)
    }
  }

  if (loading) {
    return <div>Loading...</div> // You can replace this
with a spinner component or custom loader
  }

  if (error) {
    return <div>{error}</div> // Show error message if
the API call fails
  }

  return (
    <ResumeInfoContext.Provider value={{ resumeInfo,
setResumeInfo }}>
      <div className="grid grid-cols-1 md:grid-cols-2
p-10 gap-10">
        <FormSection />
        <ResumePreview />
      </div>
    </ResumeInfoContext.Provider>
  )
}

export default EditResume
```

12. frontend/app/layout.tsx

```
import { currentUser } from '@clerk/nextjs/server'
import { redirect } from 'next/navigation'

export default async function AppAuthLayout({
  children,
}: {
  children: React.ReactNode
}) {
  const user = await currentUser()

  if (!user) {
    redirect('/')
  }

  const dbUser = await fetch(

`https://database.vigneshpamu2002.workers.dev/api/user?id
=${user.id}`
  )
  const dbUserJSON = await dbUser.json()

  if (!dbUserJSON.id) {
    const res = await fetch(

'https://database.vigneshpamu2002.workers.dev/api/user',
      {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          id: user.id,
          name: user.firstName + ' ' + user.lastName,
          email: user.emailAddresses[0].emailAddress,
        }),
      }
    )
  } else {
  }

  return <>{children}</>
}
```

13. strapi_backend/config/database.js

```
const path = require('path');

module.exports = ({ env }) => {
  const client = env('DATABASE_CLIENT', 'sqlite');

  const connections = {
    mysql: {
      connection: {
        host: env('DATABASE_HOST', 'localhost'),
        port: env.int('DATABASE_PORT', 3306),
        database: env('DATABASE_NAME', 'strapi'),
        user: env('DATABASE_USERNAME', 'strapi'),
        password: env('DATABASE_PASSWORD', 'strapi'),
        ssl: env.bool('DATABASE_SSL', false) && {
          key: env('DATABASE_SSL_KEY', undefined),
          cert: env('DATABASE_SSL_CERT', undefined),
          ca: env('DATABASE_SSL_CA', undefined),
          capath: env('DATABASE_SSL_CAPATH', undefined),
          cipher: env('DATABASE_SSL_CIPHER', undefined),
          rejectUnauthorized:
env.bool('DATABASE_SSL_REJECT_UNAUTHORIZED', true),
        },
      },
      pool: { min: env.int('DATABASE_POOL_MIN', 2), max:
env.int('DATABASE_POOL_MAX', 10) },
    },
    postgres: {
      connection: {
        connectionString: env('DATABASE_URL'),
        host: env('DATABASE_HOST', 'localhost'),
        port: env.int('DATABASE_PORT', 5432),
        database: env('DATABASE_NAME', 'strapi'),
        user: env('DATABASE_USERNAME', 'strapi'),
        password: env('DATABASE_PASSWORD', 'strapi'),
        ssl: env.bool('DATABASE_SSL', false) && {
          key: env('DATABASE_SSL_KEY', undefined),
          cert: env('DATABASE_SSL_CERT', undefined),
          ca: env('DATABASE_SSL_CA', undefined),
          capath: env('DATABASE_SSL_CAPATH', undefined),
          cipher: env('DATABASE_SSL_CIPHER', undefined),
          rejectUnauthorized:
env.bool('DATABASE_SSL_REJECT_UNAUTHORIZED', true),
        },
        schema: env('DATABASE_SCHEMA', 'public'),
      },
      pool: { min: env.int('DATABASE_POOL_MIN', 2), max:
env.int('DATABASE_POOL_MAX', 10) },
    },
    sqlite: {
      connection: {
        filename: path.join(__dirname, '..',
env('DATABASE_FILENAME', '.tmp/data.db')),
      },
      useNullAsDefault: true,
    },
  };

  return {
    connection: {
      client,
      ...connections[client],
      acquireConnectionTimeout:
env.int('DATABASE_CONNECTION_TIMEOUT', 60000),
    },
  };
};
```

14. backend/server/package.json

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "dist/index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "npx tsc",
    "start": "node dist/index.js",
    "dev": "nodemon src/index.ts"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "node-pty": "^1.0.0",
    "rate-limiter-flexible": "^5.0.3",
    "socket.io": "^4.7.5",
    "zod": "^3.23.8"
  },
  "devDependencies": {
    "@types/cors": "^2.8.17",
    "@types/express": "^4.17.21",
    "@types/node": "^20.14.11",
    "concurrently": "^8.2.2",
    "nodemon": "^3.1.4",
    "ts-node": "^10.9.2",
    "typescript": "^5.5.3"
  }
}
```

15. frontend/package.json

```
{
  "name": "ccce",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@atlaskit/pragmatic-drag-and-drop": "^1.2.3",
    "@aws-sdk/client-ecs": "^3.637.0",
    "@clerk/clerk-react": "^5.2.4",
    "@clerk/nextjs": "^5.2.3",
    "@clerk/themes": "^2.1.10",
    "@codemirror/lang-javascript": "^6.1.6",
    "@google/generative-ai": "^0.12.0",
    "@hookform/resolvers": "^3.9.0",
    "@liveblocks/client": "^1.12.0",
    "@liveblocks/node": "^1.12.0",
    "@liveblocks/react": "^1.12.0",
    "@liveblocks/yjs": "^1.12.0",
    "@monaco-editor/react": "^4.6.0",
    "@paralleldrive/cuid2": "^2.2.2",
    "@radix-ui/react-alert-dialog": "^1.0.5",
    "@radix-ui/react-avatar": "^1.1.0",
    "@radix-ui/react-context-menu": "^2.2.1",
    "@radix-ui/react-dialog": "^1.1.1",
    "@radix-ui/react-dropdown-menu": "^2.1.1",
    "@radix-ui/react-label": "^2.1.0",
    "@radix-ui/react-popover": "^1.0.7",
    "@radix-ui/react-select": "^2.1.1",
    "@radix-ui/react-slot": "^1.1.0",
    "@radix-ui/react-switch": "^1.1.0",
    "@radix-ui/react-tabs": "^1.1.0",
    "@smastrom/react-rating": "^1.5.0",
    "@types/node": "20.14.10",
    "@types/react": "18.3.3",
    "@types/react-dom": "18.3.0",
    "@uiw/codemirror-theme-vscode": "^4.19.16",
    "@uiw/react-codemirror": "^4.19.16",
    "@uiw/react-split": "^5.9.3",
    "@xterm/addon-fit": "^0.10.0",
    "@xterm/xterm": "^5.5.0",
    "assert": "^2.0.0",
    "autoprefixer": "10.4.19",
    "axios": "^1.7.9",
    "class-variance-authority": "^0.7.0",
    "clsx": "^2.1.1",
    "firebase": "^9.18.0",
    "lucide-react": "^0.408.0",
    "next": "14.2.5",
    "next-themes": "^0.3.0",
    "postcss": "8.4.39",
    "react": "18.3.1",
    "react-confetti": "^6.1.0",
    "react-dom": "18.3.1",
    "react-firebase-hooks": "^5.1.1",
    "react-hook-form": "^7.52.1",
    "react-hot-toast": "^2.5.2",
    "react-icons": "^4.8.0",
    "react-resizable-panels": "^2.1.7",
    "react-router-dom": "^6.23.1",
    "react-simple-wysiwyg": "^3.0.2",
    "react-split": "^2.0.14",
    "react-split-pane": "^0.1.92",
    "react-to-pdf": "^1.0.1",
    "react-toastify": "^9.1.3",
    "react-web-share": "^2.0.2",
    "react-youtube": "^10.1.0",
    "recoil": "^0.7.7",
    "socket.io-client": "^4.7.5",
    "sonner": "^1.5.0",
    "tailwind-merge": "^2.4.0",
    "tailwindcss": "3.4.5",
    "tailwindcss-animate": "^1.0.7",
    "typescript": "5.5.3",
    "uuid": "^10.0.0",
    "vscode-icons-js": "^11.6.1",
    "y-monaco": "^0.1.5",
    "y-protocols": "^1.0.6",
    "yjs": "^13.6.16",
    "zod": "^3.23.8"
  }
}
```

16. strapi_backend/package.json

```
{
  "name": "ai-resume-admin",
  "version": "0.1.0",
  "private": true,
  "description": "A Strapi application",
  "scripts": {
    "build": "strapi build",
    "deploy": "strapi deploy",
    "develop": "strapi develop",
    "start": "strapi start",
    "strapi": "strapi"
  },
  "dependencies": {
    "@strapi/plugin-cloud": "5.5.1",
    "@strapi/plugin-users-permissions": "5.5.1",
    "@strapi/strapi": "5.5.1",
    "pg": "8.8.0",
    "react": "^18.0.0",
    "react-dom": "^18.0.0",
    "react-router-dom": "^6.0.0",
    "styled-components": "^6.0.0"
  },
  "devDependencies": {},
  "engines": {
    "node": ">=18.0.0 <=22.x.x",
    "npm": ">=6.0.0"
  },
  "strapi": {
    "uuid": "040c695b-80d8-4927-9dd6-9b7c40c8e042"
  }
}
```

Testing Approach

1. Unit Testing

- Implemented Jest and React Testing Library for frontend component testing.
- Used Mocha and Chai for backend API validation.
- Ensured individual functions and modules work as expected.

2. Integration Testing

- Verified seamless interaction between frontend and backend.
- Tested real-time collaboration in the code editor using WebSockets.
- Ensured smooth data flow in resume generation and problem-solving modules.

3. Beta Testing

- Conducted testing with a group of students to gather feedback.
- Identified usability issues and performance bottlenecks.
- Validated AI-generated code suggestions for accuracy.

4. Modifications and Improvements Test Cases

- Regularly updated test cases based on user feedback and bug reports.
- Ensured backward compatibility after feature enhancements.
- Monitored system performance post-deployment with Cloudflare analytics.

Chapter 6 - Implementation & Testing Approach

Test Cases

Test Case ID: TC-001

Test Case: User Registration
Description: Verify if a new user can register with valid details.
Expected Result: User should be registered successfully.
Actual Outcome: User registered successfully.
Status: Passed

Test Case ID: TC-002

Test Case: User Login
Description: Verify if a registered user can log in with correct credentials.
Expected Result: User should be able to log in.
Actual Outcome: User logged in successfully.
Status: Passed

Test Case ID: TC-003

Test Case: Resume Creation
Description: Verify if a user can create a resume using the resume builder.
Expected Result: Resume should be created successfully.
Actual Outcome: Resume created successfully.
Status: Passed

Test Case ID: TC-004

Test Case: Resume Download
Description: Verify if a user can download the created resume in PDF format.
Expected Result: Resume should be downloaded.
Actual Outcome: Resume downloaded successfully.
Status: Passed

Test Case ID: TC-005

Test Case: Workspace Creation
Description: Verify if a user can create a new workspace in the collaborative editor.
Expected Result: Workspace should be created successfully.
Actual Outcome: Workspace created successfully.
Status: Passed

Test Case ID: TC-006

Test Case: Code Execution
Description: Verify if users can write and execute code in the editor.
Expected Result: Code should execute and display output.
Actual Outcome: Code executed and output displayed.
Status: Passed

Test Case ID: TC-007

Test Case: Collaboration Feature
Description: Verify if multiple users can join and collaborate in the same workspace.
Expected Result: Users should be able to collaborate in real-time.
Actual Outcome: Collaboration works successfully.
Status: Passed

Test Case ID: TC-008

Test Case: DSA Questions Listing
Description: Verify if users can view the list of DSA questions.
Expected Result: Questions should be displayed properly.
Actual Outcome: Questions displayed successfully.
Status: Passed

Test Case ID: TC-009

Test Case: DSA Question Submission
Description: Verify if a user can submit a solution for a DSA question.
Expected Result: Solution should be submitted successfully.
Actual Outcome: Solution submitted successfully.
Status: Passed

Test Case ID: TC-010

Test Case: Real-time Compilation
Description: Verify if the code editor provides real-time compilation feedback.
Expected Result: Compilation feedback should be displayed in real-time.
Actual Outcome: Real-time compilation works correctly.
Status: Passed

Test Case ID: TC-011

Test Case: Cloudflare Protection
Description: Verify if Cloudflare is handling website security and performance optimization.
Expected Result: Website should be secured and optimized.
Actual Outcome: Security and optimization working as expected.
Status: Passed

Test Case ID: TC-012

Test Case: AI Assistance
Description: Verify if Google Gemini AI assists users in coding and resume creation.
Expected Result: AI assistance should function correctly.
Actual Outcome: AI assistance working correctly.
Status: Passed

Test Case ID: TC-013

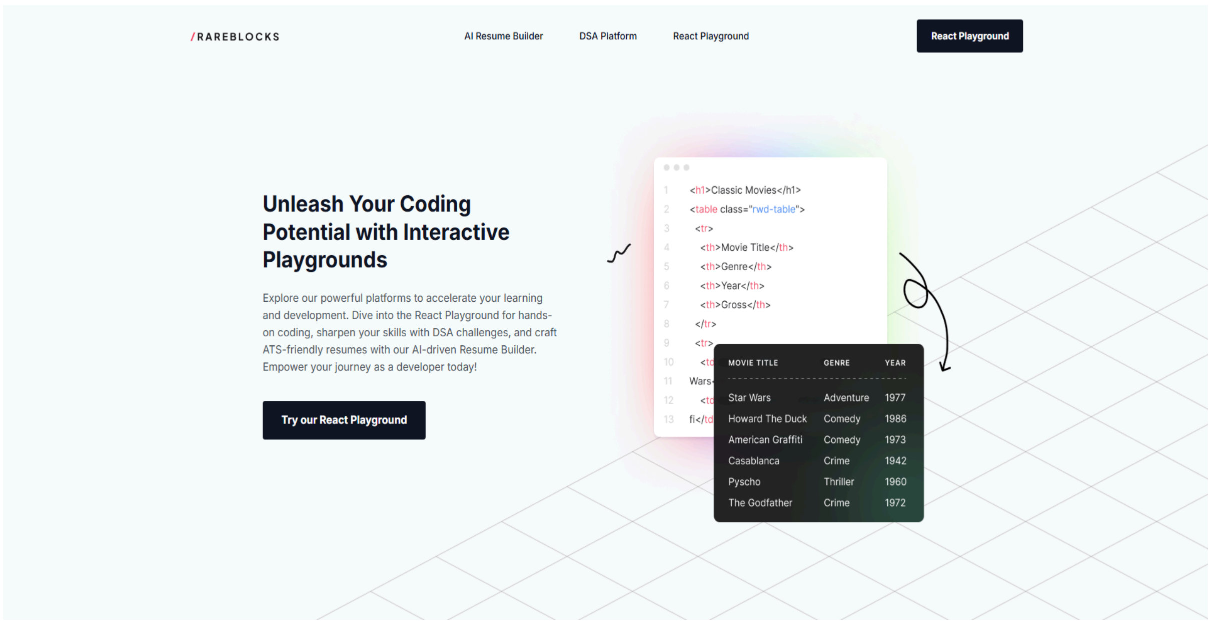
Test Case: API Response Time
Description: Verify if API responses are within acceptable limits.
Expected Result: API should respond in under 2 seconds.
Actual Outcome: API response time within limit.
Status: Passed

Test Case ID: TC-014

Test Case: UI Responsiveness
Description: Verify if the website layout is responsive across different devices.
Expected Result: Website should be responsive on mobile, tablet, and desktop.
Actual Outcome: Website is fully responsive.
Status: Passed

User Documentation:

1. Home Page



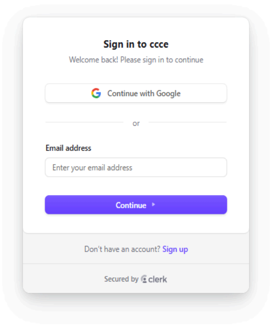
Unleash Your Coding Potential with Interactive Playgrounds

Explore our powerful platforms to accelerate your learning and development. Dive into the React Playground for hands-on coding, sharpen your skills with DSA challenges, and craft ATS-friendly resumes with our AI-driven Resume Builder. Empower your journey as a developer today!

```
<h1>Classic Movies</h1>
<table class="rwd-table">
 <tr>
  <th>Movie Title</th>
  <th>Genre</th>
  <th>Year</th>
  <th>Gross</th>
 </tr>
 <tr>
  <td
Wars<
  <td
fi</td
```

MOVIE TITLE	GENRE	YEAR
Star Wars	Adventure	1977
Howard The Duck	Comedy	1986
American Graffiti	Comedy	1973
Casablanca	Crime	1942
Pyscho	Thriller	1960
The Godfather	Crime	1972

2. Sign In Page



Sign in to ccce

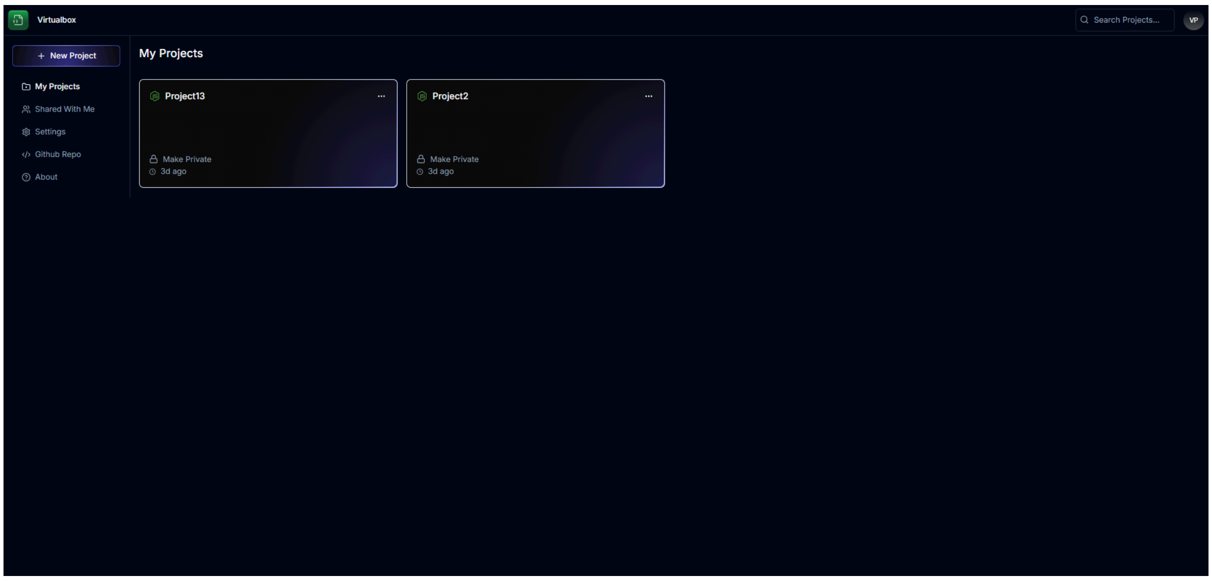
Welcome back! Please sign in to continue

Email address

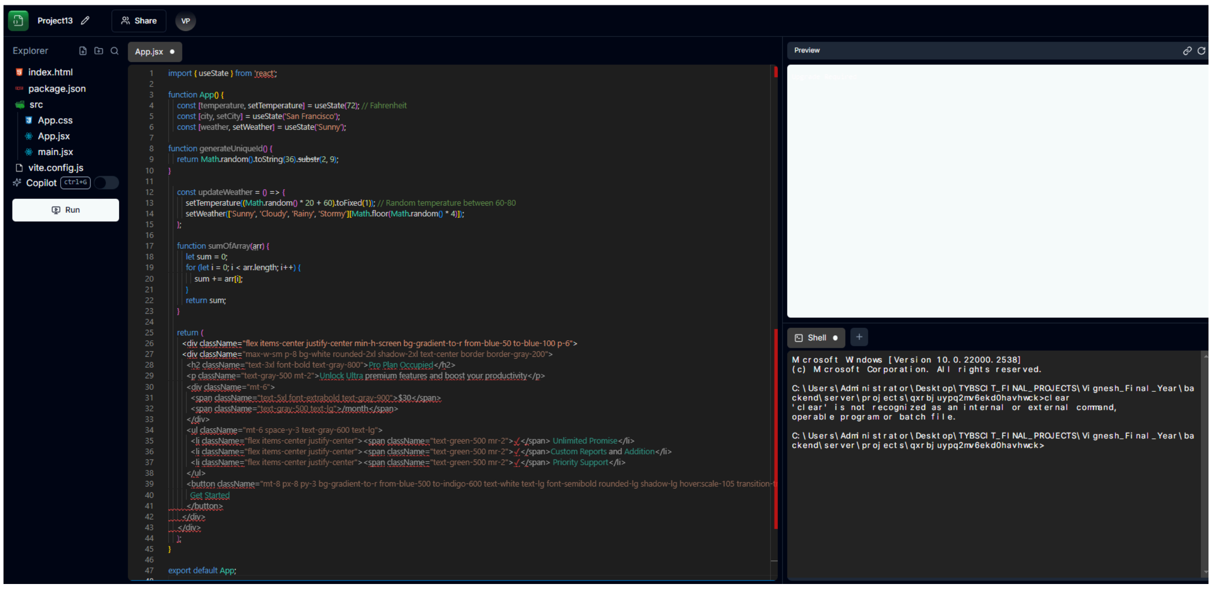
Enter your email address

Don't have an account? Sign up

3. Playground



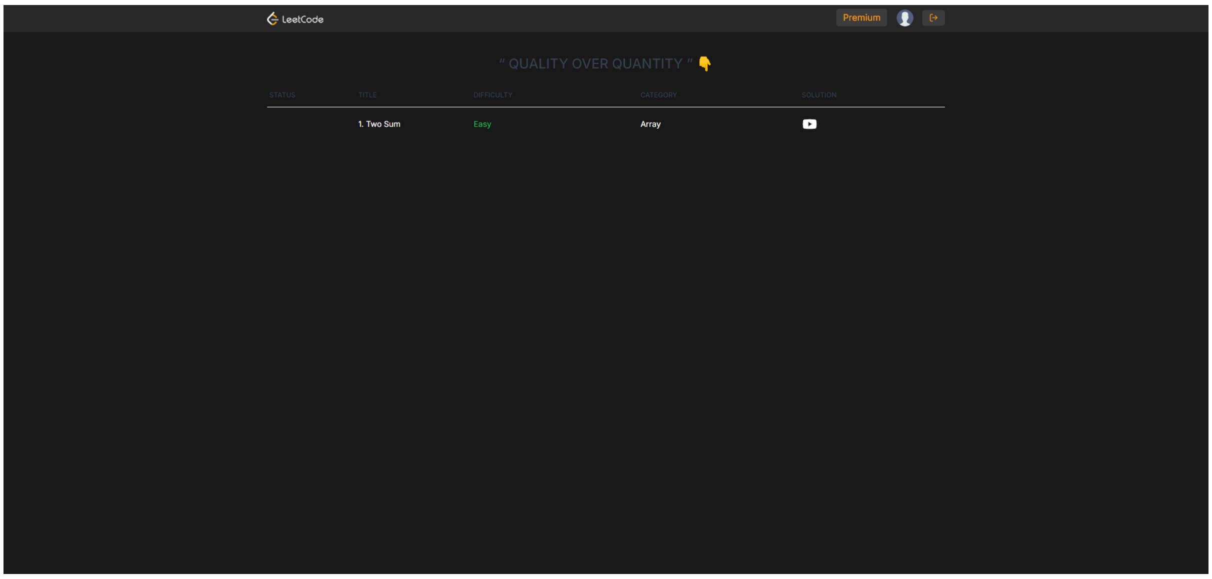
4. Single Project



```jsx
import { useState } from 'react';

function App() {
  const [temperature, setTemperature] = useState(72); // Fahrenheit
  const [city, setCity] = useState('San Francisco');
  const [weather, setWeather] = useState('Sunny');

function generateUniqueId() {
  return Math.random().toString(36).~~substr~~(2, 9);
}

  const updateWeather = () => {
    setTemperature((Math.random() * 20 + 60).toFixed(1)); // Random temperature between 60-80
    setWeather(['Sunny', 'Cloudy', 'Rainy', 'Stormy'][Math.floor(Math.random() * 4)]);
  };

  function sumOfArray(arr) {
    let sum = 0;
    for (let i = 0; i < arr.length; i++) {
      sum += arr[i];
    }
    return sum;
  }

  return (
   <div className="flex items-center justify-center min-h-screen bg-gradient-to-r from-blue-50 to-blue-100 p-6">
   <div className="max-w-sm p-8 bg-white rounded-2xl shadow-2xl text-center border border-gray-200">
    <h2 className="text-3xl font-bold text-gray-800">Pro Plan Occupied</h2>
    <p className="text-gray-500 mt-2">Unlock Ultra premium features and boost your productivity</p>
    <div className="mt-6">
     <span className="text-5xl font-extrabold text-gray-900">$30</span>
     <span className="text-gray-500 text-lg">/month</span>
    </div>
    <ul className="mt-6 space-y-3 text-gray-600 text-lg">
     <li className="flex items-center justify-center"><span className="text-green-500 mr-2">✓</span> Unlimited Promise</li>
     <li className="flex items-center justify-center"><span className="text-green-500 mr-2">✓</span>Custom Reports and Addition</li>
     <li className="flex items-center justify-center"><span className="text-green-500 mr-2">✓</span> Priority Support</li>
    </ul>
    <button className="mt-8 px-8 py-3 bg-gradient-to-r from-blue-500 to-indigo-600 text-white text-lg font-semibold rounded-lg shadow-lg hover:scale-105 transition-t
     Get Started
    </button>
   </div>
  </div>
  );
}

export default App;
```

```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop\TYBSCIT_FINAL_PROJECTS\Vignesh_Final_Year\ba
ckend\server\projects\qxrbjuypq2mv6ekd0havhwck>clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Administrator\Desktop\TYBSCIT_FINAL_PROJECTS\Vignesh_Final_Year\ba
ckend\server\projects\qxrbjuypq2mv6ekd0havhwck>
```

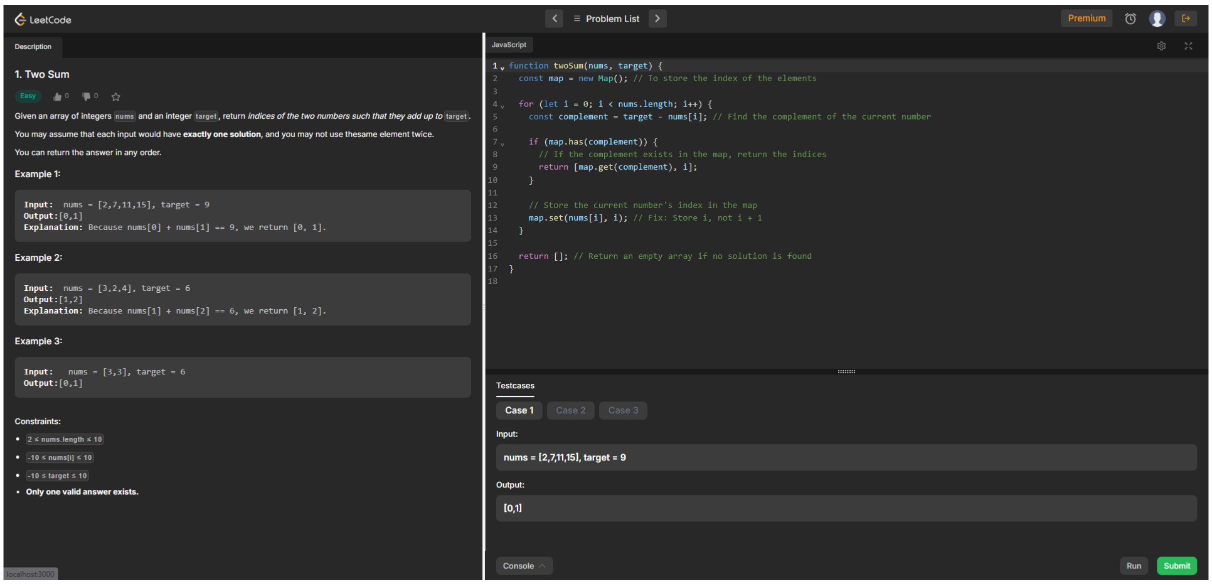
5. Leetcode Dashboard



" QUALITY OVER QUANTITY " 👇

STATUS	TITLE	DIFFICULTY	CATEGORY	SOLUTION
	1. Two Sum	Easy	Array	▶

6. Single Leetcode Question



1. Two Sum

Easy 0 0

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have **exactly one solution**, and you may not use thesame element twice.

You can return the answer in any order.

Example 1:

```
Input:  nums = [2,7,11,15], target = 9
Output:[0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Example 2:

```
Input:  nums = [3,2,4], target = 6
Output:[1,2]
Explanation: Because nums[1] + nums[2] == 6, we return [1, 2].
```

Example 3:

```
Input:   nums = [3,3], target = 6
Output:[0,1]
```

Constraints:

- `2 ≤ nums.length ≤ 10`
- `-10 ≤ nums[i] ≤ 10`
- `-10 ≤ target ≤ 10`
- **Only one valid answer exists.**

```javascript
function twoSum(nums, target) {
  const map = new Map(); // To store the index of the elements

  for (let i = 0; i < nums.length; i++) {
    const complement = target - nums[i]; // Find the complement of the current number

    if (map.has(complement)) {
      // If the complement exists in the map, return the indices
      return [map.get(complement), i];
    }

    // Store the current number's index in the map
    map.set(nums[i], i); // Fix: Store i, not i + 1
  }

  return []; // Return an empty array if no solution is found
}

```

Testcases

Case 1 Case 2 Case 3

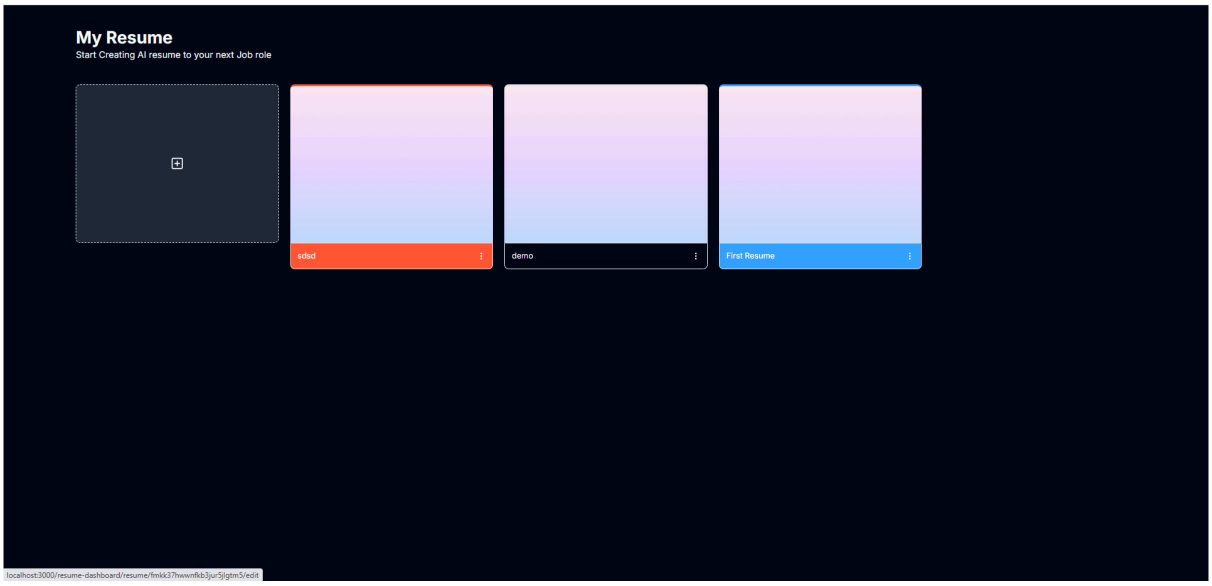
Input:

nums = [2,7,11,15], target = 9

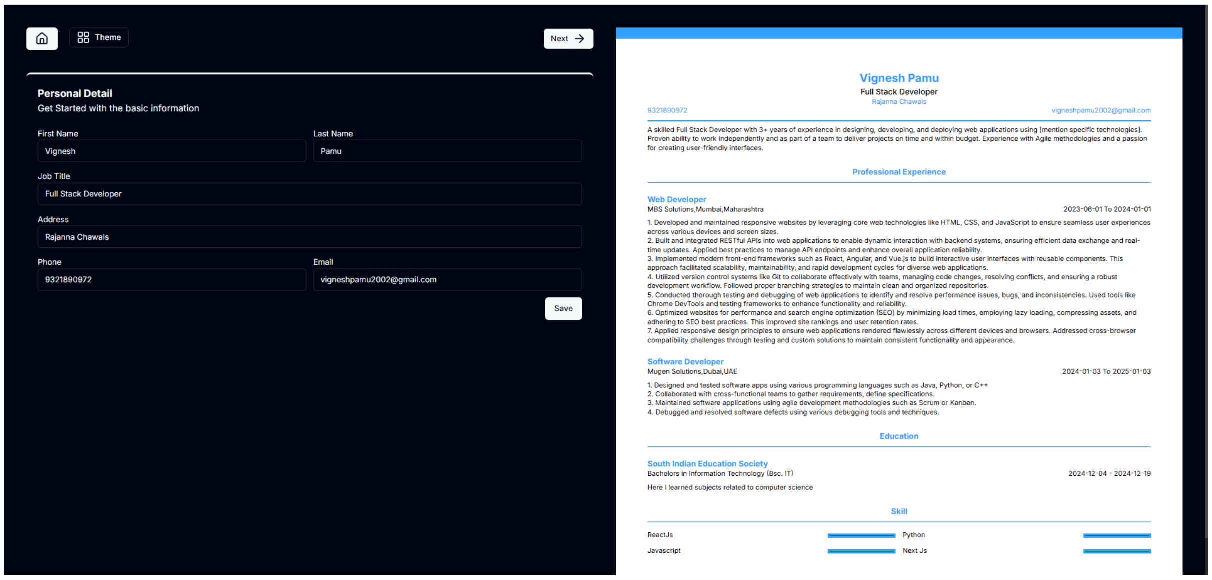
Output:

[0,1]

7. Resume Dashboard


My Resume
Start Creating AI resume to your next Job role
sdsd
demo
First Resume
localhost:3000/resume-dashboard/resume/fmkk37hwwnfkb3jur5jlgtm5/edit

8. Single Resume Edit



Personal Detail

Get Started with the basic information

First Name: Vignesh	Last Name: Pamu
Job Title: Full Stack Developer	
Address: Rajanna Chawals	
Phone: 9321890972	Email: vigneshpamu2002@gmail.com

Vignesh Pamu

Full Stack Developer

Rajanna Chawals

9321890972 vigneshpamu2002@gmail.com

A skilled Full Stack Developer with 3+ years of experience in designing, developing, and deploying web applications using [mention specific technologies]. Proven ability to work independently and as part of a team to deliver projects on time and within budget. Experience with Agile methodologies and a passion for creating user-friendly interfaces.

Professional Experience

Web Developer

MBS Solutions,Mumbai,Maharashtra 2023-06-01 To 2024-01-01

1. Developed and maintained responsive websites by leveraging core web technologies like HTML, CSS, and JavaScript to ensure seamless user experiences across various devices and screen sizes.
2. Built and integrated RESTful APIs into web applications to enable dynamic interaction with backend systems, ensuring efficient data exchange and real-time updates. Applied best practices to manage API endpoints and enhance overall application reliability.
3. Implemented modern front-end frameworks such as React, Angular, and Vue.js to build interactive user interfaces with reusable components. This approach facilitated scalability, maintainability, and rapid development cycles for diverse web applications.
4. Utilized version control systems like Git to collaborate effectively with teams, managing code changes, resolving conflicts, and ensuring a robust development workflow. Followed proper branching strategies to maintain clean and organized repositories.
5. Conducted thorough testing and debugging of web applications to identify and resolve performance issues, bugs, and inconsistencies. Used tools like Chrome DevTools and testing frameworks to enhance functionality and reliability.
6. Optimized websites for performance and search engine optimization (SEO) by minimizing load times, employing lazy loading, compressing assets, and adhering to SEO best practices. This improved site rankings and user retention rates.
7. Applied responsive design principles to ensure web applications rendered flawlessly across different devices and browsers. Addressed cross-browser compatibility challenges through testing and custom solutions to maintain consistent functionality and appearance.

Software Developer

Mugen Solutions,Dubai,UAE 2024-01-03 To 2025-01-03

1. Designed and tested software apps using various programming languages such as Java, Python, or C++
2. Collaborated with cross-functional teams to gather requirements, define specifications.
3. Maintained software applications using agile development methodologies such as Scrum or Kanban.
4. Debugged and resolved software defects using various debugging tools and techniques.

Education

South Indian Education Society

Bachelors in Information Technology (Bsc. IT) 2024-12-04 - 2024-12-19

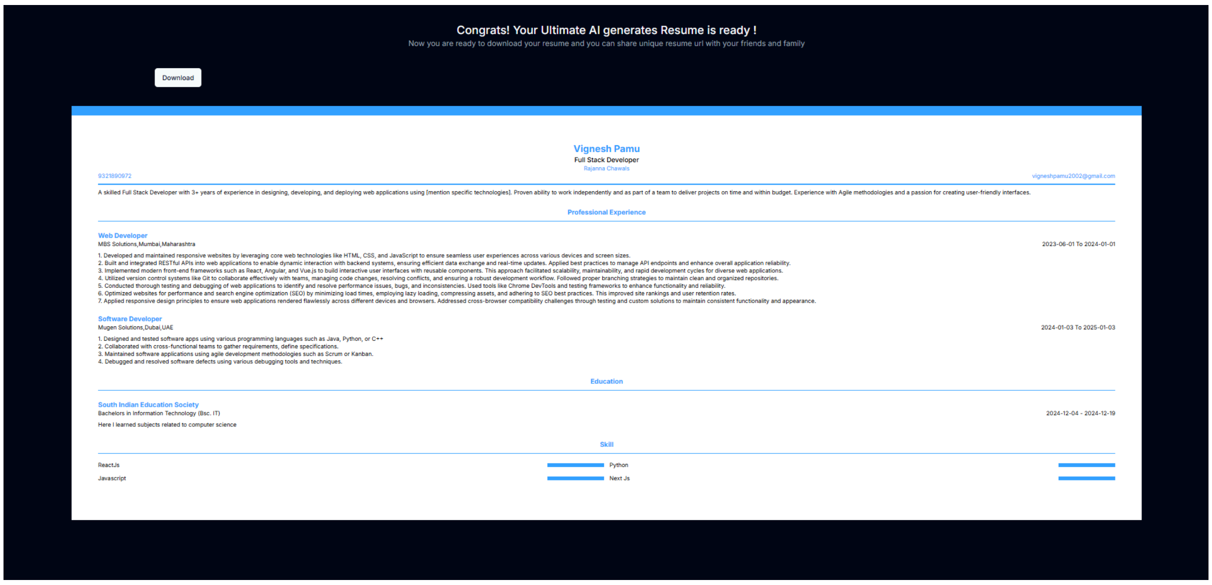
Here I learned subjects related to computer science

Skill

ReactJs Python

Javascript Next Js

9. Resume Download


Congrats! Your Ultimate AI generates Resume is ready !

Now you are ready to download your resume and you can share unique resume url with your friends and family

Vignesh Pamu

Full Stack Developer

Rajanna Chawals

9321890972 vigneshpamu2002@gmail.com

A skilled Full Stack Developer with 3+ years of experience in designing, developing, and deploying web applications using [mention specific technologies]. Proven ability to work independently and as part of a team to deliver projects on time and within budget. Experience with Agile methodologies and a passion for creating user-friendly interfaces.

Professional Experience

Web Developer

MBS Solutions,Mumbai,Maharashtra 2023-06-01 To 2024-01-01

1. Developed and maintained responsive websites by leveraging core web technologies like HTML, CSS, and JavaScript to ensure seamless user experiences across various devices and screen sizes.
2. Built and integrated RESTful APIs into web applications to enable dynamic interaction with backend systems, ensuring efficient data exchange and real-time updates. Applied best practices to manage API endpoints and enhance overall application reliability.
3. Implemented modern front-end frameworks such as React, Angular, and Vue.js to build interactive user interfaces with reusable components. This approach facilitated scalability, maintainability, and rapid development cycles for diverse web applications.
4. Utilized version control systems like Git to collaborate effectively with teams, managing code changes, resolving conflicts, and ensuring a robust development workflow. Followed proper branching strategies to maintain clean and organized repositories.
5. Conducted thorough testing and debugging of web applications to identify and resolve performance issues, bugs, and inconsistencies. Used tools like Chrome DevTools and testing frameworks to enhance functionality and reliability.
6. Optimized websites for performance and search engine optimization (SEO) by minimizing load times, employing lazy loading, compressing assets, and adhering to SEO best practices. This improved site rankings and user retention rates.
7. Applied responsive design principles to ensure web applications rendered flawlessly across different devices and browsers. Addressed cross-browser compatibility challenges through testing and custom solutions to maintain consistent functionality and appearance.

Software Developer

Mugen Solutions,Dubai,UAE 2024-01-03 To 2025-01-03

1. Designed and tested software apps using various programming languages such as Java, Python, or C++
2. Collaborated with cross-functional teams to gather requirements, define specifications.
3. Maintained software applications using agile development methodologies such as Scrum or Kanban.
4. Debugged and resolved software defects using various debugging tools and techniques.

Education

South Indian Education Society

Bachelors in Information Technology (Bsc. IT) 2024-12-04 - 2024-12-19

Here I learned subjects related to computer science

Skill

ReactJs Python

Javascript Next Js

Chapter 7 - CONCLUSIONS

Conclusions

This full-stack web application integrates essential features for students to enhance their coding skills, collaborate efficiently, and prepare job-ready resumes. By incorporating real-time collaboration, AI-powered suggestions, and an interactive problem-solving platform, the project provides a comprehensive learning experience.

Significance of the System

- **Skill Development:** Encourages students to enhance their coding, problem-solving, and resume-building skills.
- **Collaboration:** Facilitates real-time collaborative coding, making the learning experience more engaging and interactive.
- **Career Readiness:** Provides ATS-friendly resume templates to help students apply for jobs efficiently.
- **AI Integration:** Enhances user experience with AI-assisted code suggestions and problem-solving guidance.

Limitations of the System

- **High Resource Consumption:** Real-time collaboration and AI-driven functionalities may require significant computational resources.
- **Internet Dependency:** The system relies on a stable internet connection for seamless real-time collaboration and cloud services.
- **Limited Language Support:** Currently focuses on React and Node.js for coding, which may restrict users interested in other technologies.
- **AI Limitations:** The accuracy and effectiveness of AI-generated suggestions depend on the underlying model's capabilities.

Future Scope of the Project

- **Support for Additional Programming Languages:** Expanding beyond React and Node.js to include languages like Python, Java, and C++.

- **Enhanced AI Capabilities:** Improving AI-driven code suggestions, debugging, and resume recommendations.
- **Mobile Application Development:** Creating a mobile-friendly version to enhance accessibility.
- **Gamification & Leaderboard:** Introducing competitive coding elements like leaderboards, badges, and achievements.
- **Integration with Job Portals:** Connecting the resume builder with job portals to streamline job applications.

This project aims to bridge the gap between learning, collaboration, and career development, making it a valuable tool for students and aspiring software developers.