



PES UNIVERSITY, Bengaluru
Department of Computer Science and Engineering
B. Tech (CSE) – 5th Semester – Aug-Dec 2023

B.TECH. (CSE)
V Semester
UE21CS341A –Software
Engineering

PROJECT REPORT
on

CODEX - ONLINE CODE EDITOR

Submitted by :

Y MANISH KUMAR	PES2UG21CS617
YASHWANTH RAO M P	PES2UG21CSS623

ARBAZ KHAN N R	PES2UG21CS801
RACHAN REDDY	PES2UG22CS901

TEAM : 11

August – Dec 2023
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BENGALURU – 5G0100, KARNATAKA, INDIA

Table of Contents

Sl. No	Topic	Page No.
1.	Project Proposal	3 - 4
2.	Project Plan	5 - 8
3.	Software Requirements Specification	9 - 30
4.	Design Diagrams	31 - 34
5.	Test Plan, Cases	35 - 38
6.	Screenshots of Test Output	39-42

PROJECT PROPOSAL

In today's tech-driven world, remote collaboration and coding have become increasingly important. Our project, the "CodeX-Online Code Editor," aims to create a powerful and user-friendly online platform that enables developers to write, edit, and collaborate on code seamlessly.

Project Overview:

Objective:

The primary objective of our project is to develop and deploy a comprehensive Online Code Editor that caters to the needs of developers, both solo and collaborative, with a focus on ease of use, real-time collaboration, and code quality.

Features and Functionality:

Code Editing: CodeHub will provide a user-friendly code editor with syntax highlighting, auto-indentation, and code completion features.

Real-time Collaboration: Developers can collaborate in real-time on the same code file, with instant updates and chat functionality.

Version Control: Integration with Git for version control, allowing developers to commit, pull, and push code directly from the editor.

Code Execution: A built-in code execution environment for testing and debugging code without leaving the editor.

Code Sharing: Easy sharing of code files with others through shareable links or by inviting collaborators.

User Authentication: Secure user authentication and authorization mechanisms to protect user data and code.

Customizable Themes: Users can personalize the editor's appearance with customizable themes and layouts.

Error Detection: Real-time error detection and suggestions to improve code quality and prevent bugs.

Technologies:

We will leverage modern web technologies, including HTML5, CSS3, JavaScript, and frameworks like React for the front-end. For the back-end, we will use Node.js, Express.js, and a database system like MongoDB. Real-time features will be implemented using WebSockets. Security measures, such as encryption and user authentication, will be a priority.

User-Friendly:

A user-centric design approach will guide our system's development, making it intuitive and efficient for both novice and experienced developers.

Scalability:

CodeHub will be designed for scalability, allowing it to handle a growing number of users and code files as the platform gains popularity.

Benefits:

Facilitates remote collaboration among developers. Enhances

code quality through real-time error detection. Streamlines

version control and code sharing.

Reduces development time with built-in execution and debugging tools.

Project Timeline:

Our project plan includes several phases, including requirements gathering, design, development, testing, deployment, and continuous improvement. The estimated project duration is approximately [duration].

Conclusion:

The "CodeHub - Online Code Editor" project aims to provide developers with a powerful, collaborative, and user-friendly coding environment. We are committed to delivering a platform that empowers developers to code more efficiently, collaborate seamlessly, and improve code quality, thereby contributing to the software development community.

PROJECT PLAN

Project Plan Document

1: Identify the lifecycle to be followed for the execution of your project and justify why you have chosen the model.

The identified model is Waterfall model. Justification:

Clear Requirements:

The Waterfall model is best suited when project requirements are well-defined and unlikely to change significantly.

Risk Management:

The Waterfall model helps identify and address potential risks early in the project lifecycle. This is crucial in a financial platform where security and data integrity are of great importance.

Document-Driven:

Waterfall emphasizes documentation at each stage, making it easier to maintain a record of project decisions, requirements, and design choices.

Client and Stakeholder Involvement:

Waterfall encourages client/stakeholder involvement in the early stages to ensure their expectations are met.

Quality Assurance:

The testing phase in Waterfall ensures that the website is thoroughly tested before deployment, reducing the risk of critical issues.

2: Identify the tools which u want to use it throughout the lifecycle like planning tool, design tool , version control, development tool, bug tracking, testing tool.

Planning Tool:

Trello: Trello is a user-friendly project management tool that can help you create and manage tasks, set priorities, and collaborate with team members Design tool:

Canvas: Canvas is a tool for designing user interfaces (UI) and user experiences (UX). Version

Control:

Git: Git is a widely used version control system that allows for collaborative development, code tracking, and easy branching and merging.

Development tools:

Visual Studio Code: Visual Studio Code (VS Code) is a popular code editor with a wide range of extensions that can enhance development productivity. It supports various programming languages and offers features like debugging and code navigation.

Bug Tracking:

Jira: Jira is a comprehensive issue and project tracking tool that can be used for bug tracking, task management, and project management.

Testing Tools:

Selenium: Selenium is an open-source testing framework for web applications. It allows you to automate browser interactions, ensuring the functionality of your online code editor.

3: Determine all the deliverables and categorise them as reuse/build components and justify the Same.

Project Requirements Document:

Category: Reusable

Justification: The project requirements document outlines the overall project scope, objectives, and functional requirements. It serves as a reference for future projects and can be reused to guide similar initiatives. Database Schema:

Category: Reusable

Justification: A well-designed database schema can serve as a foundation for future projects or enhancements to the website. Reusing it can save time and effort in database development. Source

Code:

Category: Reusable

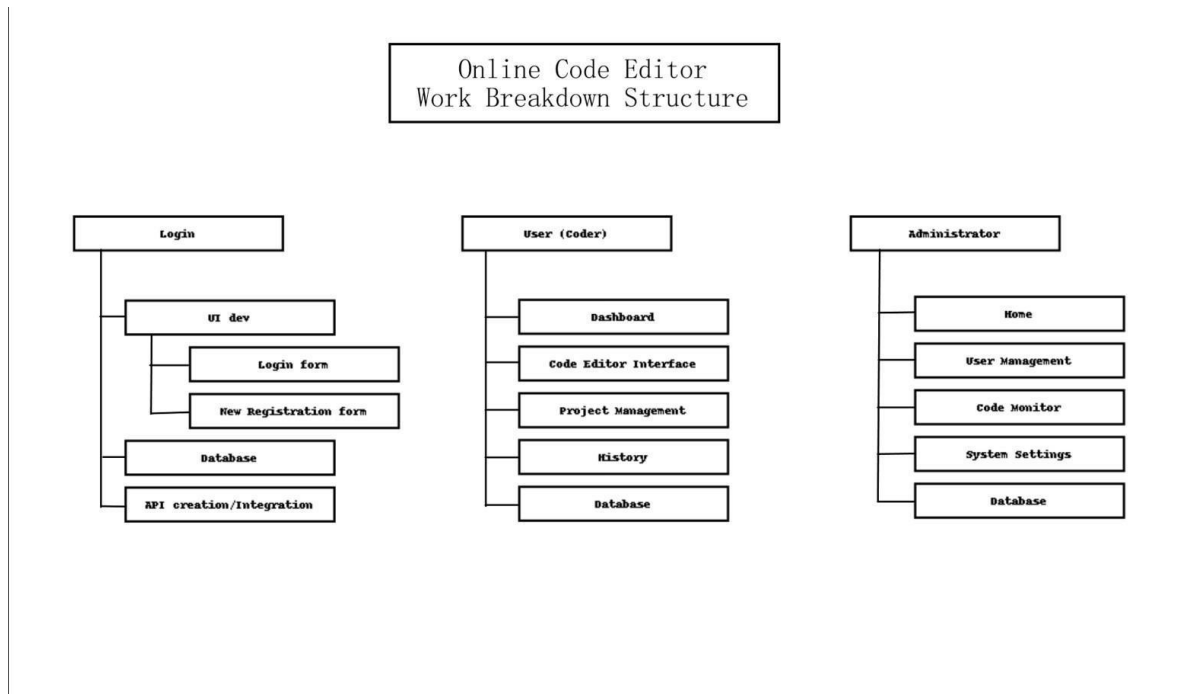
Justification: The source code contains reusable components such as coding patterns, libraries, and modules. These can be valuable for future projects, reducing development time and ensuring consistency.

Website Design Assets (Graphics, CSS Styles):

Category: Reusable

Justification: Design assets like logos, graphics, and CSS styles can be reused for branding and maintaining a consistent visual identity across multiple projects or website updates.

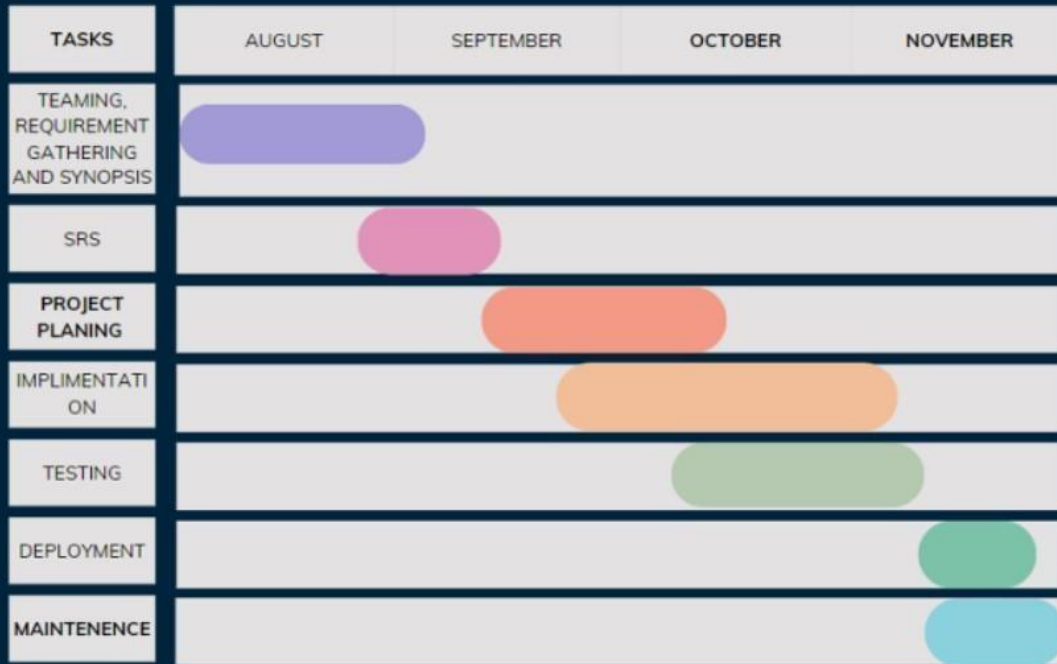
4: A WBS for the entire functionalities in detail.



5: A rough estimation of effort required to accomplish each task in terms of person months.

We are using Organic model , appox 3000 lines of code Effort = $2.4 * 3^{1.05} = 7.6$ person month 6: The Gantt Chart for scheduling .

GANTT CHART ONLINE CODE EDITIOR



Software Requirements Specification

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to outline the requirements and specifications for the development of the "Online Code Editor," a web-based platform designed to meet the specific needs of students, instructors, and researchers in the academic community.

1.2 Intended Audience and Reading Suggestions

Readers of all stripes can use the Online Code Editor SRS, including:

- **Developers:** To understand the technical requirements and implementation details of the Online Code Editor.
- **Project managers:** To gather insights into project scope, schedule, and resource allocation.
- **Testers:** To be aware of the testing standards, conditions, and anticipated results.
- **Students:** To understand the functionality of the Online Code Editor for collaborative coding.
- **Stakeholders:** To gain an overall understanding of the project.

1.3 Product Scope

The "Online Code Editor" is an innovative web-based platform tailored for the academic community. It simplifies code writing and collaboration processes, offers a user-friendly interface, supports assignment management, and provides an integrated library of learning resources.

The Online Code Editor project's main goals and objectives are as follows:

- **User-Friendly Interface:** Develop a simple and intuitive user interface that emulates the coding environment, making it accessible and easy to use for both novice and experienced programmers.
- **Safe Learning Environment:** Provide a secure platform where users can practice coding, collaborate, and experiment with code without the risk of data loss or privacy breaches.
- **Support for Diverse Coding Tasks:** Offer support for a wide range of coding tasks and programming languages, catering to the diverse needs of students, instructors, and researchers.

- **Enhanced Learning and Collaboration:** Facilitate collaborative coding and peer learning by enabling students to work together on assignments and projects, fostering a sense of teamwork.
- **Learning Resources Integration:** Integrate a library of learning resources, including tutorials, sample code, and documentation, to assist users in their learning journey.
- **Scalability:** Ensure that the platform can handle a large user base and maintain optimal performance even as it grows.
- **Privacy and Security:** Prioritize the security and privacy of user data and code, implementing robust measures to protect sensitive information.
- **Assessment and Feedback:** Enable instructors to create and manage coding assignments efficiently, streamlining the assessment and feedback process.
- **Positive Learning Environment:** Promote a positive and supportive learning environment that encourages exploration, creativity, and knowledge sharing among users.
- **Alignment with Educational Goals:** Ensure that the Online Code Editor aligns with the educational goals of the institution, supporting the enhancement of programming education and research.
- **Adaptation to Technological Trends:** Stay in sync with modern technology trends and advancements to provide users with the latest coding tools and features.
- **Accessibility:** Make the platform accessible to users with disabilities, ensuring that it complies with accessibility standards and guidelines.

1.4 References

Technical References

- MySQL Documentation: <https://dev.mysql.com/doc/>
- Python Documentation: <https://docs.python.org/3/>
- Java Documentation: <https://docs.oracle.com/en/java/>
- C++ Documentation: <https://en.cppreference.com/w/>
- JavaScript Documentation: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Educational Resources

- Codecademy: <https://www.codecademy.com/>
- Coursera: <https://www.coursera.org/>
- Udemy: <https://www.udemy.com/>
- edX: <https://www.edx.org/>

Brief Descriptions

- The MySQL documentation provides information on how to use the MySQL database management system, which could be used to store user data and code for the Online Code Editor.
- The documentation for the programming languages that will be supported by the Online Code Editor (Python, Java, C++, and JavaScript) provides information on how to use

each language's syntax, features, and libraries.

- The Codecademy, Coursera, Udemy, and edX websites provide online courses on a variety of programming topics, which could be used by students to learn about coding and the Online Code Editor.

Additional References

- The documentation for the web development frameworks and libraries that will be used to develop the Online Code Editor.
- The documentation for the testing frameworks and libraries that will be used to test the Online Code Editor.
- The documentation for the deployment and hosting environment that will be used to deploy and host the Online Code Editor.

2. Overall Description

2.1 Product Perspective

The Online Code Editor is a self-contained web-based application designed to provide a user-friendly and pedagogically oriented code editing and collaboration platform. It operates independently and does not rely on external systems.

The Online Code Editor offers the following benefits to its users:

- **Students:** The Online Code Editor provides a user-friendly environment for students to learn and practice coding. It offers syntax highlighting, linting, and other features to help students write code correctly. It also provides access to a library of learning resources, such as tutorials and sample code.
- **Instructors:** The Online Code Editor helps instructors to teach programming more efficiently. It allows them to create and manage coding assignments, assess student submissions, and provide feedback.
- **Researchers:** The Online Code Editor provides researchers with a powerful platform for writing and testing code for their research projects. It offers a variety of features to help researchers debug their code and track their progress.

2.2 Product Functions

The Online Code Editor provides the following functions:

- **User Registration and Authentication:** Users can create accounts and log in to the system.
- **Code Editing Interface:** Users can write and edit code in a variety of programming languages, including Python, Java, C/C++, and JavaScript. The code editing interface provides syntax highlighting, linting, and other features to help users write code correctly.
- **Collaboration Tools:** Users can collaborate on coding projects in real time. They can share code, communicate with each other, and comment on each other's code.

- **Assignment Management:** Instructors can create and manage coding assignments for their students. They can specify the assignment details, including the

programming language, deadline, and grading criteria. Students can submit their assignments to the system for grading.

- **Learning Resources:** Users can access a library of learning resources, such as tutorials and sample code, to help them learn about coding and the Online Code Editor.
- **Security and Privacy:** The system protects user data and code from unauthorized access and modification.
- **Scalability:** The system is designed to handle a large user base and a large number of concurrent coding sessions.
- **User Support and Feedback:** Users can access support and feedback resources through the system's website.

2.3 User Classes and Characteristics

The Online Code Editor is designed for use by the following user classes:

- **Students:** Students at all levels of experience can use the Online Code Editor to learn and practice coding. The system's user-friendly interface and integrated learning resources make it ideal for novice programmers.
- **Instructors:** Instructors can use the Online Code Editor to teach programming more efficiently. The system's assignment management and assessment features make it easy for instructors to create and manage coding assignments, assess student submissions, and provide feedback.
- **Researchers:** Researchers can use the Online Code Editor as a platform for writing and testing code for their research projects. The system's powerful code editing and debugging features make it ideal for researchers who need to write and test complex code.

2.4 Operating Environment

Hardware Platform:

The Online Code Editor is designed to operate on standard laptops and desktops. It has modest hardware requirements, including:

- Dual-core processor (1.5 GHz or higher is recommended)
- 4 GB of RAM (8 GB is recommended)
- 100 MB of available disk space

Operating System:

The Online Code Editor is compatible with the following operating systems:

- Windows 10 and above (64-bit)
- macOS 10.12 and above

- Linux distributions based on Debian and Red Hat (e.g., Ubuntu 18.04 LTS, CentOS 7)

Software Components:

The Online Code Editor is self-contained and does not require additional software during development and testing. During deployment, the Python code will manage database creation and connections, eliminating the need for separate MySQL setup on deployed instances.

Network Requirements:

The Online Code Editor requires a stable internet connection to provide real-time updates and maintain seamless functionality.

2.5 Design and Implementation Constraints

The following design and implementation constraints were considered during the development of the Online Code Editor:

- **User-Friendly Interface:** The user interface was designed to be intuitive and easy to use, even for novice programmers. User feedback will be incorporated during development for continuous improvement.
- **Security:** Security measures were implemented to ensure code encryption during transfer and storage, instilling user confidence in using the application.
- **Database Technology:** The MySQL database was chosen for its scalability and performance. Database queries were optimized for responsiveness and seamless user experiences.
- **Development Environment:** The project was developed in Python, a widely used and well-supported programming language. This ensures adherence to coding standards and best practices to maintain software integrity.

2.6 Assumptions and Dependencies

The following assumptions and dependencies apply to the Online Code Editor:

- **Assumptions:**
 - Users have basic industry knowledge and expectations of a user-friendly application.
 - The software simulates standard coding operations with utmost functionality.
 - Development and testing prioritize user experience and functionality.
- **Dependencies:**
 - The software requires a stable internet connection to provide real-time updates and maintain seamless functionality.
 - May depend on third-party integrations.

3. External Interface Requirements

3.1 User Interfaces

- **Authentication and Security:** The user must be able to securely log in using methods such as two-factor authentication and single sign-on.
- **Main Menu:** A clear and intuitive main menu should provide options for common coding tasks, such as creating a new project, opening an existing project, running code, debugging code, viewing code test results, and accessing learning resources.
- **Code Editor:** The code editor interface should provide features such as syntax highlighting, code completion, version control, code folding, code refactoring, and code linting.
- **Collaboration Tools:** Users should be able to collaborate with others on coding projects, with features such as real-time editing, commenting, and chat.
- **Assignment Management:** Instructors should have a dedicated interface for creating and managing coding assignments, including setting deadlines, assessing submissions, and providing feedback to students.
- **Learning Resources:** Users should be able to access a library of learning resources, including tutorials, sample code, and documentation for different programming languages.
- **User Feedback:** The system should provide a feedback mechanism for users to report issues or suggest improvements.

3.2 Software Interfaces

- **Database:** MySQL 8.0 or higher - The Online Code Editor's backend communicates with the database to retrieve user account information, code history, and to update code changes. This connection is essential for accessing and updating user code data. The MySQL database must be configured to allow remote connections.
- **Operating System:** Windows 10 and above (64-bit), macOS 10.12 and above, Linux distributions based on Debian and Red Hat (e.g., Ubuntu 18.04 LTS, CentOS 7) - The IDE used for developing the Online Code Editor application interacts with the operating system to carry out a range of tasks, including code storage, processing user input, as well as displaying output in the code editor.
- **Time module:** When there is a period of inactivity, the time module is used to automatically initiate a user logout from their account.
- **MySQL Connector:** To interact with the MySQL database from the Online Code Editor application, the MySQL Connector/Python library is used.

3.3 Communications Interfaces

- The primary protocol used for communication between the Online Code Editor's backend and the MySQL database server is the MySQL protocol, which typically operates over TCP/IP. The system uses port 3306 to communicate with the MySQL database.
- The Online Code Editor also uses HTTP to communicate between the backend and

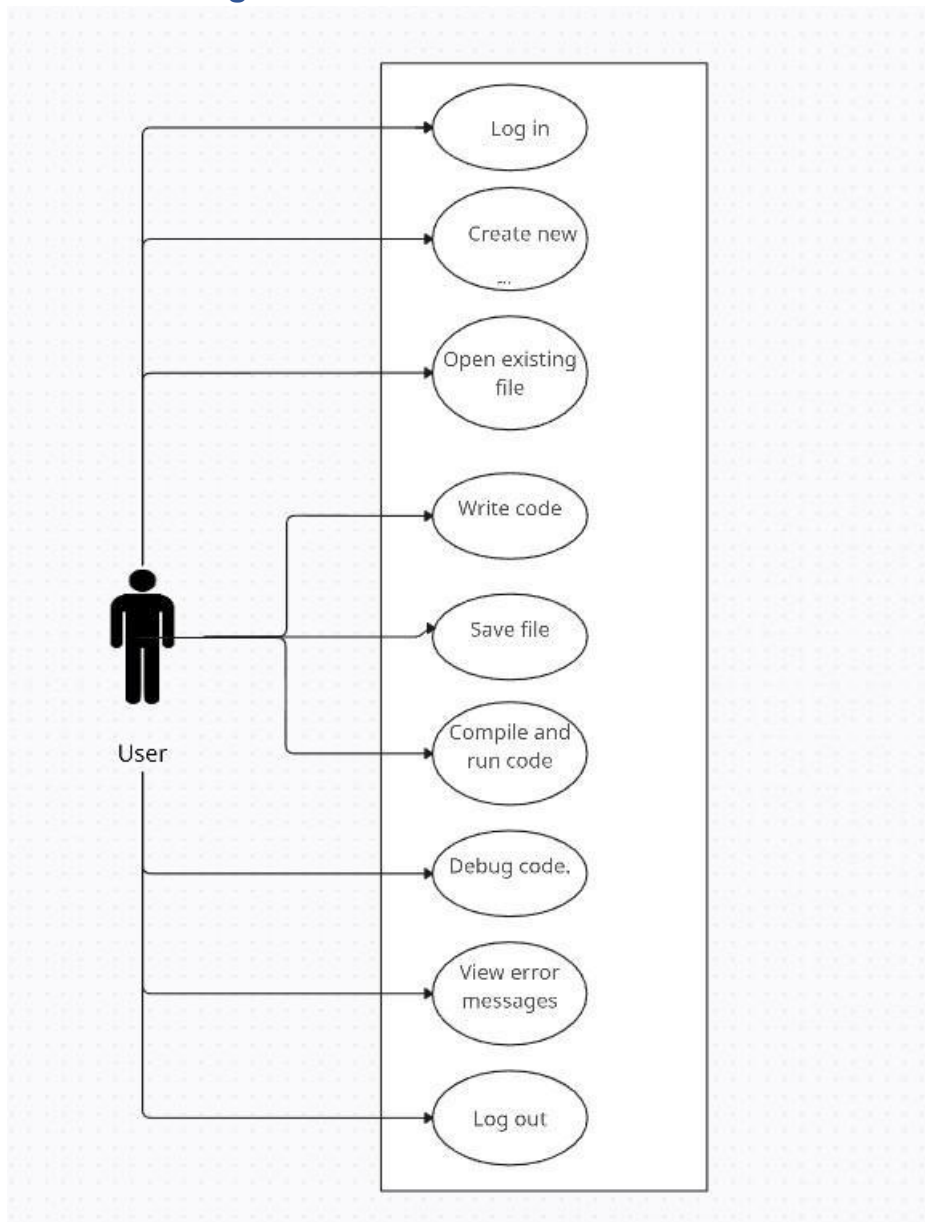
the user interface.

4. Analysis Models

The following four types of analysis models will be used to extract knowledge from data and help stakeholders better understand and improve the Online Code Editor system:

- **Descriptive analytics:** Descriptive analytics will be used to collect, clean, and summarize data about the Online Code Editor system, such as the number of active users, projects created, lines of code written, code errors detected, and user satisfaction ratings. Descriptive analytics will be used to generate reports and dashboards that will help stakeholders understand how the system is being used and identify areas for improvement.
- **Diagnostic analytics:** Diagnostic analytics will be used to drill down into data to identify the root causes of problems or trends in the Online Code Editor system. For example, diagnostic analytics could be used to identify why users are abandoning their projects, why certain types of code errors are more common than others, and which features of the system are most and least used. Diagnostic analytics will be used to identify and address problems with the Online Code Editor system so that it can be improved for users.
- **Predictive analytics:** Predictive analytics will be used to forecast future events or outcomes related to the Online Code Editor system. For example, predictive analytics could be used to forecast how many new users will sign up for the system next month, how many projects will be created next year, how many lines of code will be written next quarter, and which users are at risk of churning. Predictive analytics will be used to make better decisions about the development and marketing of the Online Code Editor system.
- **Prescriptive analytics:** Prescriptive analytics will be used to recommend actions that can be taken to improve the Online Code Editor system. For example, prescriptive analytics could be used to recommend which features of the system should be prioritized for development, which marketing campaigns are most likely to attract new users, and how the system can be improved to reduce user churn. Prescriptive analytics will be used to make the Online Code Editor system more user- friendly and effective.

Use Case Diagram



5. System Features

5.1 User Registration and Authentication

Description and Priority

This feature allows users to register for the Online Code Editor platform and authenticate their identity securely.

Stimulus/Response Sequences

1. User accesses the Online Code Editor website.
2. User selects the "Register" option.
3. User provides personal information, including a valid email address, and selects a username and password.
4. The system validates the information and creates a user account.

5. The system sends a verification email to the user's email address.
6. User clicks the verification link in the email to activate their account.
7. User logs in using the registered username and password.
8. User has the option to reset their password if they forget it.

Functional Requirements

- The system shall provide a user registration interface with input fields for personal information, including a valid email address.
- Users must select a unique username and password for their accounts.
- The system shall validate user input for correctness and ensure that usernames are unique.
- Upon successful registration, the system shall create a user account and store user data securely.
- The system shall send a verification email to the user's email address.
- Users must validate their email address to activate their accounts.
- The system shall allow users to reset their passwords if they forget them.

5.2 Code Editing Interface

Description and Priority

This feature provides users with an intuitive and feature-rich code editing environment.

Stimulus/Response Sequences

1. User logs in to the Online Code Editor.
2. User selects a coding project or creates a new one.
3. The code editor interface loads with syntax highlighting and code completion.
4. User writes and edits code in the code editor.
5. The system auto-saves changes and provides version control.
6. User refactors code to improve its structure and design.

Functional Requirements

- The code editor shall provide syntax highlighting for various programming languages.
- Users shall have access to code completion and suggestions.
- Code changes shall be automatically saved at regular intervals.
- The system shall implement version control to track code changes and revisions.
- The code editor shall provide code refactoring capabilities.

5.3 Collaboration Tools

Description and Priority

Collaboration tools enable users to work together on coding projects, providing features like real-time editing, commenting, and communication.

Stimulus/Response Sequences

1. User opens a coding project in the Online Code Editor.
2. User invites collaborators by sharing a project link.
3. Collaborators join the project and can simultaneously edit code.
4. Users and collaborators can leave comments and annotations on code lines.
5. The system tracks changes and updates in real-time.
6. Users and collaborators can communicate with each other in real-time through chat or video conferencing.
7. The system notifies users when their collaborators make changes to a project.
8. Users can create and manage private projects.
9. Users can revoke collaborator access to projects.

Functional Requirements

- Users shall have the ability to invite and revoke collaborator access to their coding projects.
- Collaborators must be able to join the project and edit code simultaneously.
- Users and collaborators can leave comments and annotations on code lines.
- The system shall provide real-time updates and highlight changes made by collaborators.
- Collaborators can see the list of users currently working on the project.
- Users and collaborators can communicate with each other in real-time through chat or video conferencing.
- The system shall notify users when their collaborators make changes to a project.
- Users can create and manage private projects.

5.4 Assignment Management

Description and Priority

Instructors can create and manage coding assignments, set deadlines, and assess student submissions efficiently.

Stimulus/Response Sequences

1. Instructor logs in to the Online Code Editor.
2. Instructor selects the "Create Assignment" option.
3. Instructor specifies assignment details, including title, description, deadline, and assessment criteria.

4. Students access the assignment, complete coding tasks, and submit their work.

5. Instructor assesses student submissions and provides feedback.

Functional Requirements

- Instructors shall have a dedicated interface to create coding assignments.
- Instructors can set assignment titles, descriptions, deadlines, and assessment criteria.
- The system shall allow instructors to define assessment criteria and requirements for submissions.
- Students shall access assignments, view instructions, complete coding tasks, and submit their work.
- Instructors must be able to assess student submissions and provide feedback.

5.5 Learning Resources

Description and Priority

Learning resources include tutorials, sample code, and interactive documentation to assist users in their coding journey. These resources can help users learn new programming concepts, brush up on existing skills, and troubleshoot problems.

Stimulus/Response Sequences

1. User accesses the Learning Resources section in the Online Code Editor.
2. User selects a category of learning resources, such as tutorials, sample code, or documentation.
3. User searches for a specific topic within the selected category.
4. The system displays a list of relevant learning resources, organized by topic and skill level.
5. User selects a learning resource and views it.

Functional Requirements

- The Online Code Editor shall provide a dedicated Learning Resources section.
- The Learning Resources section shall include tutorials, sample code, and documentation.
- Users shall be able to search for learning resources by topic or keyword.
- The system shall display a list of relevant learning resources, organized by topic and skill level.
- Users shall be able to view learning resources directly within the Online Code Editor.

Additional Requirements

- The learning resources should be comprehensive and up-to-date.
- The learning resources should be written in a clear and concise style.
- The learning resources should be accompanied by examples and illustrations to help

users understand the concepts.

- The interactive documentation should allow users to try out code snippets and see the results immediately.

5.6 Security and Privacy

Description and Priority

Security and privacy measures are critical to protect user data, code, and personal information.

Stimulus/Response Sequences

1. User logs in to the Online Code Editor.
2. User's data, including code projects and personal information, is encrypted during transfer and storage.
3. The system implements secure authentication and authorization mechanisms, including two-factor authentication (2FA).
4. Users can update their account settings, including changing passwords.
5. The system monitors and logs security-related activities for auditing.
6. Users have the option to enable 2FA.

Functional Requirements

- User data, including code and personal information, shall be encrypted during transfer and storage.
- The system shall implement secure authentication and authorization mechanisms, including two-factor authentication (2FA).
- Users must be able to update their account settings, including changing passwords.
- The system shall monitor and log security-related activities for auditing.
- Users have the option to enable 2FA.

5.7 Scalability

Description and Priority

Scalability ensures that the Online Code Editor can handle a large user base and maintain optimal performance as it grows.

Stimulus/Response Sequences

1. User registration and login processes should not significantly slow down, even with a large number of users.
2. Coding projects and collaborations should not experience performance degradation as the user base increases.
3. The system should be able to handle a growing number of assignments and submissions efficiently.

Functional Requirements

- The Online Code Editor should be designed for scalability, with performance

optimization measures in place.

- The system should implement load balancing and resource allocation to ensure optimal performance.
- Performance testing should be conducted regularly to identify and address scalability issues.
- The system should be designed to scale horizontally, allowing for the addition of more servers to handle increased traffic and load.

5.8 User Support and Feedback

Description and Priority

User support and feedback mechanisms are essential for assisting users and improving the platform based on user input.

Stimulus/Response Sequences

1. Users can access a "Help Center" or "Support" section for assistance.
2. The system shall provide documentation and FAQs to address common queries.
3. Users shall have the option to contact support through email, live chat, or a support ticket system.
4. Users can submit feedback, bug reports, and feature requests.
5. The system shall acknowledge and respond to user feedback and issues within a reasonable timeframe.
6. The system tracks user engagement metrics, such as the number of active users, the average time spent on the platform, and the most popular features.

Functional Requirements

- The Online Code Editor shall include a Help Center or Support section with documentation and FAQs.
- Users shall be able to contact support through email, live chat, or a support ticket system.
- The system shall maintain a record of user feedback, bug reports, and feature requests.
- Support staff must respond to user inquiries and issues within a reasonable timeframe.
- The system shall implement a feedback loop for continuous improvement.

6. Other Nonfunctional Requirements

The Online Code Editor platform shall meet the following other nonfunctional requirements:

6.1 Performance Requirements

- **Response Time:** The system shall respond to user input within 1 second under ideal operating conditions and within 2 seconds under normal operating conditions. First

response time shall be less than 1 second under ideal operating conditions and less

than 2 seconds under normal operating conditions. Average response time shall be less than 3 seconds under normal operating conditions.

- **Concurrent Users:** The system shall support a minimum of 1,000 concurrent user sessions without a significant decrease in responsiveness.
- **Transaction Throughput:** The system shall handle a minimum of 100 transactions per minute during peak usage times, with the following transaction types:
 - Code editing and saving
 - Code compilation and execution
 - Collaboration features (e.g., real-time editing, commenting, and communication)
 - Learning resources (e.g., tutorials, sample code, and documentation)
 - Maximum transaction throughput: 500 transactions per minute
 - The system shall be able to handle sustained peak transaction loads for at least 30 minutes without any significant degradation in performance.
- **Data Accuracy:** The system shall maintain a transaction accuracy rate of 99.9%, ensuring accurate balance updates and transaction histories. The system shall implement data reconciliation procedures to ensure that the data in the database is always consistent and accurate.
- **Fault Tolerance:** The system shall be able to recover from a single node failure without any loss of data or service.
- **Disaster Recovery:** The system shall be able to be recovered to a production state within 24 hours of a disaster event.

6.2 Safety Requirements

- **User Data Protection:** Sensitive user data, including PINs and account information, shall be encrypted during both transmission and storage.
- **Data Integrity:** Robust validation and verification mechanisms shall be implemented within the database to maintain data integrity.
- **Error Handling:** Clear and user-friendly error messages shall be provided to users in the event of software failures or system faults, without disclosing sensitive information.
- **Regular Data Backups:** The system shall implement regular database backups to prevent data loss in the event of a system failure. The retention period for the database backups shall be at least one week.
- **Disaster Recovery:** The system shall have a comprehensive disaster recovery plan in place to minimize data loss in the event of system failures or catastrophic events. The target RTO for the system is 4 hours, and the target RPO is 1 hour.

6.3 Security Requirements

- **User Authentication:** Users must provide a valid PIN or other secure authentication method to access the Online Code Editor.

- **Secure PIN Handling:** The system shall securely store and handle user PINs to prevent unauthorized access.
- **Audit Trail:** The Online Code Editor shall maintain a comprehensive log of all transactions and system activities for auditing purposes.
- **Unauthorized Access Prevention:** The system shall implement measures, such as account lockout, to prevent unauthorized access attempts and protect user accounts.
- **Strong Password Requirements:** The system shall enforce strong password requirements for user accounts, such as requiring passwords to be of a certain length and complexity.

6.4 Software Quality Attributes

- **Reliability:** The system shall maintain high uptime and minimal errors to ensure a reliable user experience. The system shall achieve a 99.9% uptime SLA.
- **Usability:** The system shall prioritize an intuitive and user-friendly interface to enhance usability.
- **Maintainability:** The software shall be designed for ease of updating and maintenance to support long-term sustainability.
- **Portability:** The Online Code Editor's software should be adaptable to different platforms, including web browsers and mobile devices.

6.5 Business Rules

- **Customer Authentication:** Access to the Online Code Editor shall be granted only to registered customers with valid accounts.
- **Transaction Limits:** The system shall define withdrawal and deposit limits per transaction and per day, differentiated by user type.
- **Transaction Logging:** All transactions shall be recorded for both customer and administrative purposes.
- **Login Restriction:** Access to the system shall be denied temporarily if an incorrect PIN is entered multiple times. The number of incorrect PIN entries allowed before the user's account is locked temporarily shall be configurable.
- **Compliance:** The Online Code Editor shall adhere to all relevant banking rules, standards, and data privacy regulations.

7. Other Requirements

7.1 Data Storage

The Online Code Editor platform shall meet the following data storage requirements:

- **Secure Database:** The system shall maintain a secure and reliable database to store customer account information, transaction history, and system logs.
- **Least Privilege:** The system shall implement a least privilege principle for database access, ensuring that only authorized users have access to the database and that they

only have access to the data that they need.

- **Data Integrity:** The system shall implement database constraints to ensure the consistency and accuracy of the data.
- **Data Replication:** The system shall replicate the database to multiple servers to improve availability and reliability.
- **Data Encryption:** All sensitive data stored in the database shall be encrypted at rest and in transit.

7.2 Backup and Recovery

The Online Code Editor platform shall meet the following backup and recovery requirements:

- **Regular Backups:** The system shall implement regular database backups to prevent data loss in the event of a system failure. The retention period for the database backups shall be at least one month.
- **Disaster Recovery:** The system shall have a comprehensive disaster recovery plan in place to minimize data loss in the event of system failures or catastrophic events. The target RTO for the system is 4 hours, and the target RPO is 1 hour.

7.3 Logging and Monitoring

The Online Code Editor platform shall meet the following logging and monitoring requirements:

- **Audit Trail:** The system shall maintain a comprehensive audit trail of all user activity and system events.
- **System Monitoring:** The system shall be monitored continuously for performance and security issues.

7.4 Security

The Online Code Editor platform shall meet the following security requirements:

- **Network Security:** The system shall be deployed in a secure network environment with appropriate firewalls and intrusion detection systems in place.
- **Application Security:** The system shall be developed and deployed using secure coding practices and security best practices.
- **Vulnerability Management:** The system shall be scanned for vulnerabilities regularly and all vulnerabilities shall be patched promptly.

7.5 Compliance

The Online Code Editor platform shall adhere to all relevant banking rules, standards, and data privacy regulations.

Appendix A: Glossary

Term	Definition
Access control	A system or process that regulates who has access to resources, such as files, databases, and networks.
Authentication	The process of verifying the identity of a user.
Authorization	The process of granting permission to users to access resources.
Code compilation	The process of converting source code into a machine-readable format.
Code execution	The process of running code on a computer.
Collaboration features	Features that allow users to work together on code, such as real-time editing, commenting, and communication.
Concurrent users	The number of users that can use a system at the same time.
Data integrity	The accuracy and consistency of data.
Data replication	The process of copying data to multiple servers to improve availability and reliability.

Data encryption	The process of converting data into a format that is unreadable without a decryption key.
------------------------	---

Disaster recovery	A plan for recovering from a system failure or catastrophic event.
Fault tolerance	The ability of a system to continue operating even if one or more components fail.
Least privilege	The principle of giving users only the access and permissions they need to perform their job duties.
Login restriction	A security measure that temporarily prevents users from accessing a system after a certain number of failed login attempts.
Performance	The responsiveness and throughput of a system.
Recovery point objective (RPO)	The maximum amount of data that can be lost in the event of a system failure or catastrophic event.
Recovery time objective (RTO)	The maximum amount of time that it can take to recover from a system failure or catastrophic event.
Secure database	A database that is protected from unauthorized access, modification, or destruction.

Strong password requirements	Requirements for passwords that make them difficult to crack, such as requiring a minimum length, complexity, and variety of characters.
Transaction accuracy	The percentage of transactions that are completed accurately.
Transaction throughput	The number of transactions that a system can process per unit time.

Usability	The ease with which a system can be used.
Vulnerability management	The process of identifying, assessing, and remediating vulnerabilities in a system.

Appendix B: Field Layouts

Field	Length	Data Type	Description	Is Mandatory
File Name	50	String	Name of the code file	Yes
File Type	20	Alphanumeric	Type of code file (e.g., Python, Java, C++)	Yes
Code Content	Unlimited	String	Content of the code file	Yes

User ID	10	Numeric	Unique identifier of the user creating the code file	Yes
---------	----	---------	--	-----

Code File Report

Field	Description
File Name	Name of the code file.
File Type	Type of code file (e.g., Python, Java, C++).

Code Content	Content of the code file.
User ID	Unique identifier of the user creating the code file.
Date Created	Date and time the code file was created.
Date Modified	Date and time the code file was last modified.

- Code File Report:
 - o Version Control System (VCS) Branch (if applicable)
 - o Code File Status (e.g., Draft, In Review, Approved)
 - o Number of Lines of Code
 - o Number of Errors
 - o Number of Warnings
- User Account Table
 - o User ID
 - o First Name

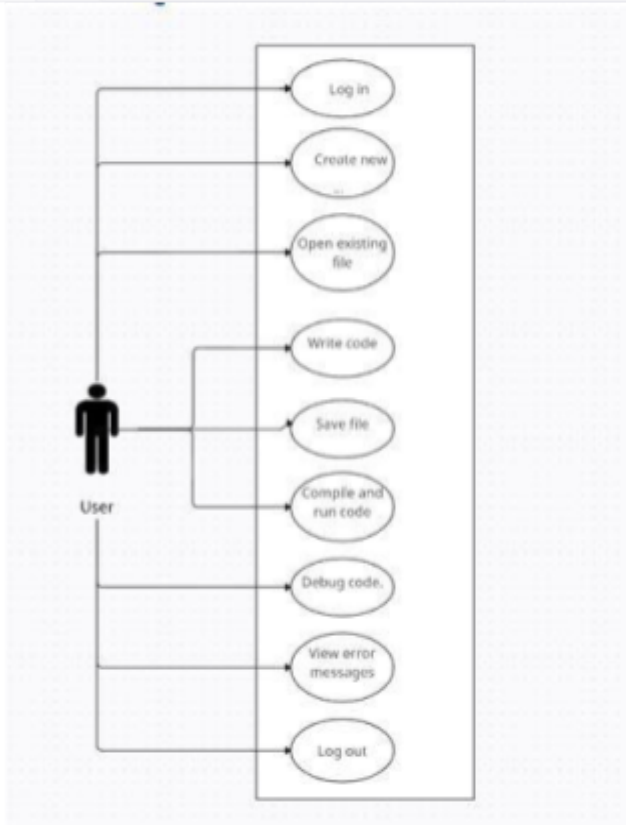
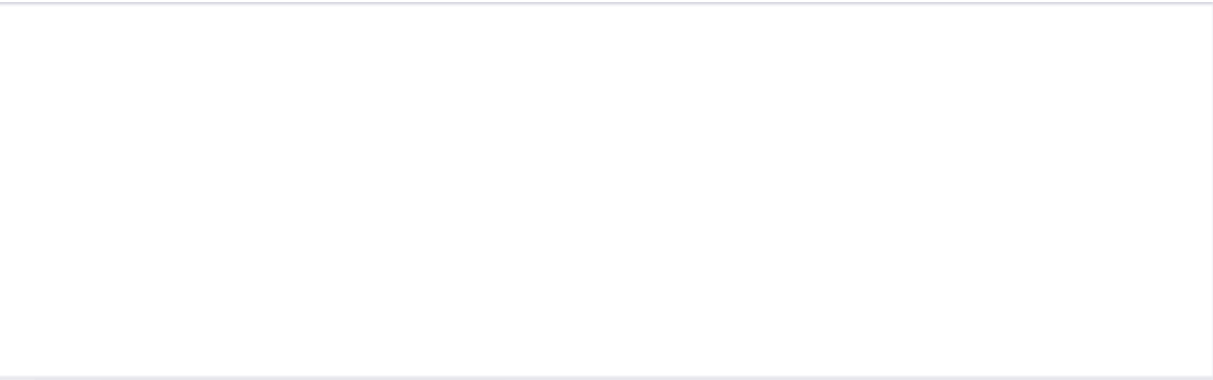
- o Last Name
 - o Email Address
 - o Password
 - o Account Type (e.g., Free, Paid)
 - o Subscription Status (if applicable)
- Collaboration Table
 - o Collaboration ID
 - o Code File ID
 - o User ID
 - o Role (e.g., Owner, Editor, Viewer)
- Code Execution Log Table
 - o Code Execution Log ID
 - o User ID
 - o Code File ID

- o Date and Time of Execution
- o Execution Status (e.g., Successful, Failed)
- o Output

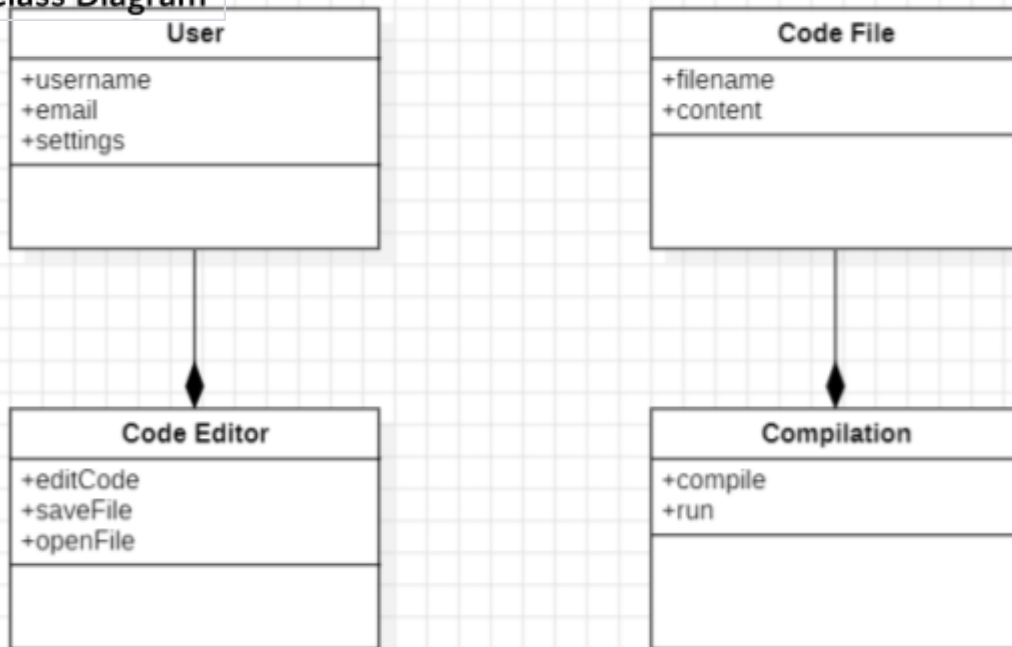
Appendix C: Requirement Traceability Matrix

Sl. No	Requirement ID	Brief Description of Requirement	Architecture Reference	Design Reference	Code File Reference	Test Case ID	System Test Case ID

DESIGN DOCUMENT

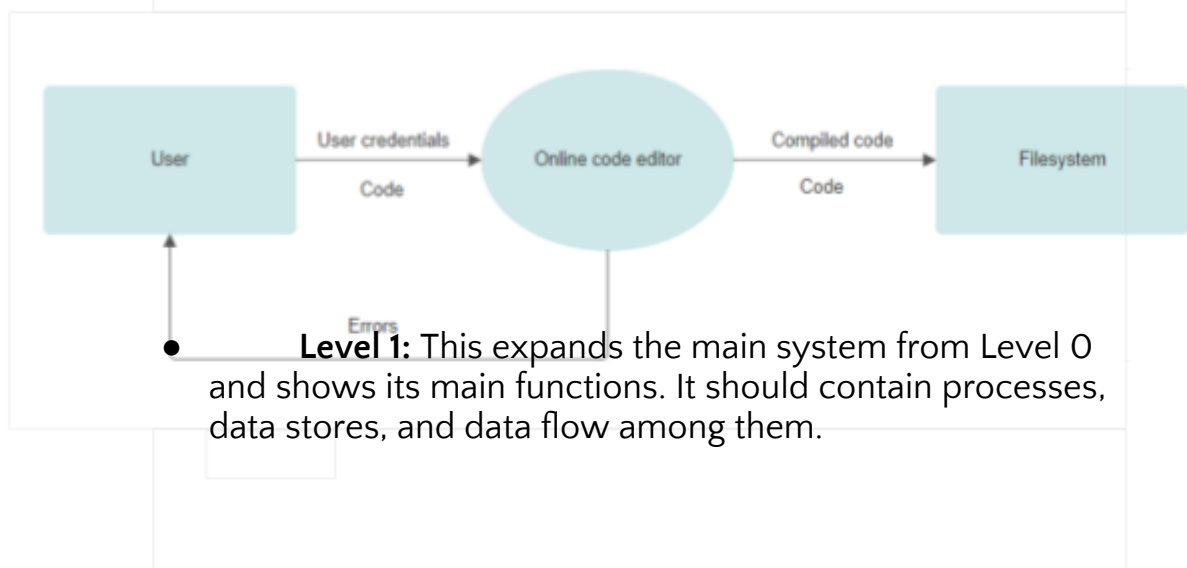


Class Diagram

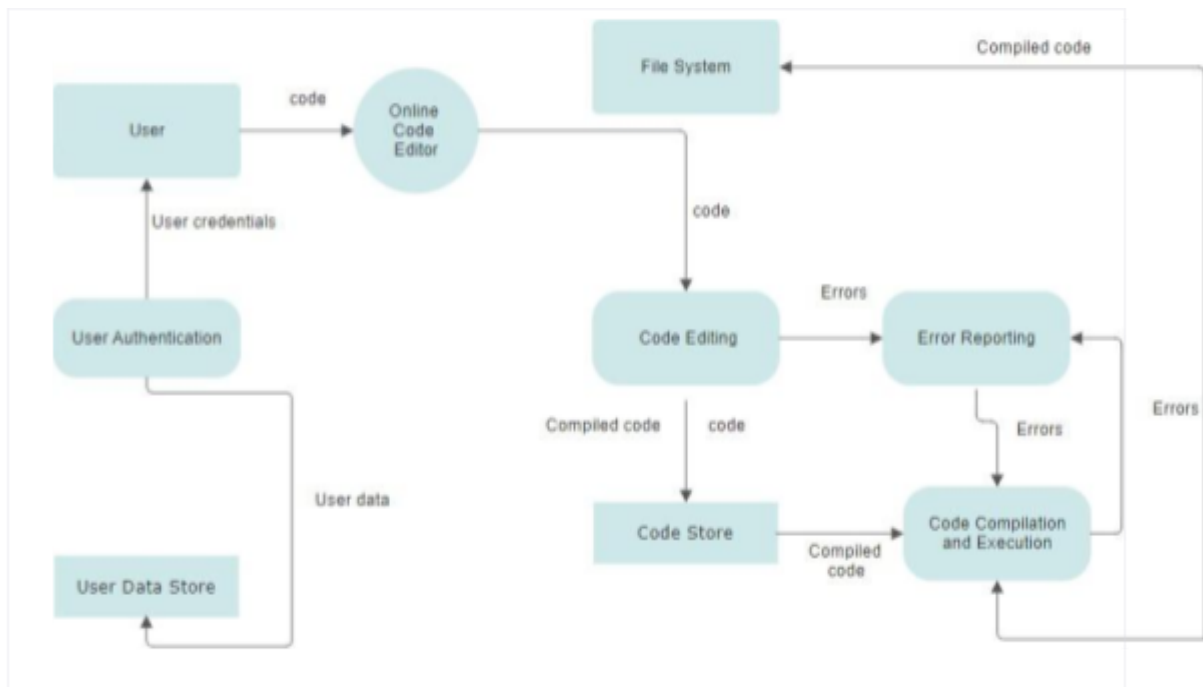


1. Incorporate DFDs:

- Develop a Data Flow Diagram (DFD) for your project:
 - Level 0 (Context Diagram):** This is the top level of the DFD, which provides a bird's eye view of the system. It should include external entities and how they interact with the main system.



- Level 1:** This expands the main system from Level 0 and shows its main functions. It should contain processes, data stores, and data flow among them.



2. Architectural Style Integration:

1. Modularity and Scalability:

- The Online Code Editor needs to accommodate users with varying levels of experience, from beginners to advanced coders.
- SOA allows breaking down the system into loosely coupled, independently deployable services.
- This modularity makes it easier to scale individual components, which is essential for handling a potentially large and diverse user base.

2. Maintainability:

- The platform should offer a user-friendly environment for learning and practice, and this requires frequent updates and improvements.
- SOA promotes reusability and maintainability by breaking the system into smaller, manageable services.
- This separation of concerns allows for easier updates and maintenance of individual services without disrupting the entire system.

3. Flexibility and Interoperability:

- The Online Code Editor might need to integrate with external learning resources and tools.
- SOA enables easy integration with external services and APIs.

- This facilitates access to learning resources, tutorials, and sample code by encapsulating external functionalities within well-defined service interfaces.

4. Performance:

- Collaborative coding and concurrent sessions can place varying demands on system resources.
- By distributing the workload across different services, you can optimize the performance of the system.
- Critical services, such as code editing and user authentication, can be deployed on high-performance servers, while allocating resources as needed for less critical services.

5. Security:

- Ensuring the security and privacy of user data and code is crucial for the Online Code Editor.
- SOA enables encapsulating security measures within individual services and controlling access through well-defined APIs.
- This enhances the security of the system, making it well-suited for protecting user data and code.

Justification:

The Online Code Editor is a complex web-based platform with various functionalities, including user registration, code editing, collaboration, assignment management, and more. SOA allows breaking down these functionalities into loosely coupled, independently deployable services, offering the flexibility to integrate external resources, optimize performance, and enhance security. This architectural style aligns with the requirements of the Online Code Editor and supports its scalability, maintainability, and performance needs.

TEST DOCUMENT

Test case ID	Name of Module	Test Case Description	Test Steps	Test Data	Expected Results By the tester	Actual Results	Test Result
1	Authentication :	To test login functionality	Enter non existent account details	Username:lefnaln@gmail.com Pass: 392932	Cannot login as account doesn't exist	Cannot login as account doesn't exist	pass
2	Authentication :	To test login functionality	Enter existent account details	Username yashmalathkar9@gmail.com Pass:yashwanth1234	Opens dashboard	Opens dashboard	pass
3	Authentication :	To test grammar	Don't enter @ in email	Username:ssdsfs Pass:123	Cannot sign up as email is invalid	Cannot sign up as email is invalid	pass

4	Authentication :	To test grammar	Enter a Non - identified dns	Username: snikis@gmail.com Pass: onion	Cannot sign up as dns is invalid	Cannot sign up as dns is invalid	pass
---	------------------	-----------------	------------------------------	---	----------------------------------	----------------------------------	------

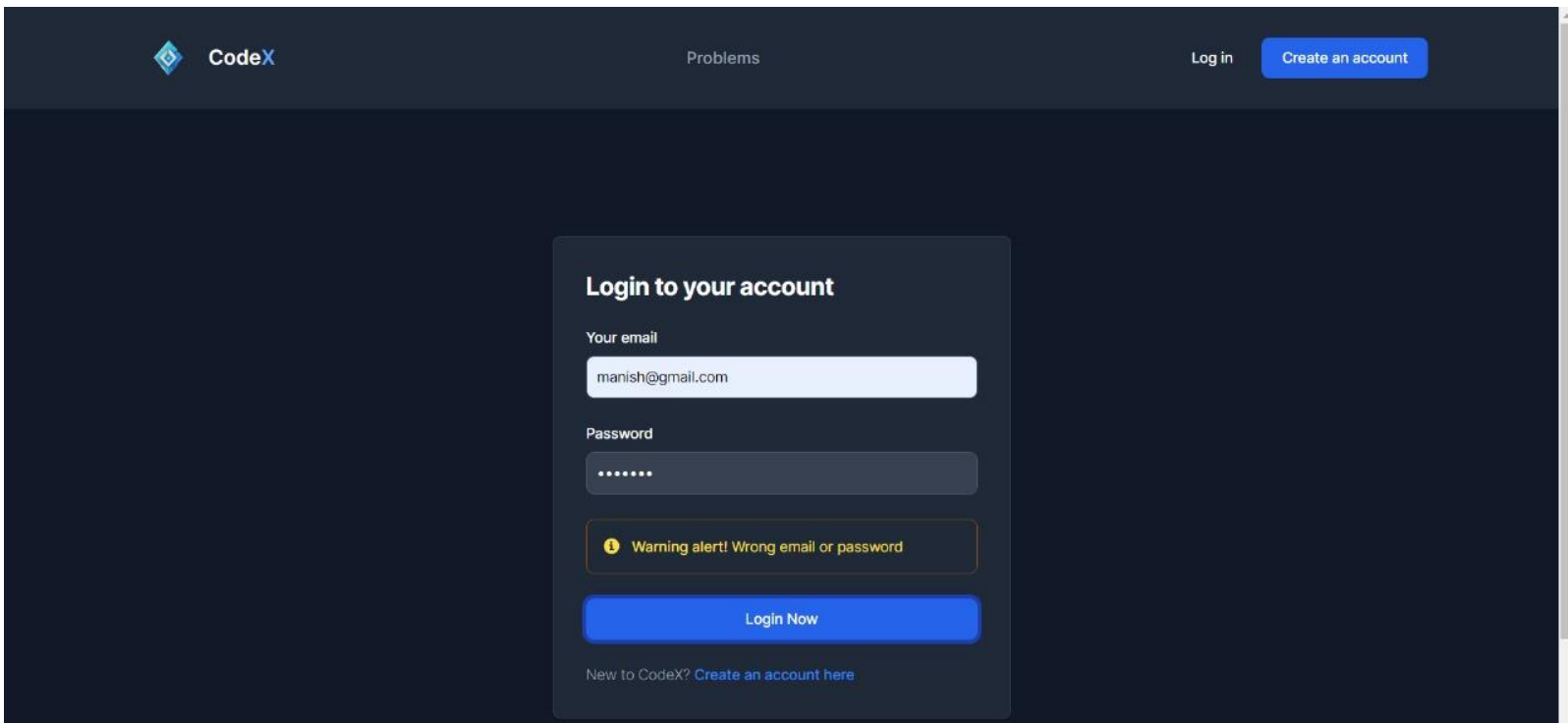
5	Authentication :	To test data matching	Enter a correct email with wrong password	Username: yashmalathkar9@gmail.com .Pass: wrong pass	Cannot sign in as account doesn't exist	Cannot sign in as account doesn't exist	pass
6.	Creating a file and folder	To test creating a file	Click on new button and then file	Enter file name	A file is created based on the give name	A file is created based on the given name	pass
7.	Creating a file and folder	To test creating a file	Click on new button and then folder	Enter folder name	A file is created based on the give name	A folder is created based on the given name	pass
8.	Programming language button	To test the change programming language button	Click change programming language button And select a programming language	Select one of the three: Python, Javascript, C++	The button should display the chosen language	The button should display the chosen language	pass

9.	Running the code	Chosen language : Javascript Code written:Python	Write the code in a language different from the chosen language	Write the code in a language different from the chosen language	Syntax error	Syntax error	pass
----	------------------	--	---	---	--------------	--------------	------

10.	Running the code	Chosen language and code are written in the python	Write the code in Python and chose Python	Click on the run button	Correct output is shown	Correct output is shown	pass
11.	Running the code	Chosen language and code are written in the C++	Write the code in C++ and Chose C++ as the language	Click on the run button	Correct output is shown	Correct output is shown	pass
12.	Running the code	Chosen language and code are written in the Javascript	Write the code in Javascript and Choose Javascript as the language	Click on the run button	Correct output is shown	Correct output is shown	pass
13.	Saving the File	To test whether the file is getting stored or not	Write the code and click on save button	The data is the code which is stored in the file	File is stored	File is stored	pass

14.	Locally stored data on server	To test whether the data is preserved locally	Write the code and logout	Again login using credentials	Data is not stored because if was not saved	Data is lost	pass
-----	-------------------------------	---	---------------------------	-------------------------------	---	--------------	------

Screenshots of Test Output



The screenshot shows the CodeX login interface. The header includes the CodeX logo, a 'Problems' link, and 'Log in' and 'Create an account' buttons. The main content area features a 'Login to your account' form with fields for 'Your email' (containing 'manish@gmail.com') and 'Password' (masked with dots). A yellow warning message states: 'Warning alert! Wrong email or password'. Below the form is a 'Login Now' button and a link to 'Create an account here' for new users.

CodeX

Problems

Log in

Create an account

Login to your account

Your email

manish@gmail.com

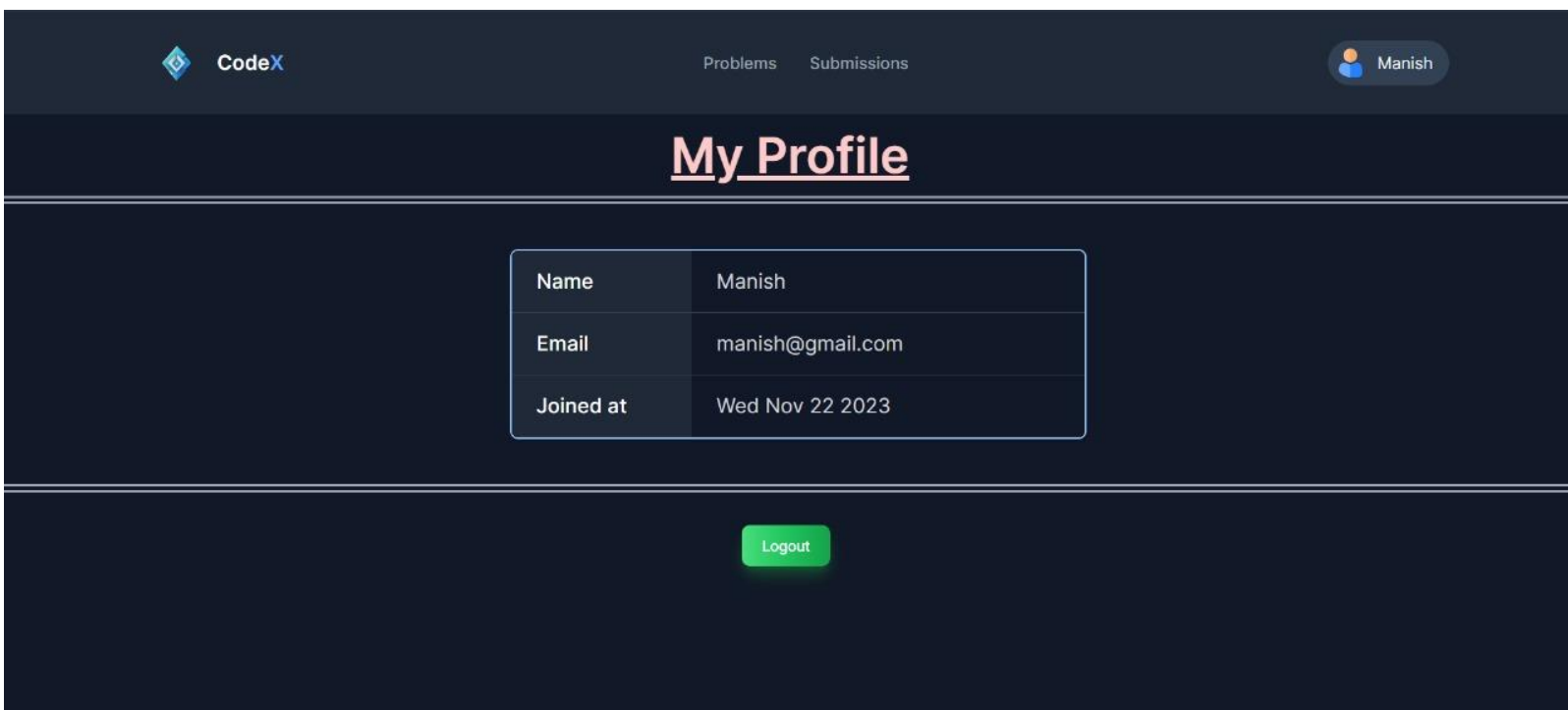
Password

.....

Warning alert! Wrong email or password

Login Now

New to CodeX? [Create an account here](#)



The screenshot shows the CodeX user profile page. The header includes the CodeX logo, 'Problems' and 'Submissions' links, and a user profile icon labeled 'Manish'. The main content area has a heading 'My Profile' and a table displaying user information. At the bottom, there is a 'Logout' button.

CodeX

Problems

Submissions

Manish

My Profile

Name	Manish
Email	manish@gmail.com
Joined at	Wed Nov 22 2023

Logout

2. Reverse String

✓ Accepted

```
function reverseString(s) { let left = 0; let right = s.length - 1; while (left < right) { // Swap characters at left and right indices [s[left], s[right]] = [s[right], s[left]]; // Move indices towards each other left++; right--; } }
```

[Go to Problem](#)



[Problems](#) [Submissions](#)

Manish

2. Reverse String

✗ Not accepted

```
#include <stdio.h> #include <string.h> void reverseString(char *s, int length) { int left = 0; int right = length - 1; while (left < right) { // Swap characters at left and right indices char temp = s[left]; s[left] = s[right]; s[right] = temp; // Move indices towards each other left++; right--; } }
```

2. Reverse String

✓ Accepted

```
#include <iostream> #include <vector> void reverseString(std::vector<char>& s) { int left = 0; int right = s.size() - 1; while (left < right) { // Swap characters at left and right indices  
std::swap(s[left], s[right]); // Move indices towards each other left++; right--; } }
```

[Go to Problem](#)

2. Reverse String

✓ Accepted

```
function reverseString(s) { let left = 0; let right = s.length - 1; while (left < right) { [s[left], s[right]] = [s[right], s[left]]; left++; right--; } } const inputString = ['h', 'e', 'l', 'l', 'o'];  
reverseString(inputString); console.log(inputString);
```

[Go to Problem](#)

2. Reverse String

✖ Not accepted

```
function reverseString(s) { let left = 0; let right = s.length - 1; while (left < right) { // Swap characters at left and right indices [s[left], s[right]] = [s[right], s[left]]; // Move indices towards each other left++; right--; } }
```

[Go to Problem](#)

STATUS	NUMBER	NAME	SUBMISSION DATE	ACTION
✖	1	Two Sum	Wed, 22 Nov 2023 05:26:53 GMT	View
✔	2	Reverse String	Wed, 22 Nov 2023 05:29:40 GMT	View
✔	2	Reverse String	Wed, 22 Nov 2023 13:20:41 GMT	View
✖	2	Reverse String	Wed, 22 Nov 2023 13:23:03 GMT	View
✔	2	Reverse String	Wed, 22 Nov 2023 13:23:29 GMT	View
✔	2	Reverse String	Wed, 22 Nov 2023 13:24:18 GMT	View
✖	2	Reverse String	Wed, 22 Nov 2023 13:33:39 GMT	View
✔	2	Reverse String	Wed, 22 Nov 2023 13:42:39 GMT	View

[Previous](#) [1](#) [2](#) [3](#) [Next](#)