CodeSync

Design Document

Team 1

Arsh Batth

Jayden Cheung

Arshnoor Randhawa

Adrien Qi

Index

Purpose

In today's world, software developers, students, and open-source contributors need a better way to work together on coding projects. Traditional tools for coding don't make it easy for people to collaborate in real-time, they often don't include help from AI, and they're not always open for everyone to use and improve. This makes it hard for people to work together efficiently, especially when they're not in the same place or when projects get complex.

That's where CodeSync comes in. It's a new kind of coding platform that's all about making it easier for people to code together, no matter where they are. CodeSync is a cloud-based platform, which means you can use it from anywhere. It lets people program together at the same time, offers help from AI to make coding faster and less error-prone, and comes with tools to manage coding projects from start to finish. And because it's open-source, anyone can help make it better.

Our goal with CodeSync is simple: we want to make coding together as easy and efficient as possible. We believe that by making a tool that's easy for everyone to use, we can help people build better software, learn more from each other, and bring their projects to life faster. Whether you're a professional developer, a student learning to code, or someone contributing to open-source projects, CodeSync is here to help you work better, together.

Functional Requirements

1. Account Management and Collaboration

As a user/developer/team member/collaborator,

1. I want to register an account so I can access the platform's features.
2. I want to log in to my account to start working on my projects.
3. I want to reset my password if I forget it, to regain access to my account.
4. I want to create a new project to start coding in a fresh workspace.
5. I want to invite others to collaborate on my project to work together in real-time.
6. I want to join a project I was invited to, so I can contribute to its development.
7. I want to see changes made by others in real time, to collaborate effectively.

2. Coding Environment

As a user/developer,

1. I want a basic text editor for coding with syntax highlighting for easier code readability.
2. I want to save my project progress so I can continue working on it later.
3. I want to use basic AI for code completion to code more efficiently.

4. I want to receive suggestions for debugging from the AI to fix errors more easily.
5. I want to search within my code to find and navigate to parts of it quickly.
6. I want to undo and redo actions to easily correct mistakes.
7. I want to view the history of changes made to the project to track progress.
8. I want Git integration for staging files, entering commit messages, committing files, viewing and comparing different branches, comparing files from previous commits, and special IDE support for merge controls.
9. I want to perform an interactive rebase with a progress bar for commits and manage merge conflicts efficiently.
10. I want to run tests on the platform to ensure the code runs as expected.
11. I want a split-screen feature to view multiple files side by side.
12. I want a quick access toolbar for frequently used features.
13. I want to customize keyboard shortcuts to work more efficiently.
14. I want to have a dashboard to view my active projects quickly.

3. Communication and Collaboration Tools

As a user/developer/team member,

1. I want to text chat with my collaborators remotely in real time.
2. I want to voice chat with my collaborators remotely in real time if time persists.
3. I want to video chat with my collaborators remotely in real time if time persists.
4. I want to share files within the platform so that my collaborators and I can easily access them.
5. I want to schedule coding sessions within the platform for planned collaboration times.
6. I want to view the profiles of my collaborators to learn about their skills and roles.
7. I want to have a whiteboard feature for brainstorming and planning to visualize ideas together.

4. Project and Code Management

As a user/developer/team member,

1. I want to customize the theme of my coding environment to fit my personal preference.
2. I want to download my project as a .zip to share it or deploy it outside the platform.
3. I want to have a feature to comment on code to leave notes for my team.
4. I want to have access to a library of code snippets to reuse common patterns easily.
5. I want to filter and sort my projects to organize my workspace.
6. I want to compile and run scripts online for supported languages.
7. I want to set permissions for collaborators to control who can edit or view the project.
8. I want to have access to documentation and tutorials to learn more about the platform's features.
9. I want to receive performance insights for my code to optimize it for better efficiency.
10. I want a feedback option to report issues or suggest improvements.

5. Task and Notification Management

As a user/team leader/team member,

1. I want to set, view, and adjust deadlines for individual tasks within the platform, enabling us to track our progress effectively against the project timeline.
2. I want the ability to assign team members to each task ensuring clear responsibility and accountability.
3. I want to view a personalized list of tasks assigned to me, complete with their deadlines.
4. I want to categorize tasks by priority, type (bug, improvement, feature), and add tags (frontend, backend, etc.) for organized task management.
5. I want to add detailed descriptions to tasks, including embedding code snippets for tasks related to bugs, with rich text formatting.
6. I want to view tasks organized by priority in a Kanban-like interface, with drag-and-drop functionality for updating task statuses.
7. I want to receive notifications within the platform (in app) for important updates or messages.
8. I want to be able to clear notifications so I do not have to see many messages.
9. I want notifications to be clickable so I can view them separately.

Non-Functional Requirements

1. Development and Deployment

As a developer,

1. I want the platform to be developed using Node.js, to leverage its efficiency for backend services.
2. I want to use React for the frontend to ensure a responsive user interface.
3. I want a simplified microservices architecture, focusing on core services like real-time collaboration and AI assistance, to ensure project feasibility within the given timeframe.
4. I want to deploy the application on a single cloud provider like AWS, using their basic managed services for scalability and reliability.

2. Database Management

As a developer,

1. I want to use a single type of database (preferably NoSQL) for ease of integration and to reduce complexity in data management.
2. I want to ensure the database supports basic scalability features to handle the initial user load effectively.

3. Usability

As a developer,

1. I want the user interface to be straightforward, ensuring ease of use with minimal training.
2. I want to prioritize mobile responsiveness, ensuring the platform is usable on desktop and tablet devices.

4. Performance

As a developer,

1. I want to ensure response times are within acceptable limits (under 2 seconds for user interactions) to provide a seamless experience.
2. I aim for the platform to support concurrent usage by hundreds of users initially, with the ability to scale as needed.

5. Security and Privacy

As a developer,

1. I want to implement basic security measures, including HTTPS for encrypted data transmission.
2. I want to use simple authentication mechanisms, like JWT, for user authentication and authorization, ensuring data protection without overcomplicating the system.
3. I plan for basic security audits to be conducted internally, focusing on common vulnerabilities.

6. Testing and Maintenance

As a developer,

1. I want to establish a clear distinction between development and production environments to facilitate testing and deployment.
2. I want automated deployment scripts for efficient updates and maintenance.
3. I aim to set up basic monitoring and logging to track the health and performance of the platform.

Design Outline

Design Decisions:

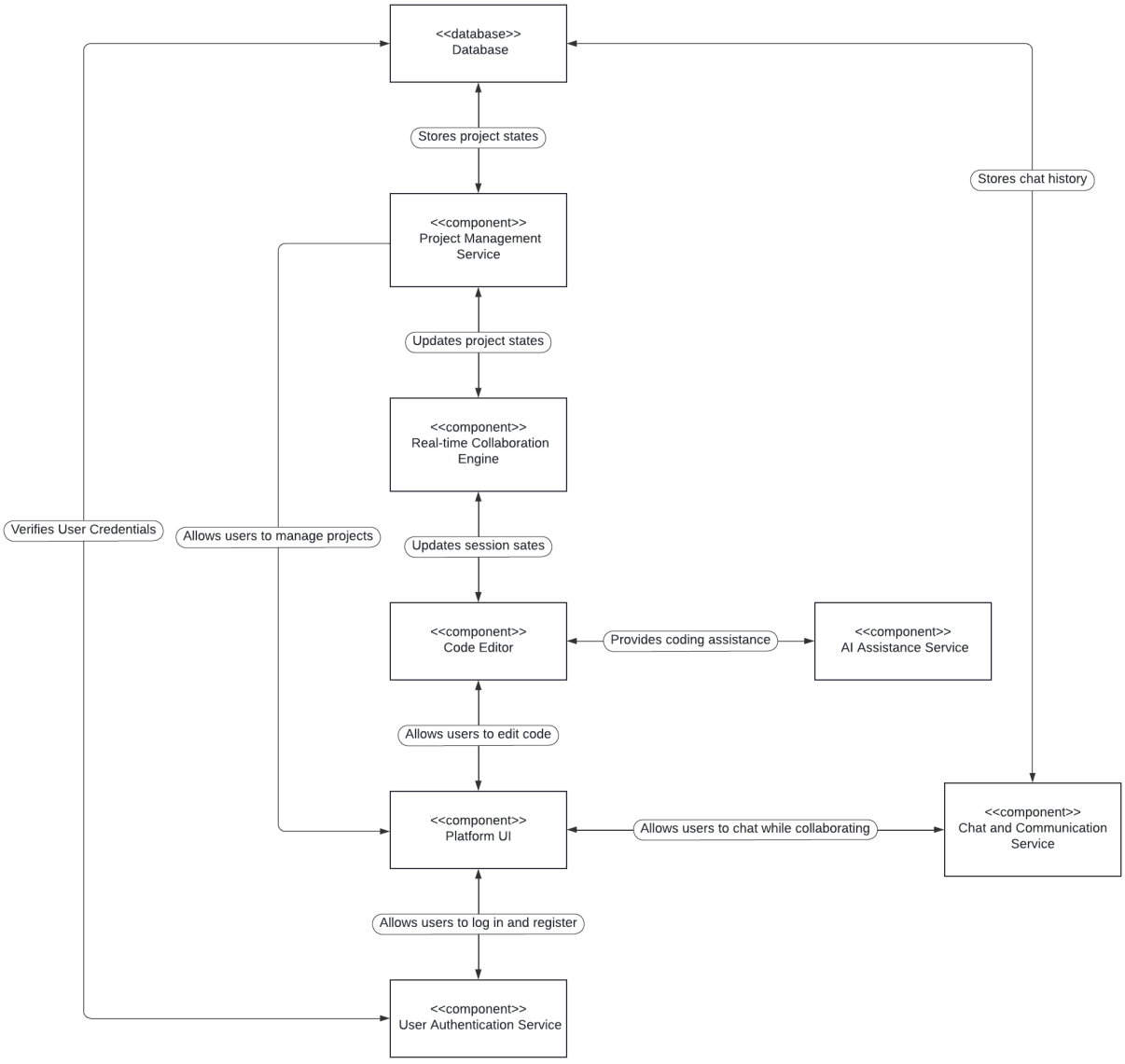
- **Client-Server Model:** Utilizes a web-based architecture for real-time collaboration and AI integration, ensuring accessibility and scalability.
- **Microservices Architecture**: Adopts microservices for modular development, independent scaling, and easier maintenance of different functionalities like real-time editing, AI assistance, and chat services.

System Components:

User Authentication Service: Manages user registration, login, and password resets.
Project Management Service: Allows creation, sharing, and editing of coding projects.
Real-time Collaboration Engine: Utilizes WebSocket for instant update sharing among collaborators.
AI Assistance Service: Provides code completion, debugging suggestions, and documentation generation.
Chat and Communication Service: Supports text, voice, and video communication within the platform.
Code Editor: A web-based text editor with syntax highlighting and AI-enhanced suggestions.
Git Integration Service: Manages version control, including commit, branch, and merge functionalities.
Database: Stores user data, project files, and chat histories.

Component Interactions:

The User Authentication Service interacts with the Database to verify user credentials. The Project Management Service communicates with the Real-time Collaboration Engine to update project states across all participants. The AI Assistance Service interfaces with the Code Editor to offer real-time coding assistance. The Chat and Communication Service is integrated into the platform UI to facilitate seamless collaboration.



High-Level Overview:

The following UML Diagram illustrates the high-level structure, showing the relationships and interactions between the components mentioned above.

Design Issues

Functional Issues:

1. **Projection Permissions and Access control:**
    a. Issue: Establishing a scalable and secure system to manage user roles and permissions for project access
    b. Solution Options:
        i. Role-Based Access Control (RBAC),
        ii. Attribute-Based Access Control (ABAC)
        iii. Project Specific Access Tokens
    c. Choice Justification: RBAC is selected for its organizational simplicity and the hierarchical nature of roles which align with the varied access requirements in a collaborative coding environment. ABAC can be overly complex for the anticipated user base, and project-specific access tokens, while useful, do not provide the same at-a-glance understanding of a user's permissions as RBAC.
2. **Project Collaboration Conflict Resolution:**
    a. Issue: Enhancing Real-Time Code Collaboration with Conflict Resolution Mechanisms
    b. Solution Options:
        i. Implement real-time conflict detection that alerts users to conflicting changes, providing an interface for manual merge resolution.
        ii. Introduce a locking mechanism where sections of code are "checked out" to prevent concurrent editing conflicts.
        iii. Use a version control system approach that automatically merges changes and highlights merge conflicts for review.
    c. Choice Justification: Real-time conflict detection is favored as it strikes a balance between maintaining fluid collaboration and providing essential feedback on conflicts. Locking mechanisms can hinder collaboration, and relying solely on version control systems can abstract the resolution process from the user, reducing immediacy and control.

3. **Signing up for an Account:**
    a. Issue: Establishing Secure Account Creation and Authentication
    b. Solution Options:
        i. Collect minimal information, requiring only a username and password, to simplify the signup process.
        ii. Require a username, password, and email to facilitate account recovery and verification, enhancing security.
        iii. Use only email and password to streamline the process while still allowing account recovery.
        iv. Integrate with GitHub using OAuth to authenticate users, leveraging existing developer accounts and streamlining the login process.
    c. Choice Justification: The combination of username, password, and email is chosen for its balance of security and user convenience. Email alone lacks a unique identifier, and while OAuth provides seamless integration, it ties accounts to a third-party service which may not be desirable for all users.
4. **Code Autocompletion Strategy:**
    a. Issues: Intelligent Code Completion Features for Enhanced Developer Productivity
    b. Solution Options:
        i. Inline autocomplete with ghost text
        ii. Autocomplete popup window
        iii. Contextual suggestion based on AI analysis
    c. Choice Justification: AI-driven contextual suggestions are preferred for their adaptability and unobtrusiveness. Inline autocompletion can disrupt the user's flow, and pop-up windows may distract from the coding experience.
5. **Project Creation and Management:**
    a. Issues: How to complete the process for users to create and manage projects within the platform
    b. Solution Options:
        i. Directions reaction within the platform with local storage
        ii. Integration with external repositories like Github or GitLab
        iii. A combination of both for flexibility.
    c. Choice Justification: The combination approach is chosen for offering the greatest flexibility, accommodating those who prefer an all-in-one platform and those who use external version control systems. Direct creation can limit collaboration, and exclusive external integration may alienate users not using these services.
6. **Team Collaboration and Task Allocation:**
    a. Issues: Dynamic Task Allocation and Management for Effective Team Collaboration
    b. Solution Options:

i. Manual task assignment through communication channels
ii. Automated task allocation based on skill set analysis
iii. Interactive Kanban board with drag-and-drop functionality

c. Choice Justification: The interactive Kanban board is the preferred choice due to its visual approach and ease of use, which enhances team collaboration. Manual assignment is time-consuming, and automated allocation may not reflect current project needs accurately.

7. **User-Facilitated Code Testing and Validation:**
    a. Issues: Developing Self-Service Code Testing Capabilities for Users
    b. Solution Options:
        i. Interactive GUI that allows users to write and execute custom test cases within the platform.
        ii. Predefined test suites that users can select and run against their code.
        iii. Integration with third-party testing services for advanced test execution.
        iv. Code playback feature where users can step through their code execution in real-time to identify issues.
    c. Choice Justification: An interactive GUI for writing and executing custom tests is selected for empowering users with the flexibility to test as needed. Predefined suites may not cover all use cases, third-party services introduce dependency and complexity, and code playback, while useful, does not replace the need for active testing.
8. **User Interface Customization:**
    a. Issue: Providing a Flexible and Personalized Development Environment
    b. Solution Options:
        i. Drag-and-drop interface for customizing layout.
        ii. Selection from a list of predefined layouts.
        iii. Minimal customization with resizable and collapsible panels.
    c. Choice Justification: Drag-and-drop interface for customizing layout is selected for its intuitiveness and the freedom it gives users to tailor the development environment to their preferences. Predefined layouts can limit personalization, and minimal customization may not meet all users' needs for an optimal workspace.
9. **Teammate Integration:**
    a. Issue: Streamlining the Process for Adding Collaborators to Projects
    b. Solution Options:
        i. Invite via email or username.
        ii. Search for users within the platform and send invitations.
        iii. Use group invitation links that can be shared externally.
    c. Choice Justification: Search for users within the platform and send invitations is the preferred method for its balance of convenience and control. It allows users to

quickly add collaborators without leaving the platform while providing the ability to review and accept teammates actively.

Non-Functional Issues:

1. **Real-Time Collaboration Synchronization:**
    a. Issue: Guaranteeing instantaneous code synchronization across all users in the collaborative environment.
    b. Solution Options:
        i. Operational Transformation (OT)
        ii. Conflict-Free Replicated Data Types (CRDTs)
    c. Choice Justification: CRDTs are chosen for their robust eventual consistency model, which is well-suited for distributed systems and ensures all users reach the same state without the need for complex conflict resolution, making them ideal for real-time applications.
2. **Scalability of the Real-time Collaboration Engine:**
    a. Issue: Scaling the collaboration engine to support a growing number of users without performance degradation.
    b. Solution Options:
        i. Vertical scaling with more powerful server hardware.
        ii. Horizontal scaling by adding more servers to the infrastructure.
    c. Choice Justification: Horizontal scaling is selected due to its alignment with a microservices architecture, offering enhanced resilience and the ability to scale out efficiently by adding more servers, which is especially beneficial for handling resource-intensive services.
3. **User Interface Responsiveness:**
    a. Issue: Ensuring a responsive and fluid user interface for an optimal user experience.
    b. Solution Options:
        i. Utilizing React's component-based architecture for optimized rendering.
        ii. Asynchronous loading of UI components.
        iii. Alternative frameworks or vanilla JavaScript for UI construction.
    c. Choice Justification: React, with proper optimization techniques, such as lazy loading and memoization, is chosen for its efficiency in updating and rendering components, which is critical for maintaining a responsive UI. While alternative frameworks or plain JavaScript could be used, React's widespread adoption and comprehensive tooling make it a reliable choice for a project of this scale.
4. **Database Management:**

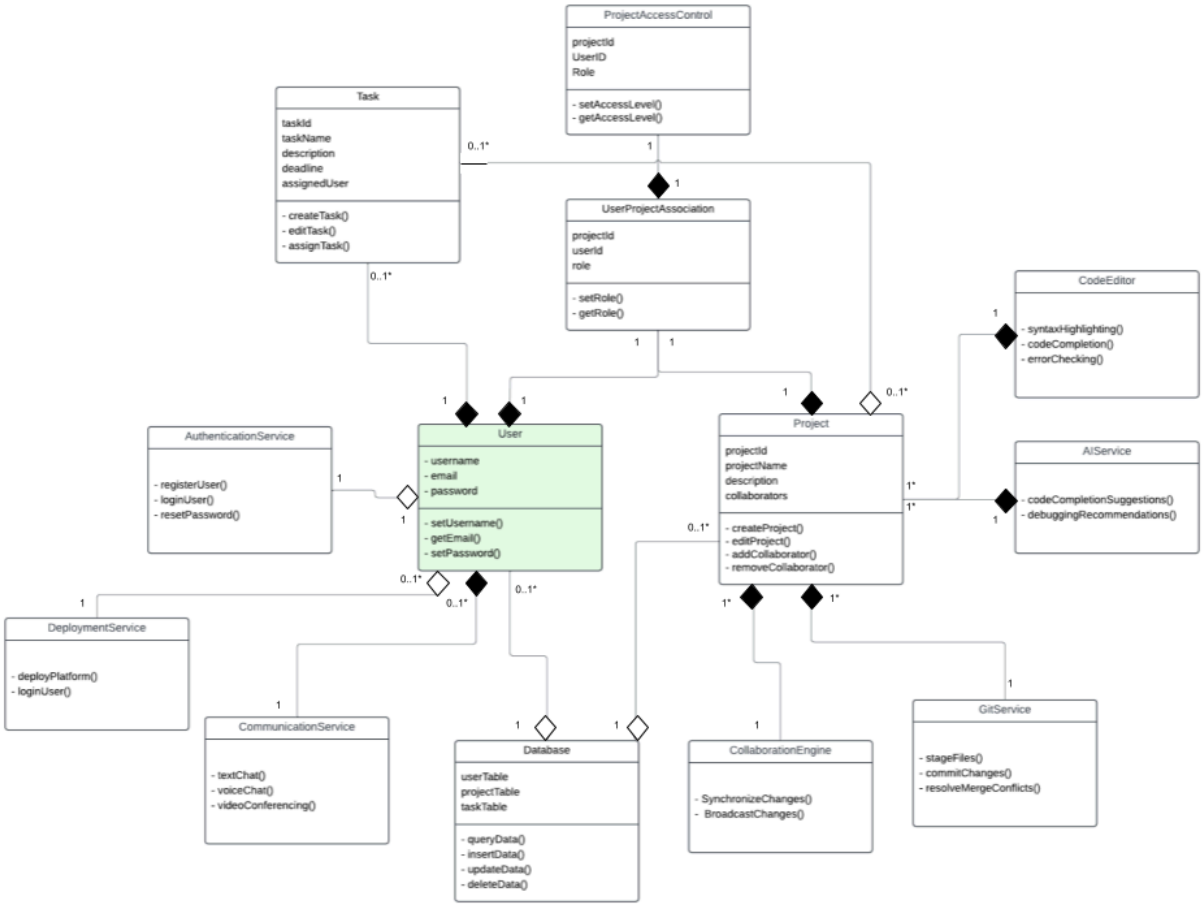
a. Issue: Selecting an appropriate database management system for scalability, data integrity, and performance.
b. Solution Options:
    i. Relational databases like MySQL or PostgreSQL.
    ii. NoSQL databases like MongoDB or Cassandra.
c. Choice Justification: NoSQL databases are selected, with MongoDB as the preferred choice due to its flexible schema, scalability, and strong performance with large, unstructured datasets, which are common in collaborative development platforms.

5. **Deployment Strategy:**
    a. Issue: Choosing an effective deployment strategy to ensure reliability and scalability of the platform.
    b. Solution Options:
        i. Cloud services like AWS, Azure, or Google Cloud Platform.
        ii. On-premises deployment with dedicated infrastructure.
    c. Choice Justification: AWS is chosen for its market-leading services and extensive experience in hosting scalable applications. Its managed services simplify deployment and scaling operations, and familiarity with AWS among the team reduces the learning curve.
6. **Security Compliance:**
    a. Issue: Ensuring the platform adheres to security standards and protects user data effectively.
    b. Solution Options:
        i. Security audits and compliance checks.
        ii. Implementation of an Information Security Management System (ISMS).
        iii. Adoption of security frameworks like OWASP Top 10 for web applications.
    c. Choice Justification: Implementing an ISMS, complemented by regular security audits, is chosen to establish a continuous security monitoring framework, reinforcing the platform's defense against threats and ensuring compliance with international security standards.
7. **Performance Optimizations:**
    a. Issue: Maximizing application performance to handle complex operations efficiently.
    b. Solution Options:
        i. Code profiling and optimization.
        ii. Implementation of efficient caching strategies.
    c. Choice Justification: A combination of code optimization and efficient caching strategies is chosen to improve the speed and efficiency of the platform, ensuring that performance remains high even as the complexity of operations increases.

Design Details

Class Diagram:

A detailed class diagram will be provided for each system component, outlining their attributes, methods, and relationships. For example, the Project Management Service will include classes such as Project, UserProjectAssociation, and ProjectAccessControl.



User Management:

1. User:
    a. Attributes:
        i. username: Unique identifier for the user.
        ii. email: User's email address.
        iii. password: Encrypted password for user login.
    b. Methods: setUsername(), getEmail(), setPassword(), etc.
        i. setUsername(newUsername): Updates the user's username.
        ii. getUsername(): Returns the current username.
        iii. getEmail(): Retrieves the user's email address.
        iv. setPassword(newPassword): Sets a new password for the user.
        v. verifyPassword(password): Checks if the provided password matches the user's current password.
        vi. updateProfile(profileData): Updates user's profile information based on the provided data

2. AuthenticationService
    a. Methods: registerUser(), loginUser(), resetPassword(), etc.
        i. registerUser(username, email, password): Creates a new user account.
        ii. loginUser(email, password): Authenticates a user and returns a session token.
        iii. resetPassword(email): Initiates the password reset process for the user.
        iv. verifyResetToken(token): Validates the password reset token.
        v. changePassword(userId, newPassword): Updates the user's password.

Project Management:

1. Project:
    a. Attributes:
        i. projectId: Unique identifier for the project.
        ii. projectName: Name of the project.
        iii. description: Brief description of the project.
        iv. collaborators: List of users collaborating on the project
    b. Methods:
        i. createProject(projectData): Initializes a new project with the given data.
        ii. editProject(projectId, projectData): Updates project details.
        iii. addCollaborator(projectId, userId): Adds a new collaborator to the project.
        iv. removeCollaborator(projectId, userId): Removes a collaborator from the project

2. UserProjectAssociation:
    a. Attributes:
        i. projectId: Identifier of the project.
        ii. userId: Identifier of the user.
        iii. role: The role of the user within the project (e.g., developer, manager)
    b. Methods:
        i. setRole(userId, projectId, role): Assigns a role to a user within a project.
        ii. getRole(userId, projectId): Retrieves the role of a user within a project.
3. ProjectAccessControl:
    a. Attributes:
        i. projectId: Identifier of the project.
        ii. accessLevel: Level of access granted to a user or role.
    b. Methods:
        i. setAccessLevel(projectId, userId, accessLevel): Sets the access level for a user on a project.
        ii. getAccessLevel(projectId, userId): Gets the access level of a user on a project.

Code Editor:

1. CodeEditor:
    a. Methods:
        i. syntaxHighlighting(language): Applies syntax highlighting based on the specified programming language.
        ii. codeCompletion(currentText): Provides code completion suggestions based on the current text.
        iii. errorChecking(code): Performs syntax and logical error checking on the provided code.
2. AIService:
    a. Methods:
        i. codeCompletionSuggestions(context): Generates code completion suggestions based on the given context.
        ii. debuggingRecommendations(code): Provides debugging recommendations for the supplied code snippet

Version Control (Git Integration):

1. GitService:
    a. Methods:

i. stageFiles(filePaths): Stages specified files for commit.
ii. commitChanges(commitMessage): Commits the staged changes with the given commit message.
iii. resolveMergeConflicts(conflictResolution): Resolves merge conflicts using the provided resolutions

Real-time Collaboration:

1. CollaborationEngine
    a. Methods:
        i. synchronizeChanges(changeSet): Synchronizes code changes across all collaborators in real-time.
        ii. broadcastChanges(changeSet): Broadcasts code changes to all participants in the session.

Communication Tools:

1. CommunicationService
    a. Methods:
        i. textChat(message, userId): Sends a text message to a chat session.
        ii. voiceChat(audioStream, userId): Initiates a voice chat session.
        iii. videoConferencing(videoStream, userIds): Starts a video conferencing session with multiple users

Task and Notification Management:

1. Task:
    a. Attributes:
        i. taskId: Unique identifier for the task.
        ii. taskName: Name of the task.
        iii. description: Detailed description of the task.
        iv. deadline: Deadline for the task completion.
        v. assignedUser: User to whom the task is assigned.
    b. Methods:
        i. createTask(taskData): Creates a new task with the specified details.
        ii. editTask(taskId, taskData): Updates the details of an existing task.
        iii. assignTask(taskId, userId): Assigns a task to a specified user.
2. NotificationService

a. Methods:
    i. sendNotification(userId, message): Sends a notification to a specific user.
    ii. receiveNotification(): Receives incoming notifications.

Database Management:

1. Database:
    a. Attributes:
        i. tables: Collection of tables (e.g., userTable, projectTable, taskTable) used within the database.
    b. Methods:
        i. queryData(query): Executes a given SQL query and returns the result.
        ii. insertData(table, data): Inserts data into the specified table.
        iii. updateData(table, data, conditions): Updates data in a specified table based on conditions.
        iv. deleteData(table, conditions): Deletes data from a specified table based on conditions

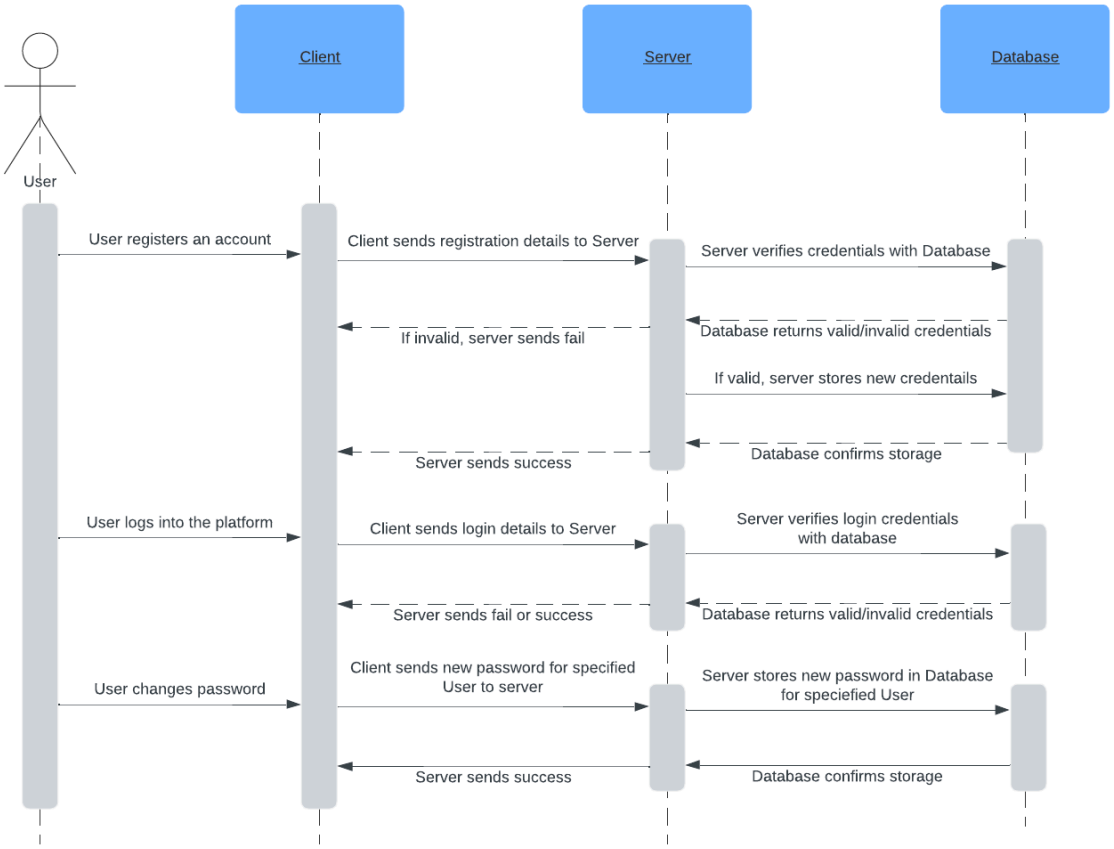
Deployment and Infrastructure:

1. DeploymentService:
    a. Methods:
        i. deployPlatform(applicationData): Deploys the application to a specified environment.
        ii. manageInfrastructure(infrastructureData): Manages the infrastructure settings and resources for the platform

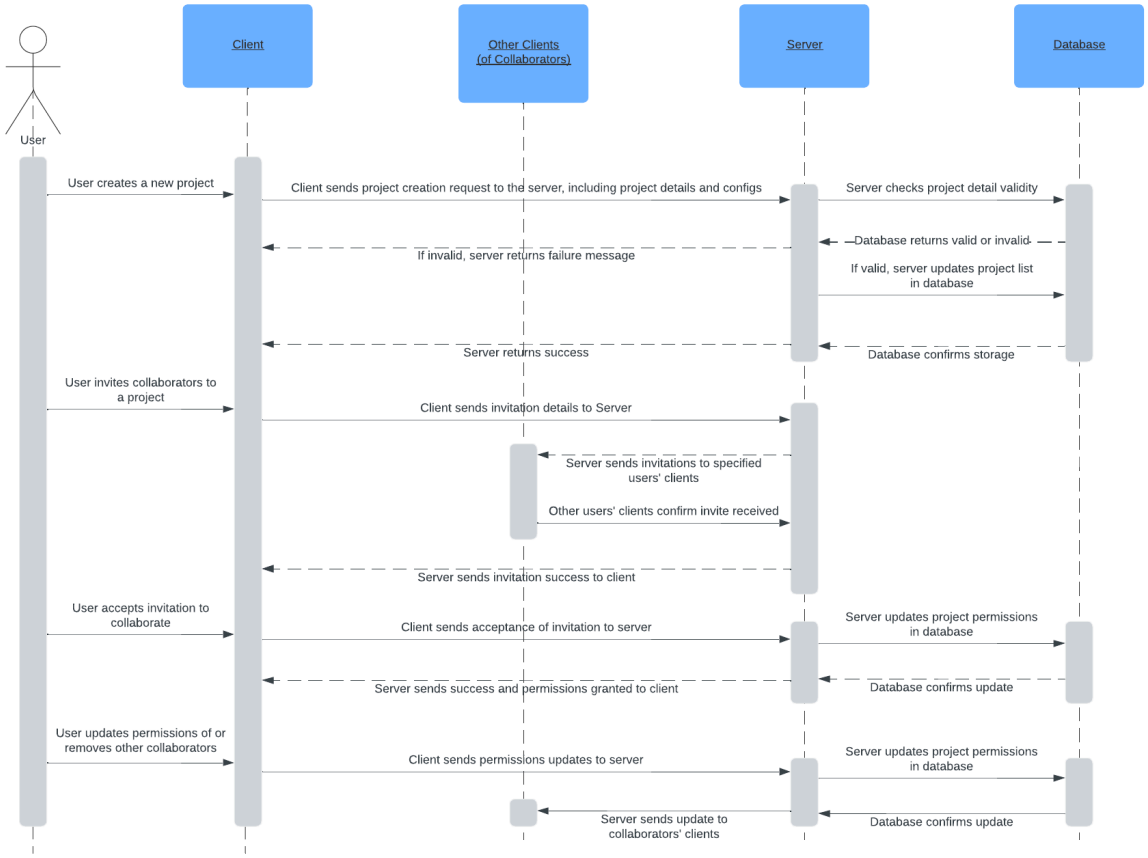
Sequence Diagrams:

The following Sequence Diagrams illustrates key activities such as user login, project creation, real-time collaboration flow, and AI-generated coding suggestions, detailing interactions between system components and services.

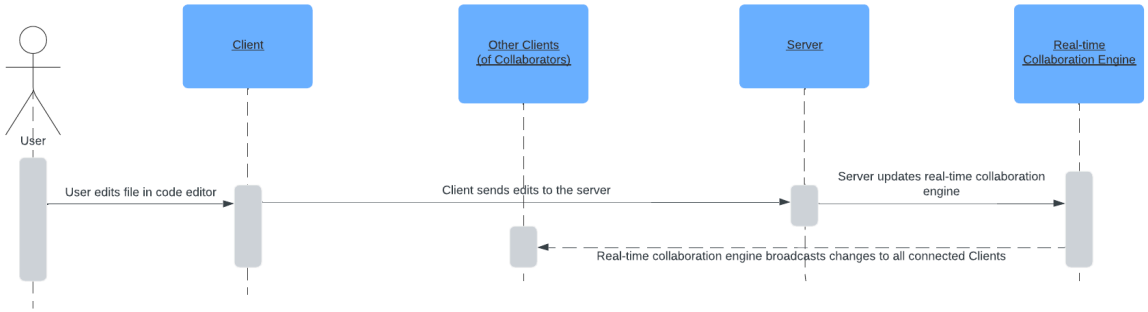
1. User Authentication Process



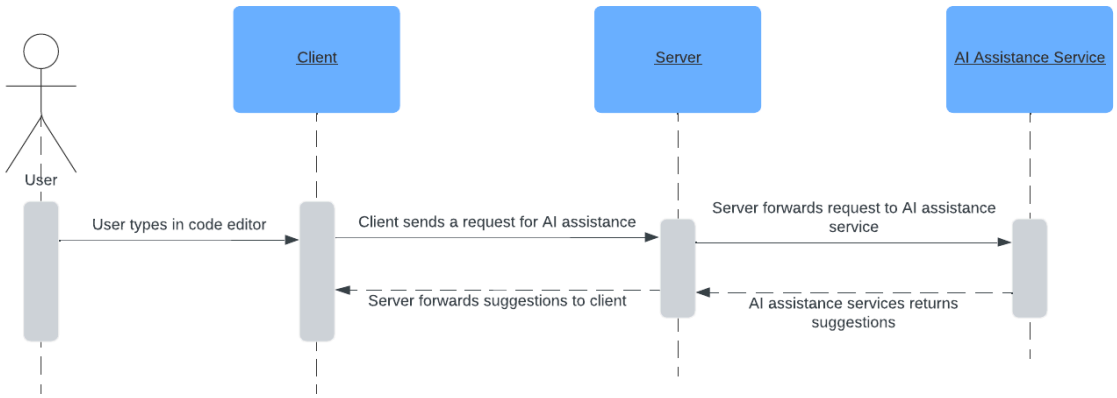
2. Project Management



3. Real-time Collaboration

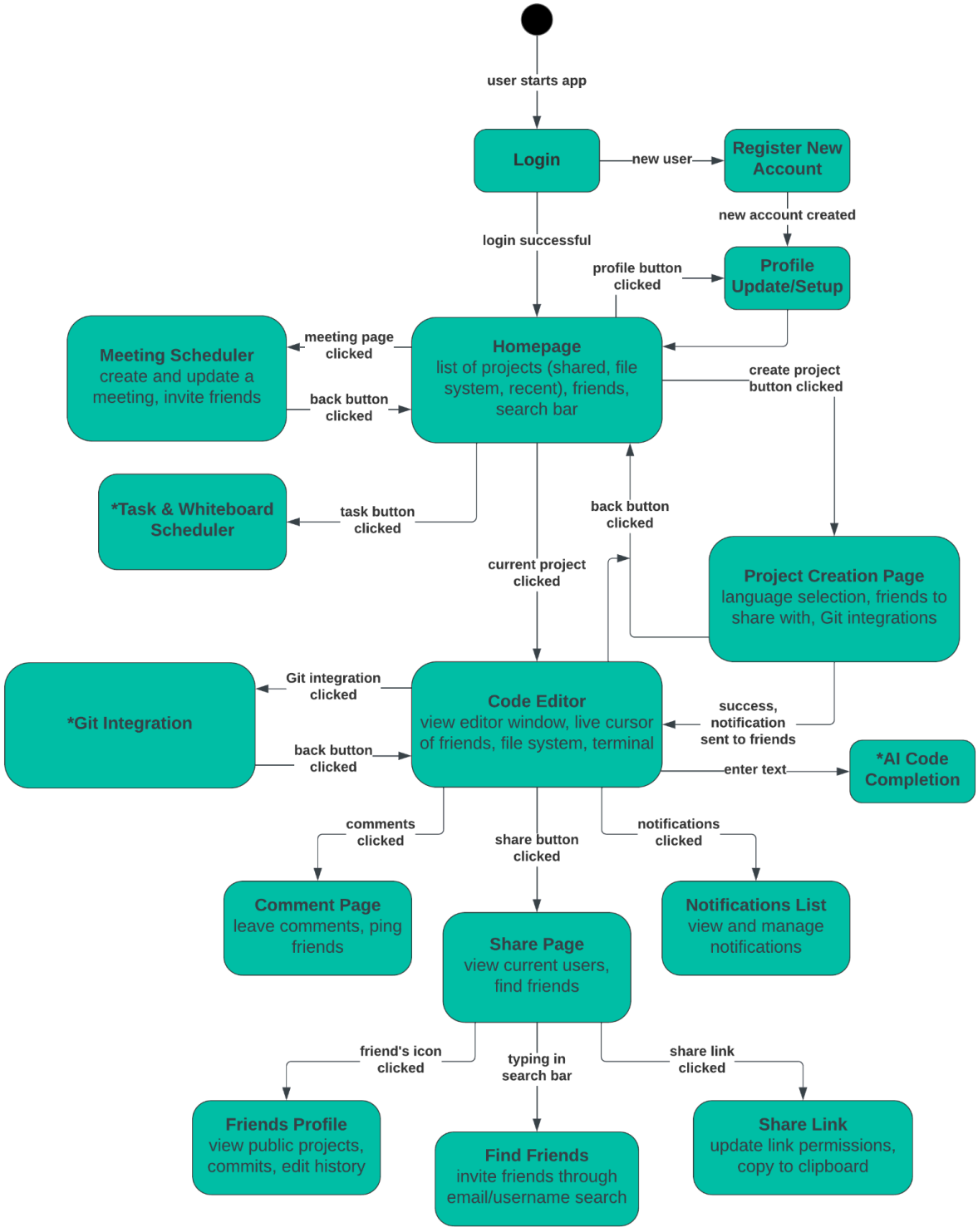


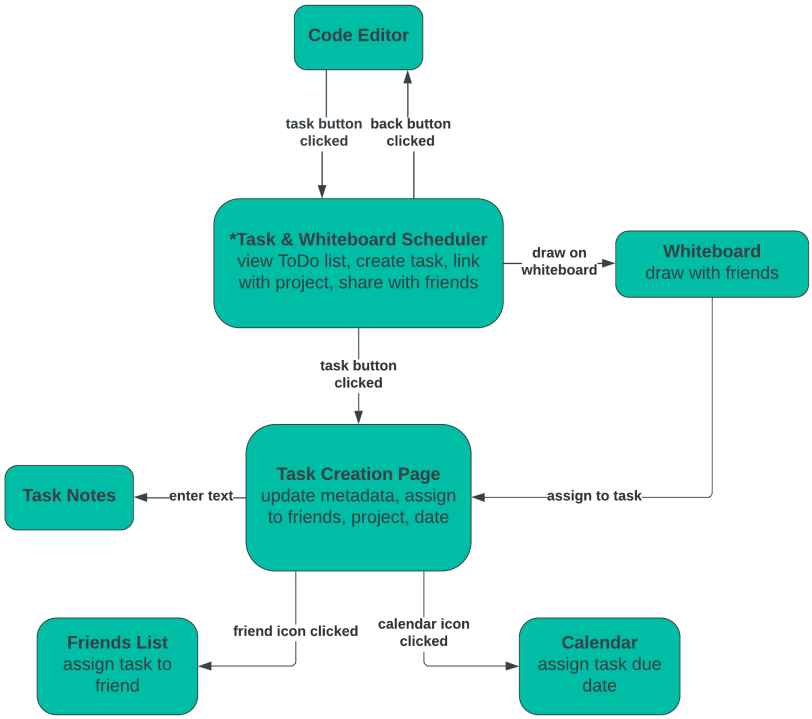
4. AI-Assisted Coding

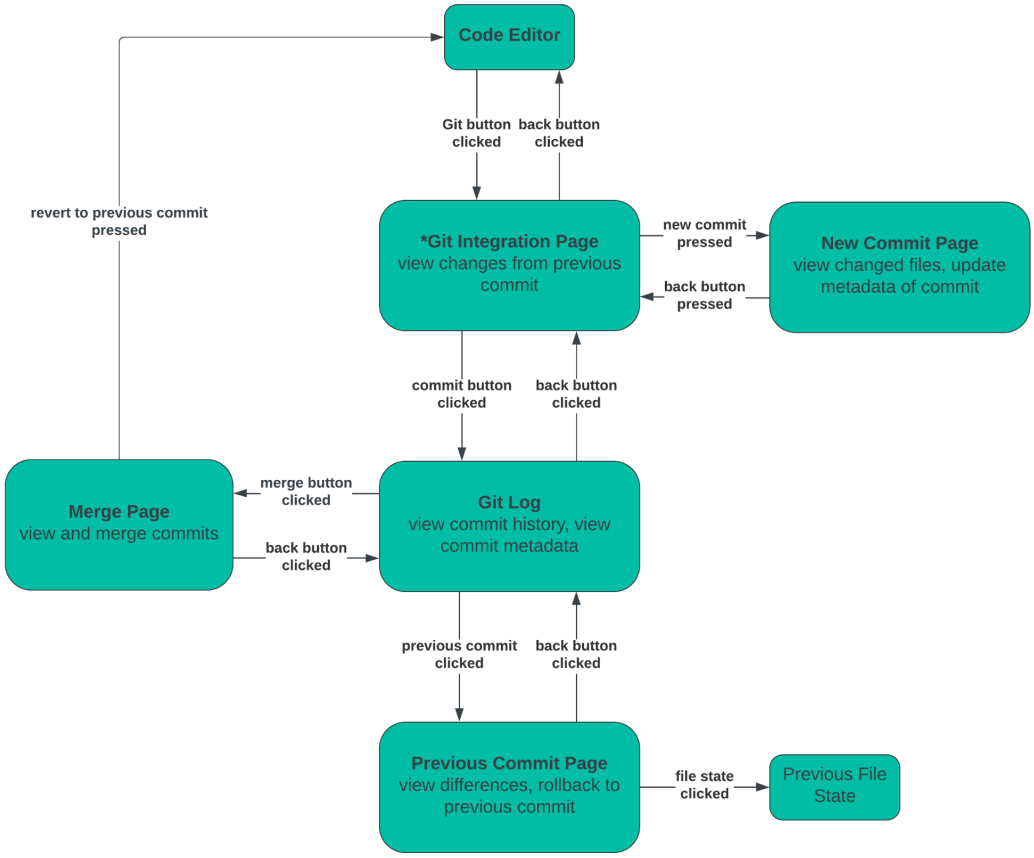

Client
Server
AI Assistance Service
User
User types in code editor
Client sends a request for AI assistance
Server forwards request to AI assistance service
Server forwards suggestions to client
AI assistance services returns suggestions

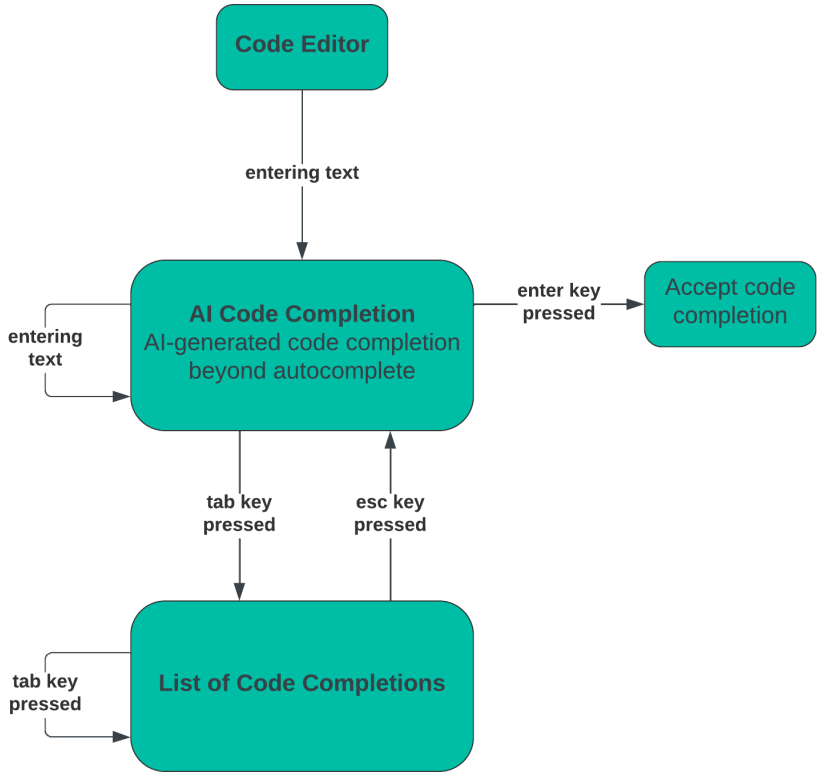
Activity Diagram:

The activity diagram below represents a high level diagram of features, Git integration, Task & Whiteboard scheduler, and AI autocompletion.




Code Editor
task button clicked
back button clicked
*Task & Whiteboard Scheduler
view ToDo list, create task, link with project, share with friends
draw on whiteboard
Whiteboard
draw with friends
task button clicked
Task Creation Page
update metadata, assign to friends, project, date
Task Notes
enter text
assign to task
Friends List
assign task to friend
friend icon clicked
calendar icon clicked
Calendar
assign task due date

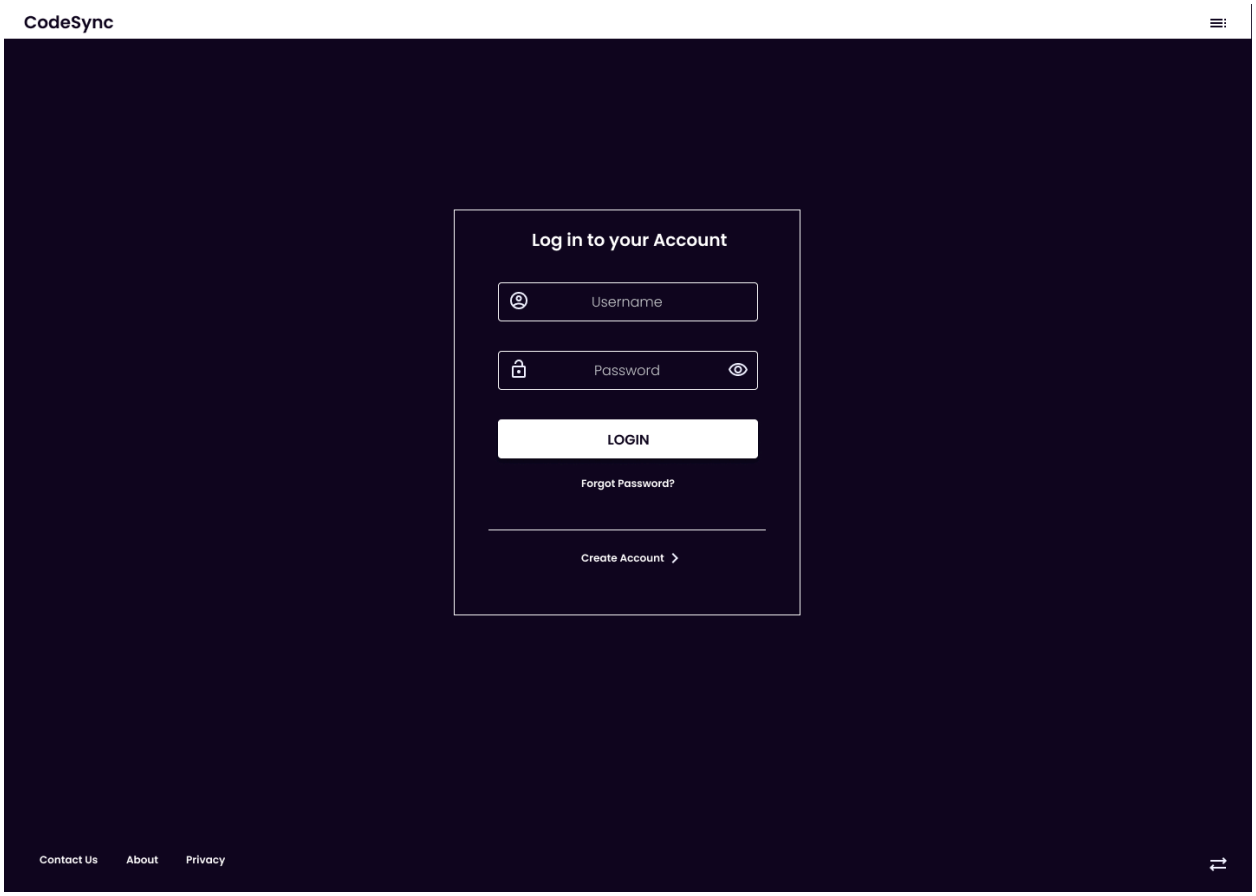

Code Editor
Git button clicked
back button clicked
revert to previous commit pressed
*Git Integration Page
view changes from previous commit
new commit pressed
back button pressed
New Commit Page
view changed files, update metadata of commit
commit button clicked
back button clicked
Merge Page
view and merge commits
merge button clicked
back button clicked
Git Log
view commit history, view commit metadata
previous commit clicked
back button clicked
Previous Commit Page
view differences, rollback to previous commit
file state clicked
Previous File State



UI Mockup:

The following UI Mockup showcases the design of the web interface, highlighting the code editor, project management dashboard, chat interface, and settings menu for user customization.

Login Page:

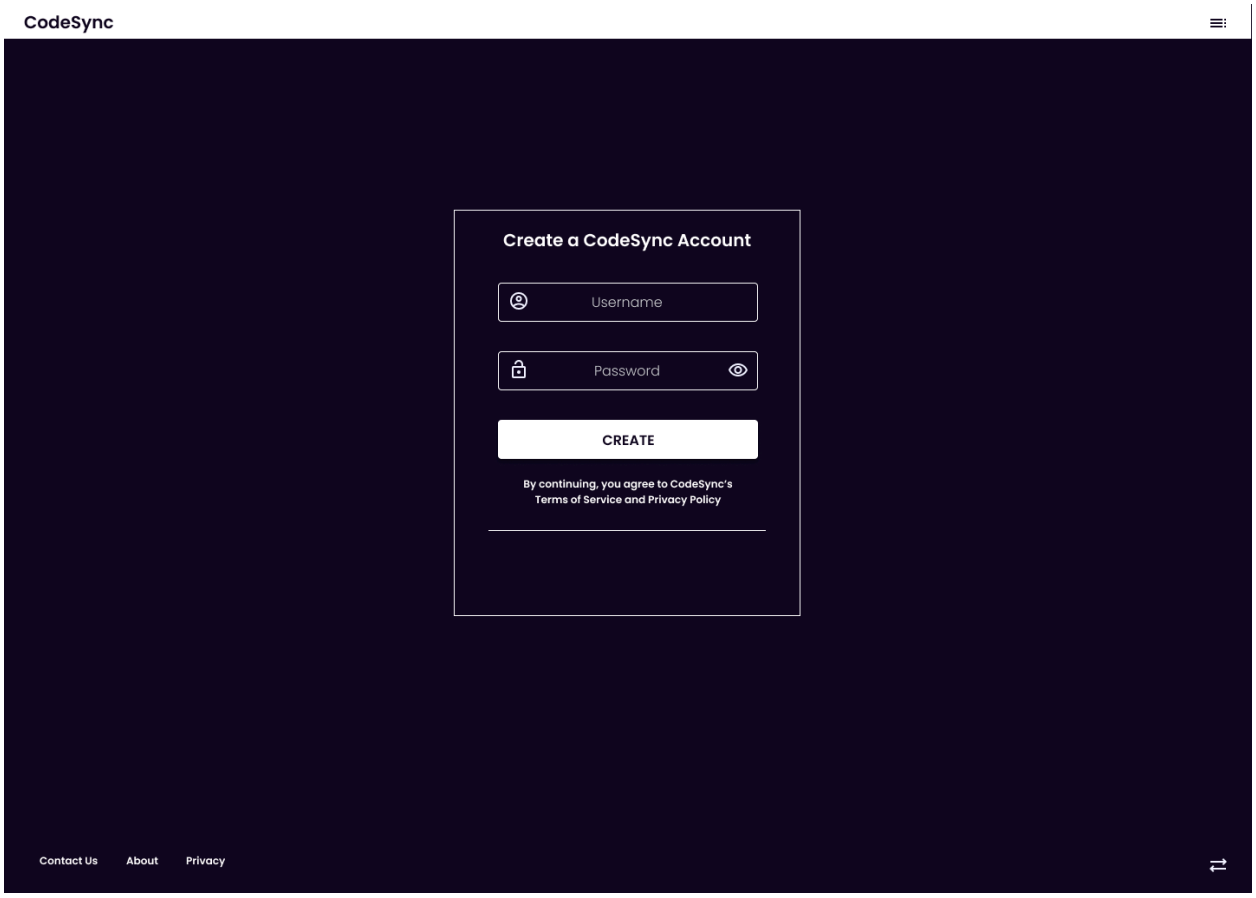


This is the login page of the application. It allows the user to enter their Username and Password and to be routed to the dashboard when the login button is pressed and valid credentials are provided. If invalid credentials are provided then an error popup will appear detailing the invalid credentials. There are also buttons for if the user forgets their password which will send them to the password reset page as well as buttons for the create account page which will send them to the registration page. The top right three bars also provide the user the ability to access different pages that are available to non-logged in users. The bottom left allows the user to access the ‘Contact Us’, ‘About’, and ‘Privacy’ pages. The bottom right allows the user to change the theme.

Associated Functional Requirements:

1. As a user, I would like to log in to my account so that I can start working on my projects.

Registration Page:

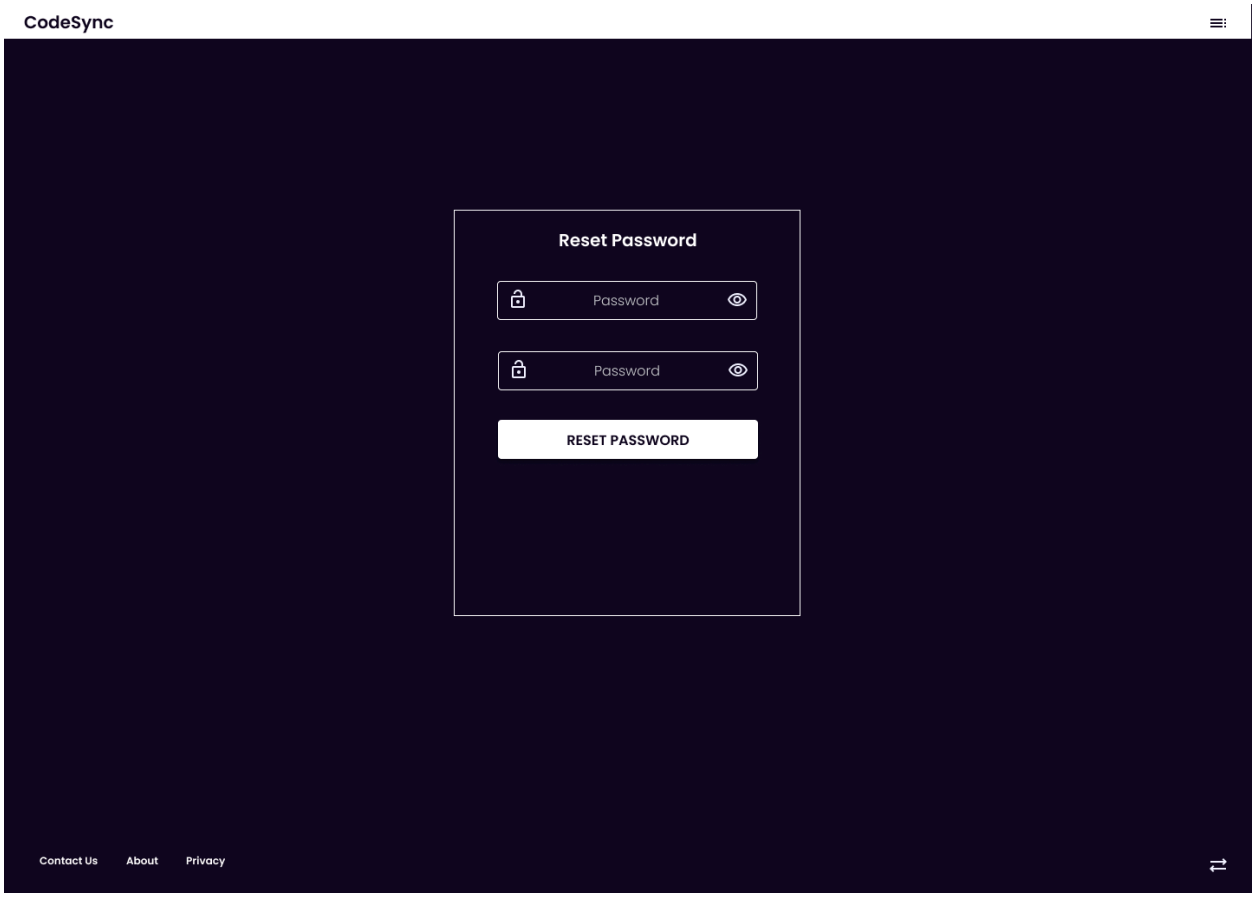


This is the registration page of the application. It allows the user to enter the username and password. The create button will create the profile and add it to the database once it has verified the credentials are valid. It will add them in a prepared statement for security. The page also allows for the user to read the ‘Terms of Service’ and ‘Privacy Policy’ if users click on the words. The top right three bars also provide the user the ability to access different pages that are available to non-logged in users. The bottom left allows the user to access the ‘Contact Us’, ‘About’, and ‘Privacy’ pages. The bottom right allows the user to change the theme.

Associated Functional Requirements:

1. As a developer, I would like to register an account so that I can access the platform's features.

Password Reset Page:

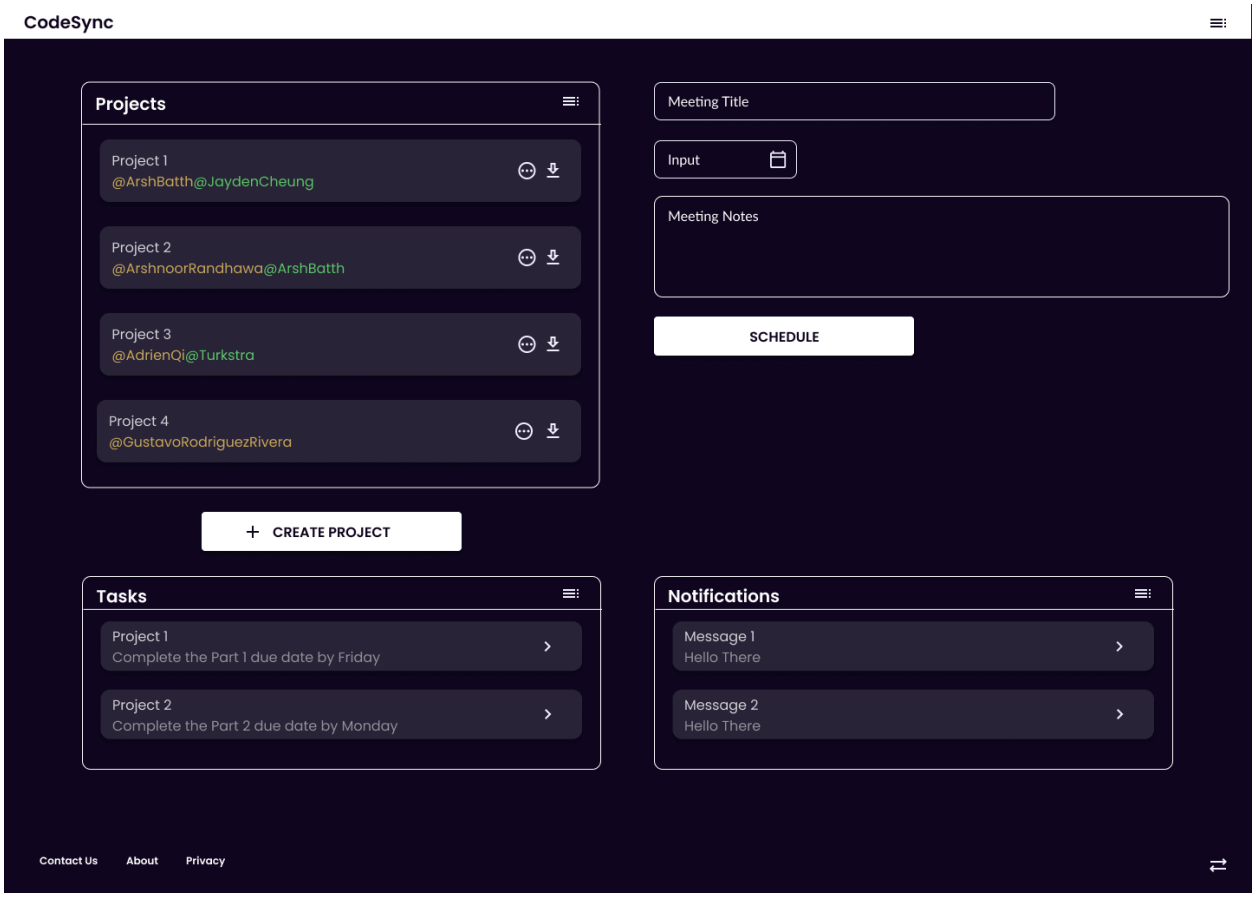


This is the password reset page of the application. It allows the user to enter the password twice. The reset password button will give an error popup if the user provides mismatched passwords or weak passwords. If there are no errors, the password will be updated in the database. The top right three bars also provide the user the ability to access different pages that are available to non-logged in users. The bottom left allows the user to access the ‘Contact Us’, ‘About’, and ‘Privacy’ pages. The bottom right allows the user to change the theme.

Associated Functional Requirements:

1. As a user, I would like to reset my password if I forget it to regain access to my account.

Dashboard/Home Page:



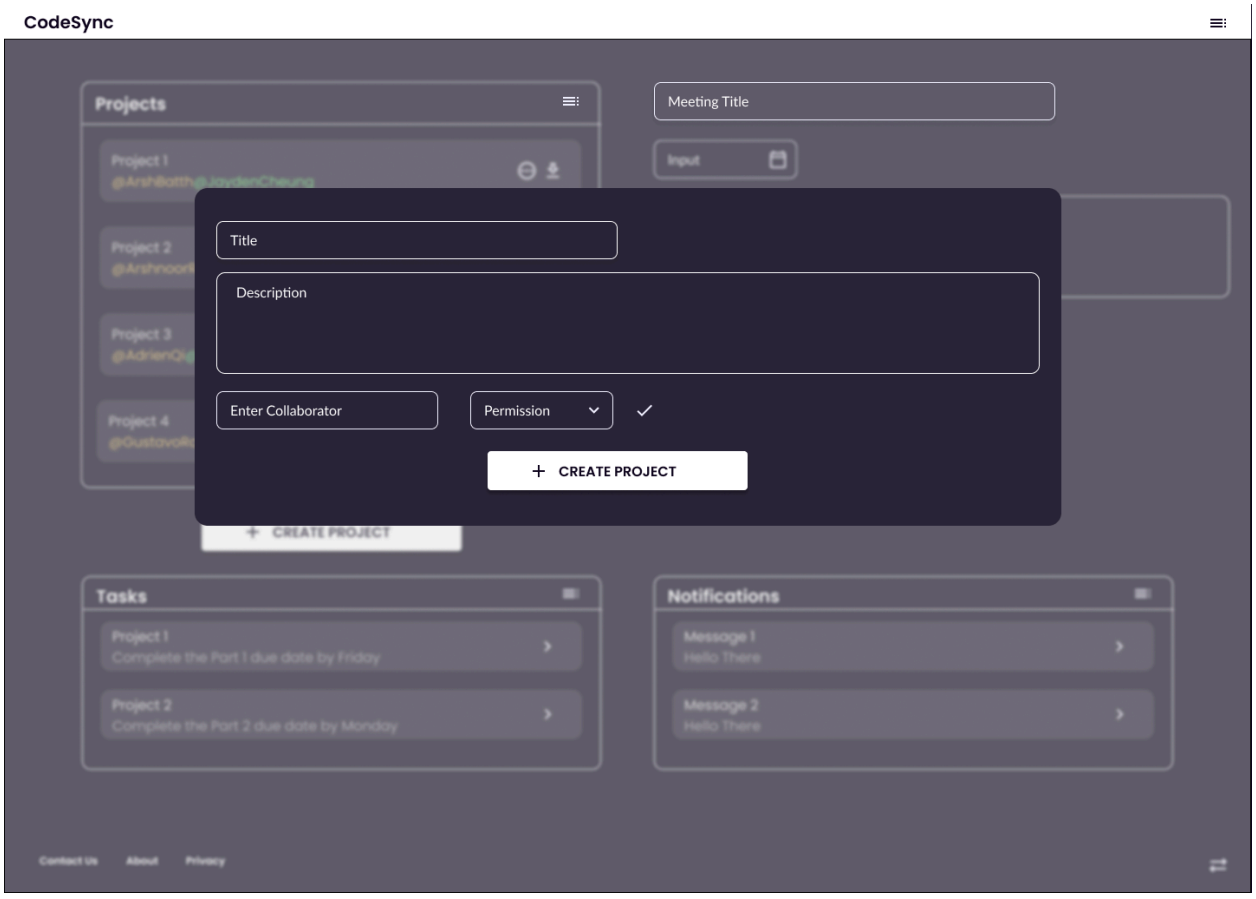
This is the dashboard/home page of the application. The projects tab in the top left lists all the projects that the user is a part of or has been invited to. The three bars at the top right of the project tab allows the user to filter the project based on certain parameters. Each individual project tab lists the title of the project as well as the current members of it. Additionally the download button allows the project to be extracted as a .zip file. The three dots allows the users to do actions on the project (edit, delete etc.). The top right allows the user to schedule the coding sessions. It takes in parameters of the meeting title, date and notes. Pressing the schedule button will send the meeting to the invited members in their tasks tab. The tasks tab lists the tasks that are assigned by the team leader or other members of the team. You can filter them by using the three lines at the top right. The notifications tell the notifications and you can filter them, read them by pressing the >, or clear all of them. You can also hover on the '@USER' to open up a profile popup of the user.

Associated Functional Requirements:

1. As a collaborator, I would like to join a project I was invited to so that I can contribute to its development.

2. As a user, I would like to receive notifications within the platform (in app) for important updates or messages.
3. As a user, I would like to be able to clear notifications so that I do not have to see many messages
4. As a user, I would like to have notifications be clickable so that I can go and view them separately
5. As a team member, I would like to schedule coding sessions within the platform so that we can plan collaboration times.
6. As a user, I would like to download my project as a .zip so that I can share it or deploy it outside the platform.
7. As a user, I would like to filter and sort my projects so that I can organize my workspace.
8. As a team member, I would like to view the profiles of my collaborators so that I can learn more about their skills and roles.
9. As a team member, I would want to view a personalized list of tasks assigned to me, complete with their deadlines. This feature should provide an overview of my immediate priorities and help me manage my workload effectively.

Project Creation Page:

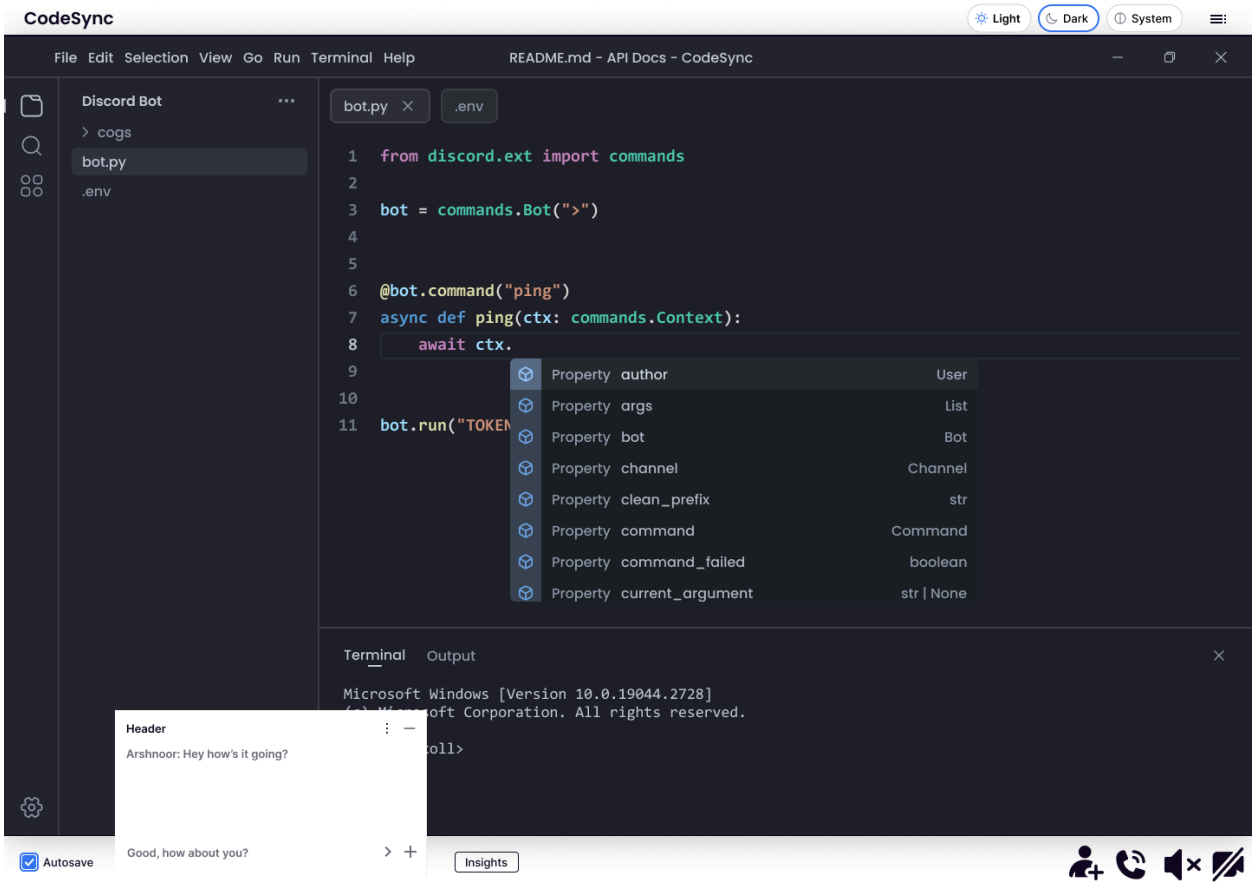


This is the project creation popup of the application. It comes up when the user clicks on the '+ CREATE PROJECT' button. It allows the user to enter in the Title of the project, a description for the project as well as enter collaborators with a drop down for permissions. When the user clicks the check box they will be added and displayed below the collaborator input and permission drop down. Once you click create project, it will create the project and send the invites.

Associated Functional Requirements:

1. As a developer, I would like to create a new project so that I can start coding on a fresh workspace.
2. As a team member, I would like to invite others to collaborate on my project so that we can work together in real time.

Project Workspace/IDE Page:



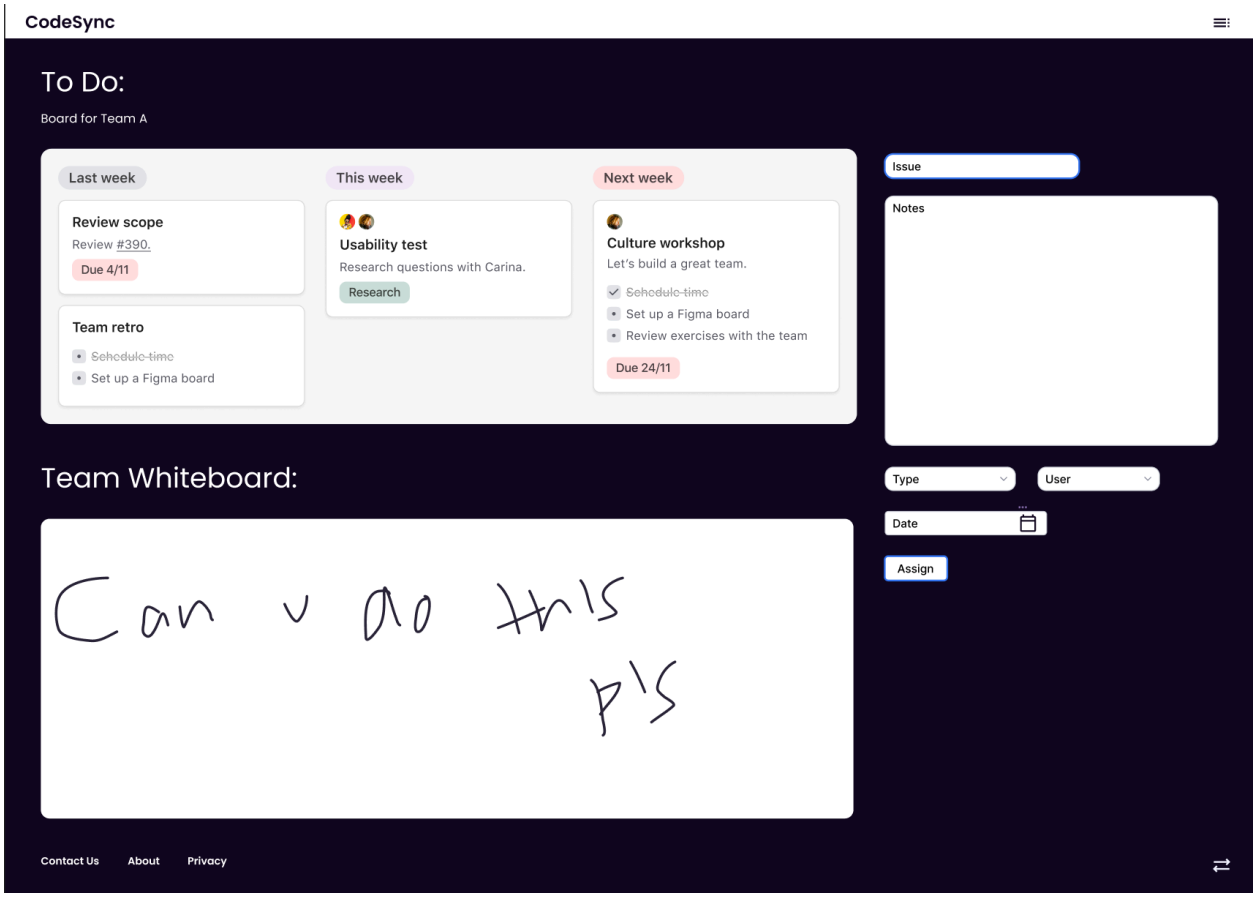
This is the project development workspace and IDE. It comes up when the user opens up a project they have created or been invited to. At the top there are tabs to switch between the visual editor, code editor, terminal, database, and other development tools. Within the code editor tab, users can write code collaboratively in real-time. There are code formatting, highlighting, auto-complete, and debugging assist features powered by AI. On the right side panel, there are options to open the Git version control features, user management, forum-style messaging between project members, and voice/video chat. Along the top, there is a toolbar to save, share, download, and manage files related to the project. Along the buttons there are options to autosave, chat, call and video call as well as code insights.

Associated Functional Requirements:

1. As a developer, I would like to see changes made by others in real time so that I can collaborate effectively.
2. As a user, I would like to have a basic text editor for coding with syntax highlighting for easier code readability.
3. As a user, I would like to save my project progress so that I can continue working on it later.
4. As a developer, I would like to text chat with my collaborators remotely in real time so that we can discuss the project as we work.
5. As a developer, I would like to voice chat with my collaborators remotely in real time so that we can discuss the project as we work if time persists.
6. As a developer, I would like to video chat with my collaborators remotely in real time so that we can discuss the project as we work if time persists.
7. As a team member, I would like to share files within the platform so that my collaborators and I can easily access them.
8. As a user, I would like to customize the theme of my coding environment so that it fits my personal preference.
9. As a developer, I would like to use basic AI for code completion so that I can code more efficiently.
10. As a user, I would like to receive suggestions for debugging from the AI so that I can fix errors more easily.
11. As a developer, I would like to search within my code so that I can find and navigate to parts of it quickly.
12. As a user, I would like to undo and redo actions so that I can easily correct mistakes.
13. As a user, I would like to view the history of changes made to the project so that I can track progress [Git Integration]
14. As a user, I would like to stage files, enter commit messages, and commit those files in a dedicated view [Git Integration]
15. As a user, I would like to be able to view and compare different branches.
16. As a user, I would like to compare files from previous commits so that I can revert individual files if necessary or revert the entire commit [Git Integration]

17. As a user, I would like to have special IDE support for merge controls which let me view each conflict and accept them so that I can resolve them with ease [Git Integration]
18. As a user, I would like to perform an interactive rebase, which displays a progress bar for how many commits have been applied, and presents each merge conflict so that I can make sure they are resolved and the code has been rebased [Git Integration]
19. As a developer, I would like to run tests on the platform so that I can ensure the code runs as expected.
20. As a team leader, I would like to set permissions for collaborators so that I can control who can edit or view the project.
21. As a developer, I would like to have a split-screen feature so that I can view multiple files side by side.
22. As a developer, I would like to compile and run scripts online for supported languages.
23. As a developer, I would like to receive performance insights for my code so that I can optimize it for better efficiency.
24. As a developer, I would like to have a feature to comment on code so that I can leave notes for my team.
25. As a user, I would like to have a quick access toolbar so that I can easily reach frequently used features.
26. As a developer, I would like to customize keyboard shortcuts so that I can work more efficiently.
27. As a user, I would like to have a dashboard to view my active projects so that I can quickly access them.
28. As a developer, I would like to have access to a library of code snippets so that I can reuse common patterns easily.

Task and Program Management and Whiteboard/Brainstorming Page:



This is the task and whiteboard page of the application. It comes up when the user clicks on 'Tasks & Whiteboard' in the sidebar. It allows the user to access the digital whiteboard canvas by clicking on the 'Whiteboard' tab, where they can draw diagrams and collaborate in real time. There is also a 'Tasks' tab that displays the Kanban-style task management board. Here users can create new tasks by clicking the '+ New Task' button. This brings up a popup where the user enters the task title, description, tags, priority, and assignee. They can also set a due date with the date picker. Once a task is created, it displays as a card on the Kanban board that can be dragged between workflow columns.

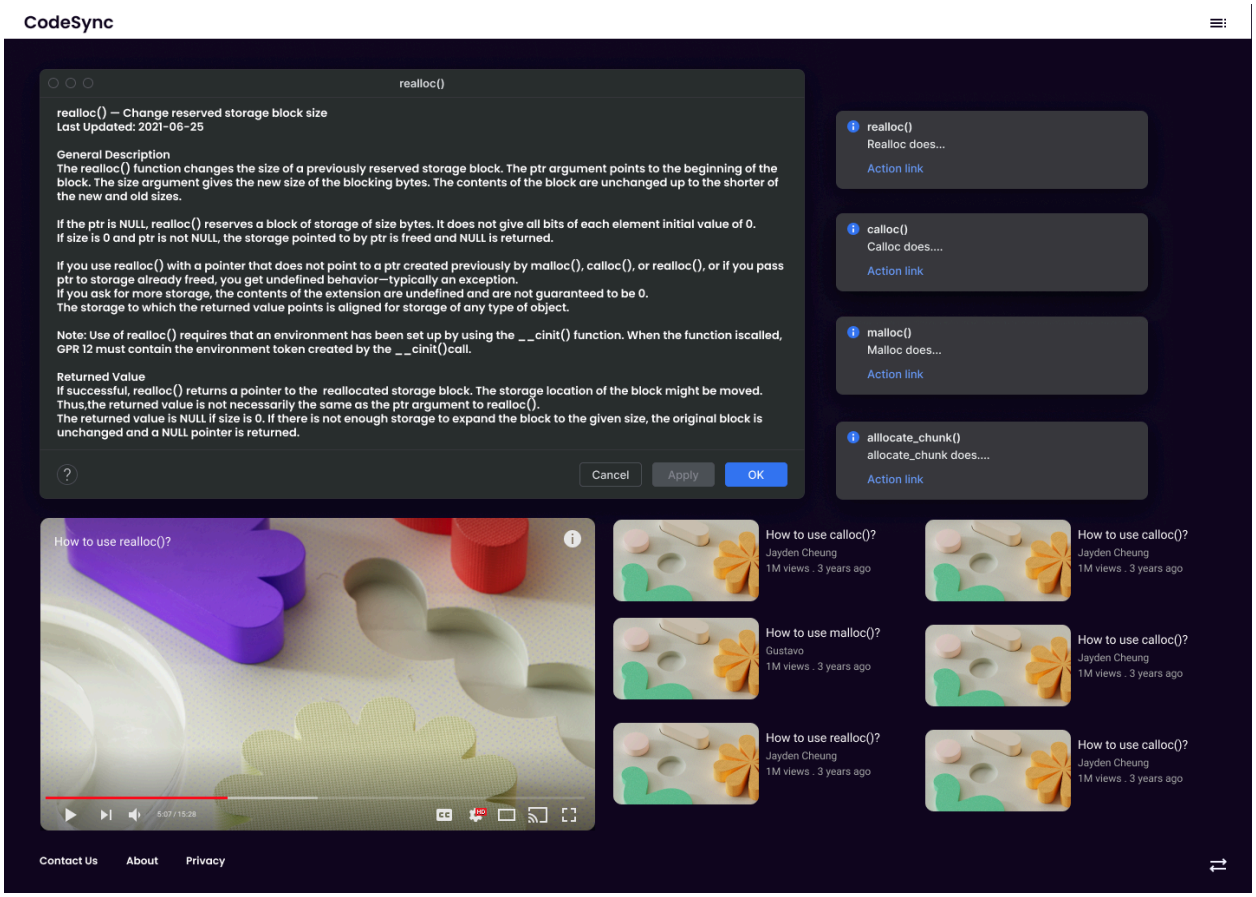
Associated Functional Requirements:

1. As a team member, I would like to have a whiteboard feature for brainstorming and planning so that we can visualize ideas together.
2. As a team leader, I want to set, view, and adjust deadlines for individual tasks within the platform enabling us to track our progress against the project timeline effectively. This

feature allows for the easy identification of upcoming, overdue, and completed tasks based on their deadlines.

3. As a team leader, I want the ability to assign team members to each task ensuring clear responsibility and accountability. This functionality should support multiple team members in a task if necessary and allow reassignment as the project evolves.
4. As a team leader, I need to categorize tasks by priority, type (bug, improvement, feature), and add tags (frontend, backend, etc.). This will facilitate organized task management, allowing the team to filter and prioritize work based on these categories.
5. As a team member, I want to add detailed descriptions to tasks, including the ability to embed code snippets for tasks related to bugs. This feature should support rich text formatting to enable clear communication of task requirements and details.
6. As a team member, I need to view tasks organized by priority in a Kanban-like interface. This visualization should allow for easy identification of high-priority tasks and support drag-and-drop functionality to update task statuses as they move through different stages of completion.

Documentation and Tutorials Page:

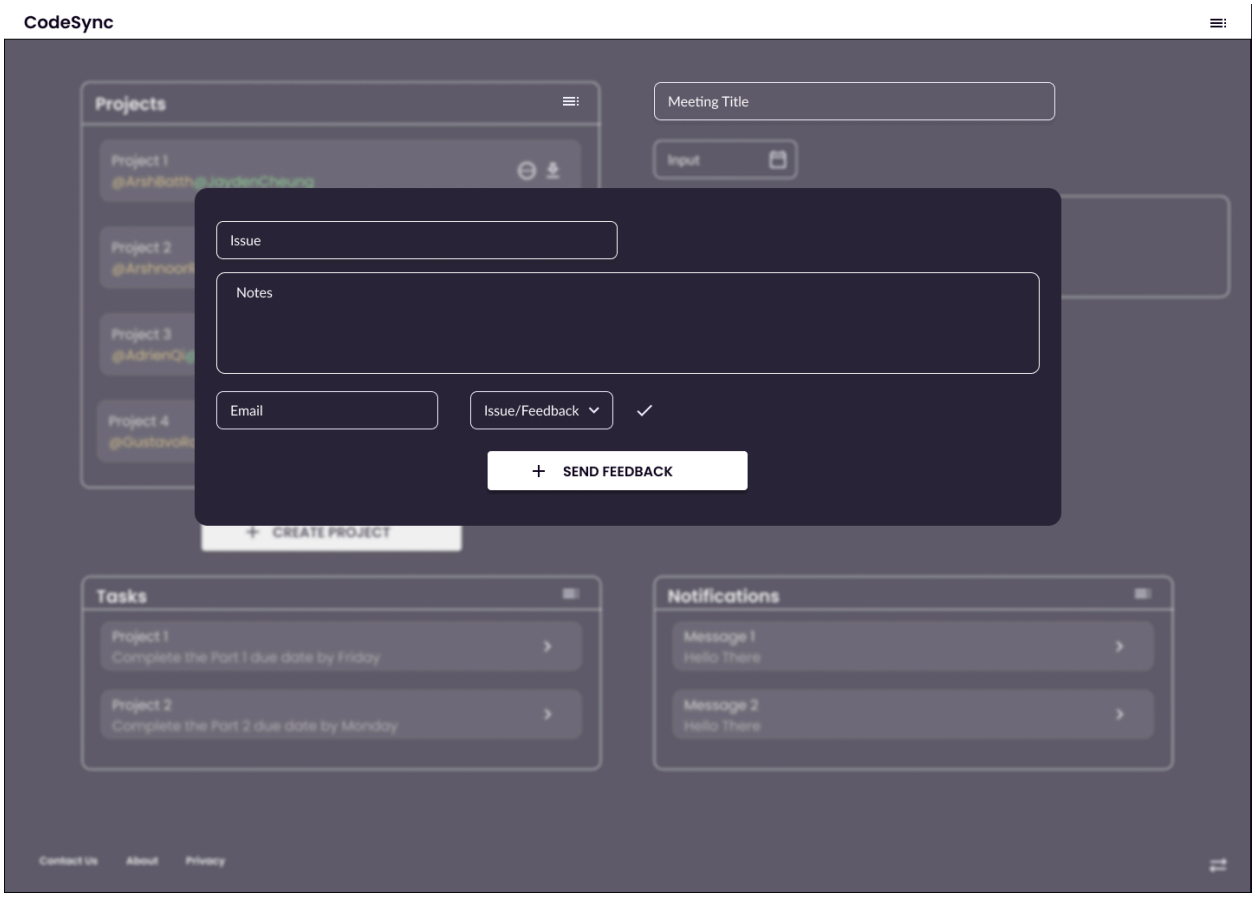


This is the help and tutorials page of the application. It comes up when the user clicks on 'Help & Tutorials' in the sidebar. The top of the page features a search bar where users can quickly find documentation and guides by keyword. There is also a section of featured 'Getting Started' tutorials for beginners. Scrolling down brings you to categorized video tutorials covering intermediate and advanced features. Clicking on any tutorial thumbnail will launch the tutorial video overlay, walking through detailed steps related to that specific feature or concept.

Associated Functional Requirements:

1. As a user, I would like to have access to documentation and tutorials so that I can learn more about the platform's features.

Feedback and Support Page:



This is the feedback and support page of the application. It comes up when the user clicks on their user icon in the top right and selects 'Help & Support' from the dropdown menu. There is a feedback form where the user can select the feedback type from a dropdown menu, fill in details about their issue or suggestion, and click submit. This will open a ticket with the support team,

which the user can track via status notifications. Below this, there is a searchable FAQ knowledge base to find help articles related to common questions. At the bottom, there is contact info for getting in direct touch with customer support agents.

Associated Functional Requirements:

1. As a user, I would like to have a feedback option so that I can report issues or suggest improvements.