CS3009 - Software Engineering - Spring 2025

VRITE | COLLABORATIVE EDITOR

Deliverable 3 - Final Deliverable

Team Members:

1. Tabish Noman Khan i220831
2. Muhammad Danish Haroon i220955
3. Arqam Zia i221170

Submission Date: 28-04-25

Table of Contents

1. Final Report

1. Introduction:

Problem Statement:

The problem of	the lack of a unified, real-time collaborative editor
Affects	Developers, writers, and teams who require flexible and private collaboration tools
The impact of which is	A gap for a unified tool that handles diverse content types in a single, shareable document, along with concerns about data security and privacy, lack of robust version control, and limited self-hosting options
A successful solution would be	Vrite, an open-source, self-hosted collaborative editor with real-time collaboration and rich text formatting. Users can host it on their own systems, ensuring full control over data and seamless collaboration.

Project Description:

Vrite is a self-contained collaborative editor that can be self-hosted by users on their own infrastructure. It aims to address the gap in the market for a unified, real-time collaborative editor that supports multiple content types with a focus on data privacy and security. Vrite is designed to function independently but can be integrated with existing systems through web links and export options

Roles and Responsibilites:

1. Tabish (Scrum Master)

Duties:

- Facilitates Scrum ceremonies (Daily Standups, Sprint Planning, Review, and Retrospective).
- Ensures the team follows Agile best practices.
- Removes any obstacles blocking the team’s progress.
- Acts as a bridge between the team and stakeholders.
- Helps improve team efficiency and productivity.

2. Arqam (Product Owner)

Duties:

- Defines and prioritizes the product backlog.
- Works closely with stakeholders to gather requirements.
- Ensures that the development team understands user needs.
- Accepts or rejects completed work based on requirements.
- Makes strategic decisions to maximize product value.

3. Danish (Scrum Team - Developer)

Duties:

- Implements features and fixes bugs as per sprint goals.
- Collaborates with the team to ensure high-quality code.
- Actively participates in code reviews and testing.
- Updates progress in Scrum meetings and adapts to feedback.
- Continuously improves the product based on iterations.

2. Functional and Non-Functional Requirements

2.1 Functional Requirements

2.1.1 Real-time Collaborative Editing

- FR-1.1: The system shall allow multiple users to edit the same document simultaneously.
- FR-1.2: The system shall update the document in real-time for all connected users.
- FR-1.3: The system shall show cursor positions of other users in the document.
- FR-1.4: The system shall handle conflict resolution when multiple users edit the same section.

2.1.4 Self-hosting

- FR-4.1: The system shall provide installation instructions for self-hosting.
- FR-4.2: The system shall store all data on the user's own server.
- FR-4.3: The system shall provide backup and restore functionality.
- FR-4.4: The system shall be configurable for different server environments.

2.1.5 Document Sharing

- FR-5.1: The system shall generate unique URLs for document access.
- FR-5.2: The system shall allow setting permission levels for shared documents.
- FR-5.3: The system shall allow revoking access to shared documents.
- FR-5.4: The system shall notify document owners when shared documents are accessed.

2.1.6 User Authentication

- FR-6.1: The system shall require user registration with email verification.
- FR-6.2: The system shall provide secure login functionality.
- FR-6.3: The system shall allow password reset.
- FR-6.4: The system shall provide multi-factor authentication options.

2.1.7 Role-based Access Control

- FR-7.1: The system shall support different user roles (Viewer, Editor, Admin).
- FR-7.2: The system shall restrict document actions based on user roles.
- FR-7.3: The system shall allow administrators to modify user roles.
- FR-7.4: The system shall log access and modifications for audit purposes.

2.1.8 Offline Editing

- FR-8.1: The system shall allow users to continue editing when offline.
- FR-8.2: The system shall automatically synchronize changes when reconnected.
- FR-8.3: The system shall detect and handle conflicts during synchronization.
- FR-8.4: The system shall provide an indicator of offline status.

2.1.9 Document Export

- FR-9.1: The system shall allow exporting documents in multiple formats (PDF, DOCX, etc.).
- FR-9.2: The system shall preserve formatting when exporting.
- FR-9.3: The system shall provide options to customize export settings.
- FR-9.4: The system shall notify users when export is complete.

2.1.10 Real-time Notifications

- FR-10.1: The system shall notify users when documents are edited by others.
- FR-10.2: The system shall allow configuring notification preferences.
- FR-10.3: The system shall provide visual indicators for document activities.
- FR-10.4: The system shall display user presence information.

2.2 Non-Functional Requirements

2.2.1 Product Requirements

2.2.1.1 Usability

- NFR-U1: The system shall have an intuitive, clean user interface.
- NFR-U2: The system shall provide keyboard shortcuts for common actions.
- NFR-U3: The system shall include comprehensive help documentation.
- NFR-U4: The system shall be accessible on both desktop and mobile devices.

2.2.1.2 Performance

- NFR-P1: The system shall update collaborative changes within 500ms.
- NFR-P2: The system shall support up to 20 simultaneous users per document.
- NFR-P3: The system shall automatically save changes at least every 30 seconds.
- NFR-P4: The system shall maintain responsiveness for documents up to 1MB in size.

2.2.1.3 Reliability

- NFR-R1: The system shall recover from crashes with minimal data loss.
- NFR-R2: The system shall maintain a backup of all document versions.
- NFR-R3: The system shall have an uptime of at least 99.5%.
- NFR-R4: The system shall handle network interruptions gracefully.

2.2.1.4 Security

- NFR-S1: The system shall encrypt all data in transit and at rest.
- NFR-S2: The system shall implement secure authentication practices.
- NFR-S3: The system shall validate all user inputs to prevent injection attacks.
- NFR-S4: The system shall detect and alert unauthorized access attempts.

2.2.2 Organizational Requirements

2.2.2.1 Development Standards

- NFR-D1: The system shall be developed using industry-standard coding practices.
- NFR-D2: The system shall include comprehensive test coverage.
- NFR-D3: The system shall follow a consistent code style and documentation format.
- NFR-D4: The system shall use semantic versioning for releases.

2.2.2.2 Operational Requirements

- NFR-O1: The system shall include deployment documentation for various environments.
- NFR-O2: The system shall provide logging for troubleshooting.
- NFR-O3: The system shall allow configuration without code changes.
- NFR-O4: The system shall provide health monitoring endpoints.

2.2.3 External Requirements

2.2.3.1 Ethical Requirements

- NFR-E1: The system shall respect user privacy by not collecting unnecessary data.
- NFR-E2: The system shall provide clear information about data storage and usage.
- NFR-E3: The system shall allow users to delete their data completely.
- NFR-E4: The system shall comply with relevant accessibility standards.

2.2.3.2 Legal Requirements

- NFR-L1: The system shall comply with data protection regulations.
- NFR-L2: The system shall use only properly licensed third-party components.
- NFR-L3: The system shall provide appropriate licensing information.
- NFR-L4: The system shall include terms of service and privacy policy documents.

Features List

1. Real-time Collaborative Editing: Multiple users can edit the same document simultaneously.
2. Self-hosting Capability: Users can host the server on their own infrastructure, ensuring full control over their data.
3. Web-Tunneled Links for Document Access: Unique URLs for documents that are shareable across different client instances.
4. Role-based Access Control: Different levels of access for collaborators (e.g., Viewer, Editor, Admin).
5. Document Sharing: Shareable links for users to collaborate on documents from different locations and devices.
6. Multiple Clients per Server: One server can host multiple documents, with each document accessible via a unique link.
7. User Authentication: Secure user login and management system to maintain privacy.
8. Real-time Notifications: Alert users when someone else is editing or when a document has been saved or updated.
9. Offline Editing Mode: Allows users to continue editing even if they temporarily lose internet connection.
10. Document Export: Option to export documents in different formats (e.g., PDF, Word, Markdown).
11. Mobile and Desktop Support: Responsive design to allow users to access the editor on both desktop and mobile devices.

User Stories:

1. Real-Time Collaborative Editing

As a document collaborator, I want to edit a document in real-time with other users so that we can all contribute simultaneously.

Acceptance Criteria:

Scenario: Multiple users editing the same document simultaneously
Given two or more users are viewing the same document
When one user makes a change to the document
Then other users should see those changes appear in real-time
And all changes should be preserved accurately

2. Multiple Editing Modes

As a user, I want to switch between different modes (Markdown, WYSIWYG, Code, etc.) within a document so that I can format my work according to its type.

Acceptance Criteria:

Scenario: Switching between editing modes
Given I am working on a document in Markdown mode
When I switch to WYSIWYG mode
Then my content should be preserved and formatted correctly in the new mode
And I should be able to continue editing without any loss of data

Scenario: Maintaining content integrity across modes
Given I have created complex formatting in one mode
When I switch to another mode and back
Then my formatting should remain intact

3. Document Sharing via Web Links

As a user, I want to be able to easily share a document with others via a web link so that my collaborators can access and edit it from anywhere.

Acceptance Criteria:

Scenario: Sharing document via link
Given I have a document I want to share
When I generate a shareable link
Then the recipient should be able to access the document using that link

Scenario: Setting link permissions
Given I am creating a shareable link
When I set the link permission to "Edit" or "View Only"
Then recipients of the link should have only the specified level of access

4. Rich Text Formatting

As a content creator, I want to be able to use rich text formatting to style my document without coding knowledge.

Acceptance Criteria:

Scenario: Applying text formatting
Given I am editing a document in rich text mode
When I select text and apply formatting (bold, italic, heading, etc.)
Then the text should display with the applied formatting

Scenario: Using formatting toolbar
Given I am in rich text editing mode
When I click on formatting options in the toolbar
Then the corresponding formatting should be applied to selected text or at cursor position

5. Self-Hosted Document Storage

As a user, I want my document to be stored locally on my own server so that I maintain control over my data.

Acceptance Criteria:

Scenario: Setting up self-hosted storage
Given I have installed the Vrite server software on my own hardware
When I create and save a document
Then the document should be stored on my local server
And not be sent to any third-party cloud storage

Scenario: Data persistence
Given I have stored documents on my self-hosted server
When the server restarts
Then all my documents and their version histories should remain intact

6. Real-Time Notifications

As a collaborator, I want to receive real-time notifications when someone makes changes to the document so I can stay up to date.

Acceptance Criteria:

Scenario: Receiving edit notifications
Given I have access to a shared document
When another user makes changes to the document
Then I should receive a notification about the change
And the notification should include who made the change

Scenario: Notification preferences
Given I am in my user settings
When I update my notification preferences
Then I should only receive the types of notifications I've opted into

7. Document Export Options

As a user, I want to have the ability to export my document in multiple formats (PDF, DOCX, etc.) so I can share it with others who may not use Vrite.

Acceptance Criteria:

Scenario: Exporting to PDF
Given I have a completed document
When I select "Export as PDF" from the export menu
Then a PDF version of my document should be generated and downloaded
And the PDF should accurately reflect the formatting of my document

Scenario: Exporting to other formats
Given I have a document I want to share
When I select "Export as DOCX" or other available formats
Then the document should be converted to the selected format while preserving formatting

8. Organisation Access Control

As an administrator, I want to control who can access a organisation and its documents to ensure privacy and security.

Acceptance Criteria:

Scenario: Managing organisation users
Given I am logged in as a server administrator
When I add a new user to the organisation
Then that user should be able to access the server with the permissions I've assigned

Scenario: Restricting organisational access
Given I am logged in as a organisation administrator
When I remove a user's access to the server
Then that user should no longer be able to log in or access any documents on the organisation

9. Automatic Document Saving

As a user, I want my document to automatically save changes to prevent data loss.

Acceptance Criteria:

Scenario: Auto-saving during editing
Given I am actively editing a document
When I make changes to the document
Then the changes should be automatically saved at regular intervals
And I should see an indicator showing when the last auto-save occurred

Scenario: Recovery after crash
Given I was editing a document
When the application or browser crashes
Then upon reopening, I should find my document with minimal or no data loss

10. Offline Editing Capabilities

As a user, I want the editor to be able to work offline so that I can continue editing even without an internet connection.

Acceptance Criteria:

Scenario: Editing while offline
Given I have the document open in the editor
When my internet connection is lost
Then I should still be able to make and save changes locally

Scenario: Syncing after reconnection
Given I have made changes while offline
When my internet connection is restored
Then my changes should automatically sync to the server
And conflicts with other users' changes should be handled appropriately

11. Document Backup System

As a developer, I want to have a backup system in place to store documents in case of server failure.

Acceptance Criteria:

Scenario: Scheduled backups
Given I have configured backup settings
When the scheduled backup time arrives
Then the system should create a backup of all documents and their histories
And store it in the designated backup location

Scenario: Backup restoration
Given a server failure has occurred
When I initiate the restoration process from a backup
Then all documents and version histories should be restored to the state they were in at the time of backup

12. Multiple Document Management

As a server owner, I want to have the flexibility to add and manage multiple documents on a single server.

Acceptance Criteria:

Scenario: Creating multiple documents
Given I am logged into my server
When I create multiple documents
Then all documents should be stored and accessible from my server dashboard

Scenario: Organizing documents
Given I have multiple documents on my server
When I create folders and move documents between them
Then the documents should be organized according to my folder structure

13. Clean User Interface

As a user, I want a clean and simple user interface to focus on content creation without distractions.

Acceptance Criteria:

Scenario: Distraction-free mode
Given I am editing a document
When I activate distraction-free mode
Then all non-essential UI elements should be hidden
And only the document content should remain visible

Scenario: Responsive design
Given I access the editor on different devices
When I view and edit documents
Then the interface should adapt to my screen size without sacrificing usability

14. Help Documentation Access

As a user, I want to have quick access to help documentation or support if I encounter any issues while using Vrite.

Acceptance Criteria:

Scenario: Accessing help documentation
Given I am using the editor
When I click on the help icon
Then a comprehensive help documentation should open
And I should be able to search for specific topics

Scenario: Contextual help
Given I am using a specific feature
When I request help for that feature
Then I should see documentation specifically relevant to what I'm doing

1

5. Real-Time Collaborator Presence

As a collaborator, I want to see who else is working on the document in real-time so I can avoid editing the same section simultaneously.

Acceptance Criteria:

Scenario: Viewing active collaborators
Given multiple users are editing the same document
When I look at the collaborator panel
Then I should see a list of all currently active users
And their cursor positions in the document

Scenario: Section collision avoidance
Given another user is editing a specific section
When I navigate to that section
Then I should see a visual indicator showing who is working there

16. Self-Hosted Environment Setup

As a server administrator, I want to set up a self-hosted environment to have full control over security and privacy.

Acceptance Criteria:

Scenario: Following installation guide
Given I have the installation documentation
When I follow the step-by-step instructions
Then I should successfully install and configure the server
And all components should work correctly

Scenario: Custom configuration
Given I am setting up a self-hosted environment
When I modify configuration parameters
Then the server should operate according to my custom settings

17. Cross-Device Synchronization

As a user, I want a document to automatically sync changes across all devices when I log in to a new device.

Acceptance Criteria:

Scenario: Accessing documents on a new device
Given I have edited a document on one device
When I log in to my account on a different device
Then I should see the latest version of my documents
And all my changes should be synchronized

Scenario: Conflict resolution
Given I have edited a document on multiple devices while offline
When I reconnect to the internet
Then the system should detect conflicts
And provide a way to resolve them

18. AI-content assistance

As a content creator, I want an option to ask an AI for support that has access to the entire document to help me create or suggest content.

Acceptance Criteria:

Scenario: Asking AI for content support Given I am editing a document and need help generating content When I activate the AI assistance feature and provide a prompt or context, potentially referencing the existing document content Then the AI should generate relevant content suggestions or text that I can review and insert into my document

Product Backlog

1. Real-Time Collaborative Editing

   - Priority: High
   - User Story: As a document collaborator, I want to edit a document in real-time with other users so that we can all contribute simultaneously.
   - Acceptance Criteria: Ensure real-time syncing, preserve all changes accurately, and handle simultaneous edits.

2. Multiple Editing Modes

   - Priority: High
   - User Story: As a user, I want to switch between different modes (Markdown, WYSIWYG, Code, etc.) within a document so that I can format my work according to its type.
   - Acceptance Criteria: Implement different editing modes with seamless switching and formatting preservation.

3. Document Sharing via Web Links

   - Priority: High
   - User Story: As a user, I want to be able to easily share a document with others via a web link so that my collaborators can access and edit it from anywhere.
   - Acceptance Criteria: Ensure creation of shareable links with editable and view-only access.

4. Rich Text Formatting

   - Priority: Medium
   - User Story: As a content creator, I want to be able to use rich text formatting to style my document without coding knowledge.
   - Acceptance Criteria: Implement a toolbar for rich text editing (bold, italic, headings, etc.) and real-time formatting.

5. Self-Hosted Document Storage

    - Priority: High
    - User Story: As a user, I want my document to be stored locally on my own server so that I maintain control over my data.

    - Acceptance Criteria: Provide local document storage on user's server with data persistence and security.

6. Real-Time Notifications

    - Priority: Medium
    - User Story: As a collaborator, I want to receive real-time notifications when someone makes changes to the document so I can stay up to date.
    - Acceptance Criteria: Implement push notifications on changes with user preference settings.

7. Document Export Options

    - Priority: Medium
    - User Story: As a user, I want to have the ability to export my document in multiple formats (PDF, DOCX, etc.) so I can share it with others who may not use Vrite.
    - Acceptance Criteria: Implement exporting features to PDF, DOCX, and other popular formats.

8. Organisation Access Control

    - Priority: High
    - User Story: As an administrator, I want to control who can access an organisation and its documents to ensure privacy and security.
    - Acceptance Criteria: Allow control of organizational access and permission management.

9. Automatic Document Saving

    - Priority: High
    - User Story: As a user, I want my document to automatically save changes to prevent data loss.
    - Acceptance Criteria: Implement auto-saving functionality and recovery options.

10. Offline Editing Capabilities

- Priority: Medium
- User Story: As a user, I want the editor to be able to work offline so that I can continue editing even without an internet connection.
- Acceptance Criteria: Enable offline editing with local storage and auto-sync when reconnected.

11. Document Backup System

- Priority: Low
- User Story: As a developer, I want to have a backup system in place to store documents in case of server failure.
- Acceptance Criteria: Implement backup scheduling and restoration from backups.

12. Multiple Document Management

- Priority: High
- User Story: As a server owner, I want to have the flexibility to add and manage multiple documents on a single server.
- Acceptance Criteria: Allow document creation, storage, and organization through a server dashboard.

13. Clean User Interface

- Priority: Medium
- User Story: As a user, I want a clean and simple user interface to focus on content creation without distractions.
- Acceptance Criteria: Implement distraction-free mode and responsive design across devices.

14. Help Documentation Access

- Priority: Low
- User Story: As a user, I want to have quick access to help documentation or support if I encounter any issues while using Vrite.
- Acceptance Criteria: Provide comprehensive and context-sensitive help documentation.

15. Real-Time Collaborator Presence

- Priority: Medium
- User Story: As a collaborator, I want to see who else is working on the document in real-time so I can avoid editing the same section simultaneously.
- Acceptance Criteria: Implement a collaborator panel and section collision avoidance features.

16. Self-Hosted Environment Setup

- Priority: High
- User Story: As a server administrator, I want to set up a self-hosted environment to have full control over security and privacy.
- Acceptance Criteria: Provide installation and configuration guides, including custom configurations.

17. Cross-Device Synchronization

- Priority: Medium
- User Story: As a user, I want a document to automatically sync changes across all devices when I log in to a new device.
- Acceptance Criteria: Implement automatic synchronization of changes and conflict resolution when switching devices.

18. AI Content Assistance

- Priority: Medium
- User Story: As a content creator, I want an option to ask an AI for support that has access to the entire document to help me create or suggest content.
- Acceptance Criteria: Scenario: Asking AI for content support Given I am editing a document and need help generating content, When I activate the AI assistance feature and provide a prompt or context, Then the AI should generate relevant content suggestions or text that I can review and insert into my document

Sprint Backlog:

Sprint 1 Backlog:

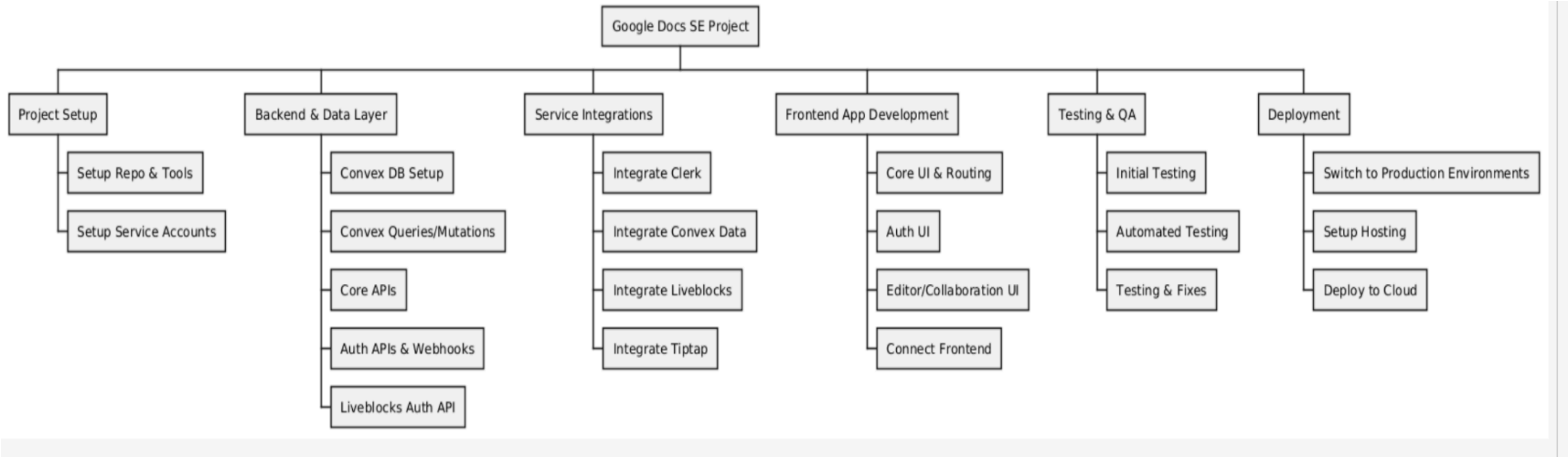
Task	Priority
Initial Project Architecture Setup	High
Basic Document Creation & Editing	High
Real-time Collaboration: Basic Typing Synchronization	High
User Authentication and Registration - Basic Signup/Login	High
Simple Document Saving (Local Storage Backup)	Medium

Sprint 2 Backlog:

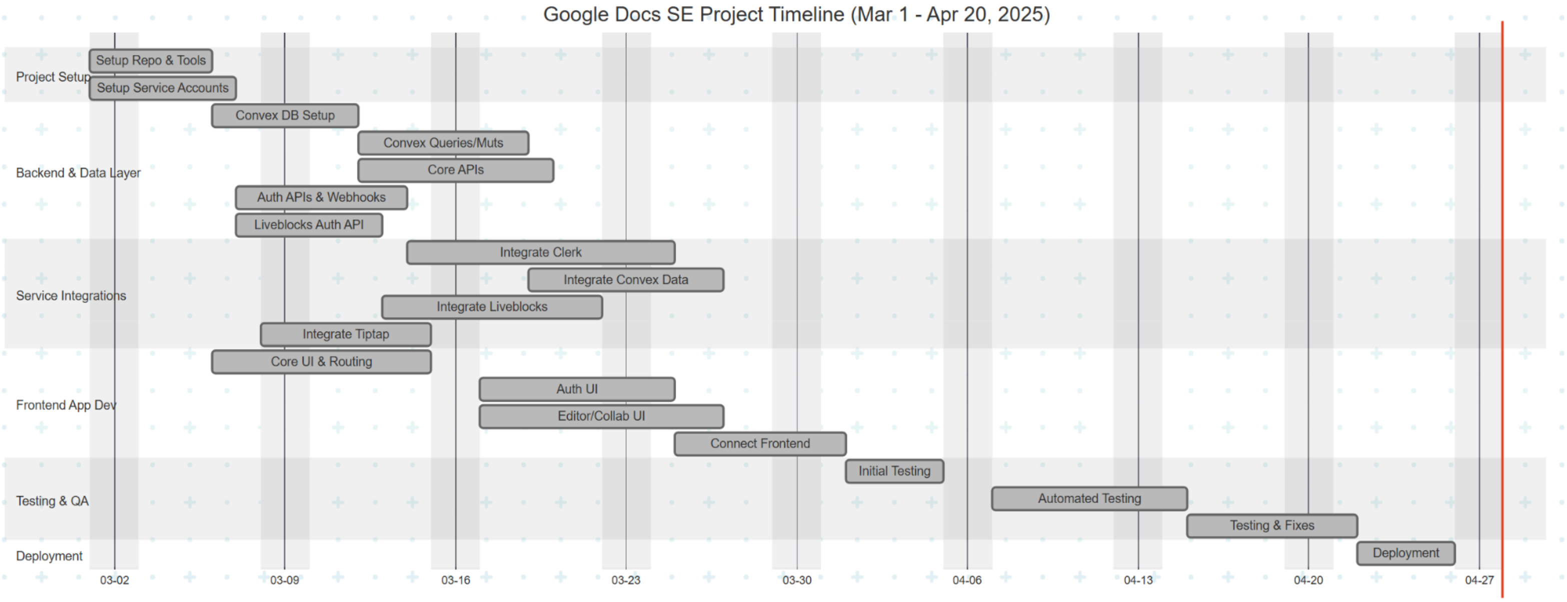
Task	Priority
Version Control	**High**
Document Export	**High**
Offline Mode	**High**
Share Document (Basic Link Sharing)	**Medium**
View Mode (Read-Only Access)	**Medium**
Toolbar	**Medium**
Organization Management	**Medium**

Software Project Plan

Work Breakdown Structure:

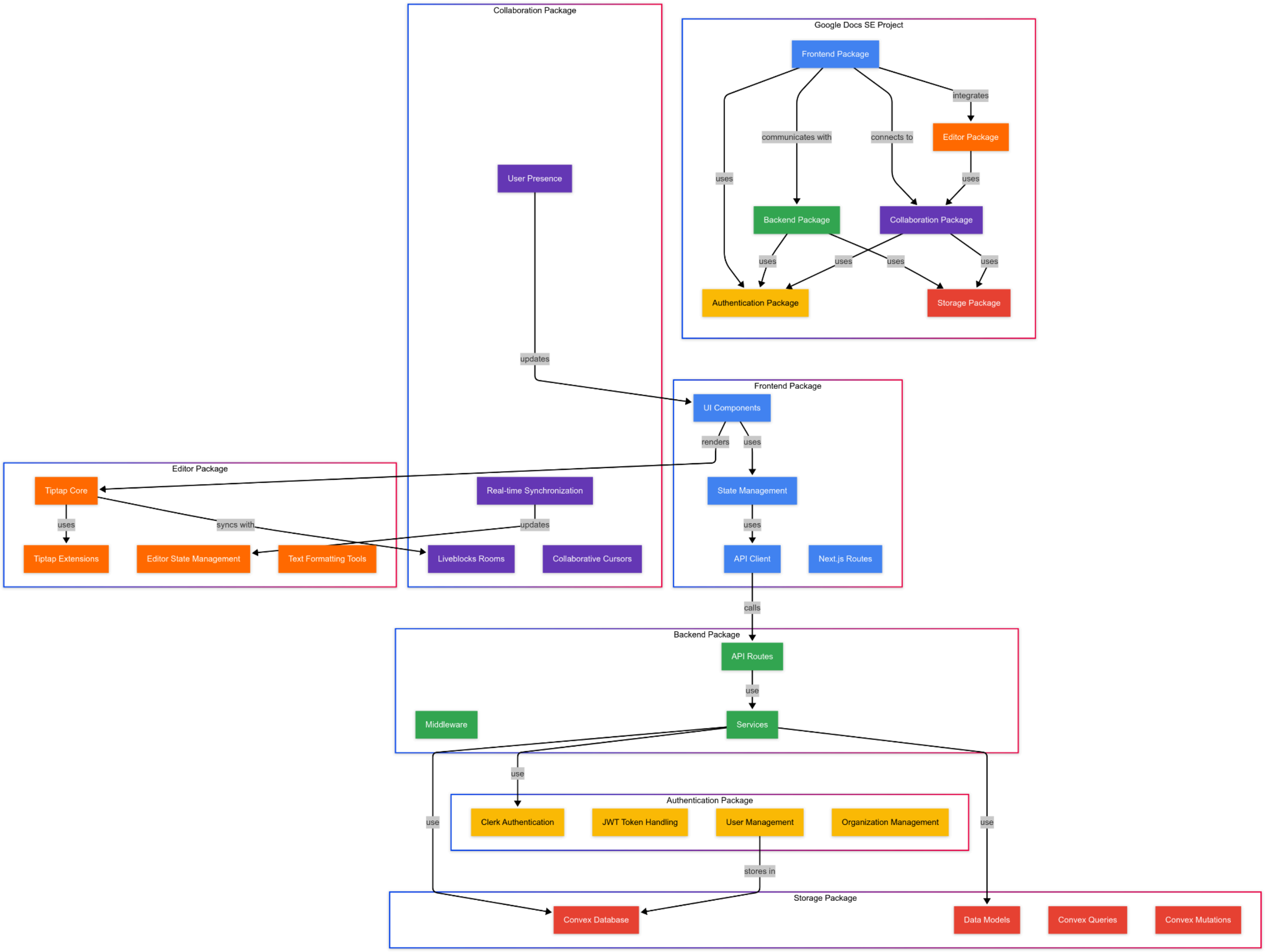


Gantt-Chart:


Google Docs SE Project Timeline (Mar 1 - Apr 20, 2025)
Project Setup
Setup Repo & Tools
Setup Service Accounts
Backend & Data Layer
Convex DB Setup
Convex Queries/Muts
Core APIs
Auth APIs & Webhooks
Liveblocks Auth API
Service Integrations
Integrate Clerk
Integrate Convex Data
Integrate Liveblocks
Integrate Tiptap
Frontend App Dev
Core UI & Routing
Auth UI
Editor/Collab UI
Connect Frontend
Testing & QA
Initial Testing
Automated Testing
Testing & Fixes
Deployment
Deployment
03-02
03-09
03-16
03-23
03-30
04-06
04-13
04-20
04-27

System Architecture:

1. UML Package Diagram:



Frontend Package

- Manages the user interface (UI), client-side logic, and responsive components for document editing/management.
- Handles application routing, state management, API communication, user interactions, and real-time updates.

Backend Package

- Manages server-side logic,API endpoints, and implements integrations.
- Processes incoming requests, coordinates between subsystems, performs data validation, and handles errors.

Authentication Package

- Manages user authentication via Clerk, handling JWT token generation, validation, and session management.
- Controls user profiles, permissions, organization memberships, and ensures secure access to system resources.

Storage Package

- Utilizes Convex for persistent data storage and real-time data synchronization.
- Defines data models/schemas, implements data operations (queries/mutations), and ensures data integrity.

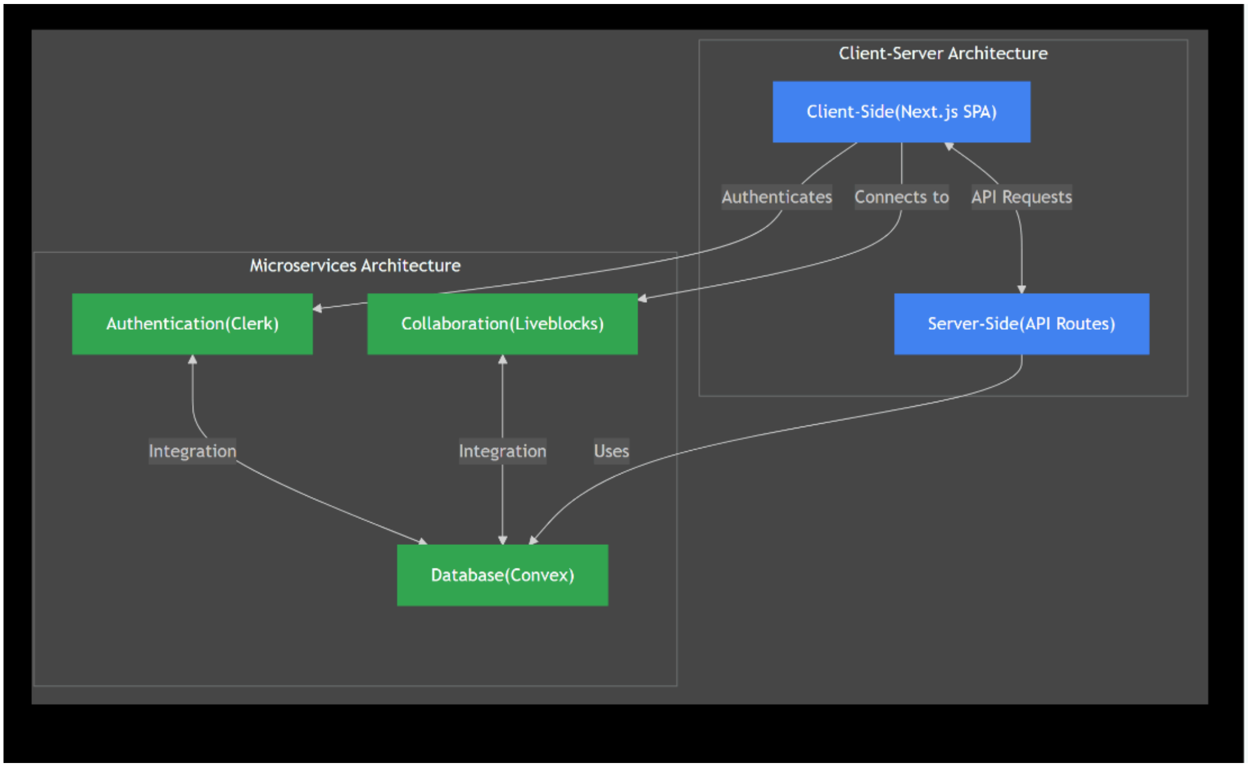
Collaboration Package

- Leverages Liveblocks to enable real-time collaborative features and manage multi-user document editing rooms.
- Tracks user presence, cursor positions, editing status, and synchronizes document changes between connected clients.

Editor Package

- Implements rich text editing using Tiptap, providing features for text formatting, styling, and document structure.

2. Architectural Diagram:



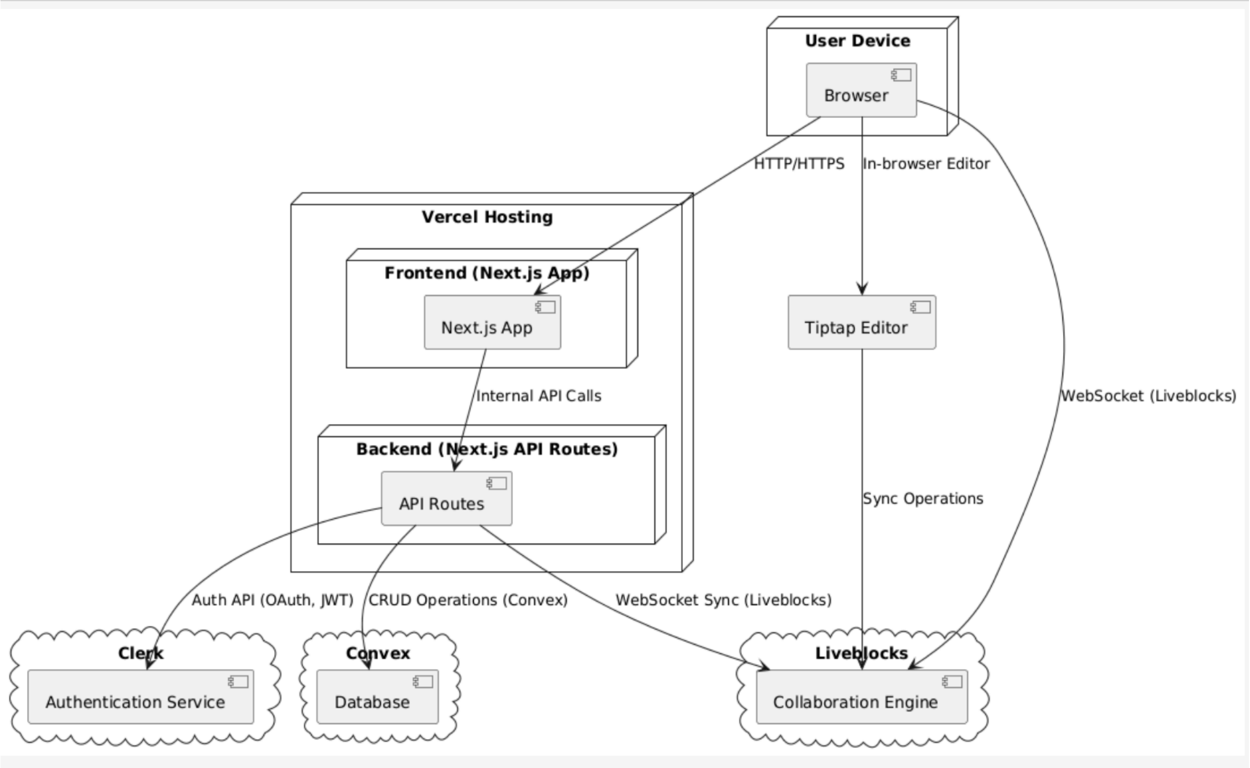
Client-Server:

- Next.js frontend (client) interacts with API routes (server).
- *Rationale:* Separates UI/user interaction from data/business logic for independent development and scaling.

Microservices (via External Services):

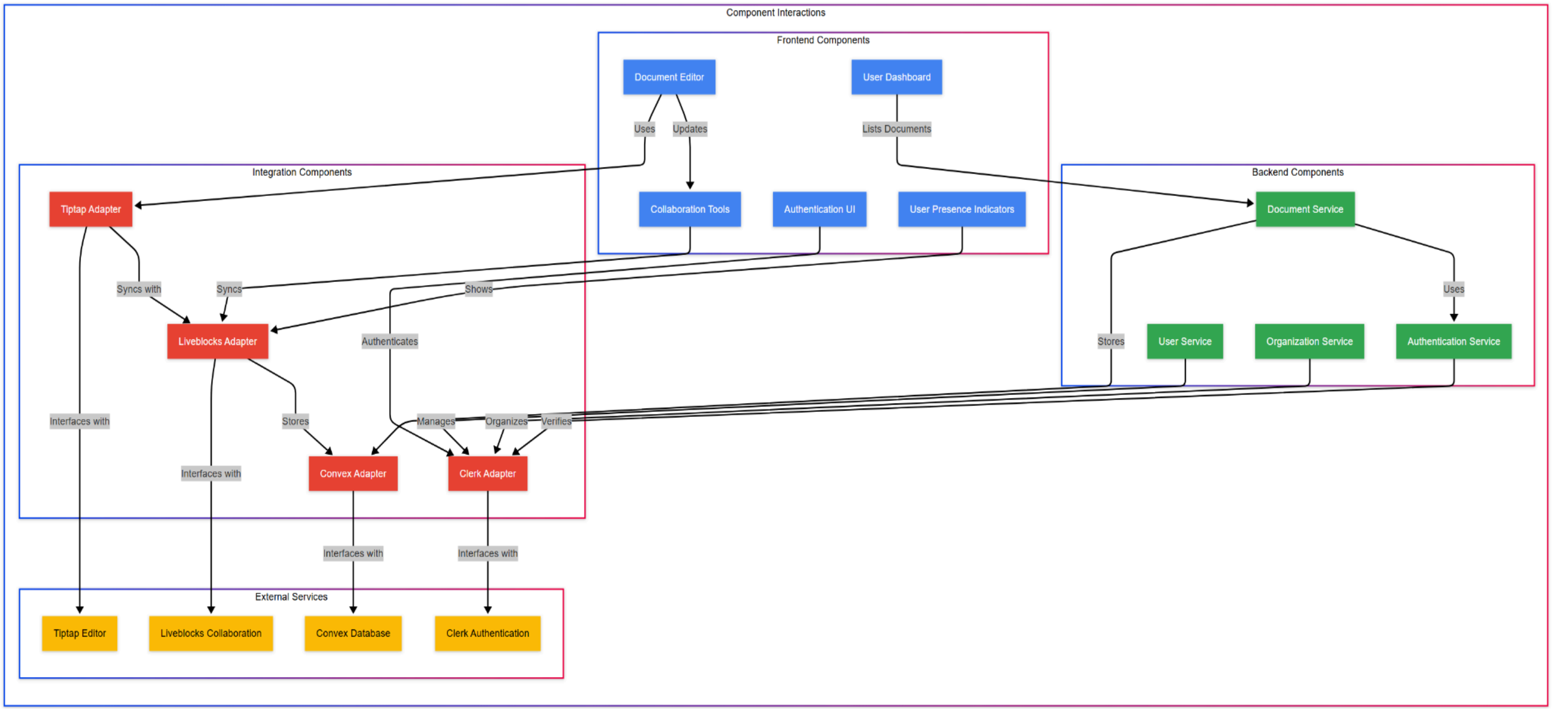
- Integrates with specialized services (Clerk, Convex, Liveblocks
- *Rationale:* Leverages best-in-class external solutions for reduced development and maintenance.

3.Deployment Diagram for Client Deployments



- Web Browser: Runs the client application, handling UI and user interactions.
- Vercel Platform: Hosts the Next.js app, serving assets and running serverless functions
- Clerk Authentication: Manages user identity, authentication, and secure access via WTs.
- Liveblocks Service: Provides real-time collaboration features, syncing presence and document state.
- Convex Database: Stores application data with real-time sync and provides data operations.

4. Component Diagram:



Frontend Components:

- Document Editor: The main interface for editing documents, built using Tiptap.
- Authentication UI: Handles user login, registration, and profile management.
- Dashboard: Displays user's documents and organization information.
- Collaboration Tools: Provides features for real-time collaboration.
- User Presence Indicators: Shows which users are currently viewing/editing a document.

Backend Components:

- Authentication Service: Manages user authentication and authorization.
- Document Service: Handles document CRUD operations and metadata.
- User Service: Manages user profiles and preferences.
- Organization Service: Handles organization membership and permissions.

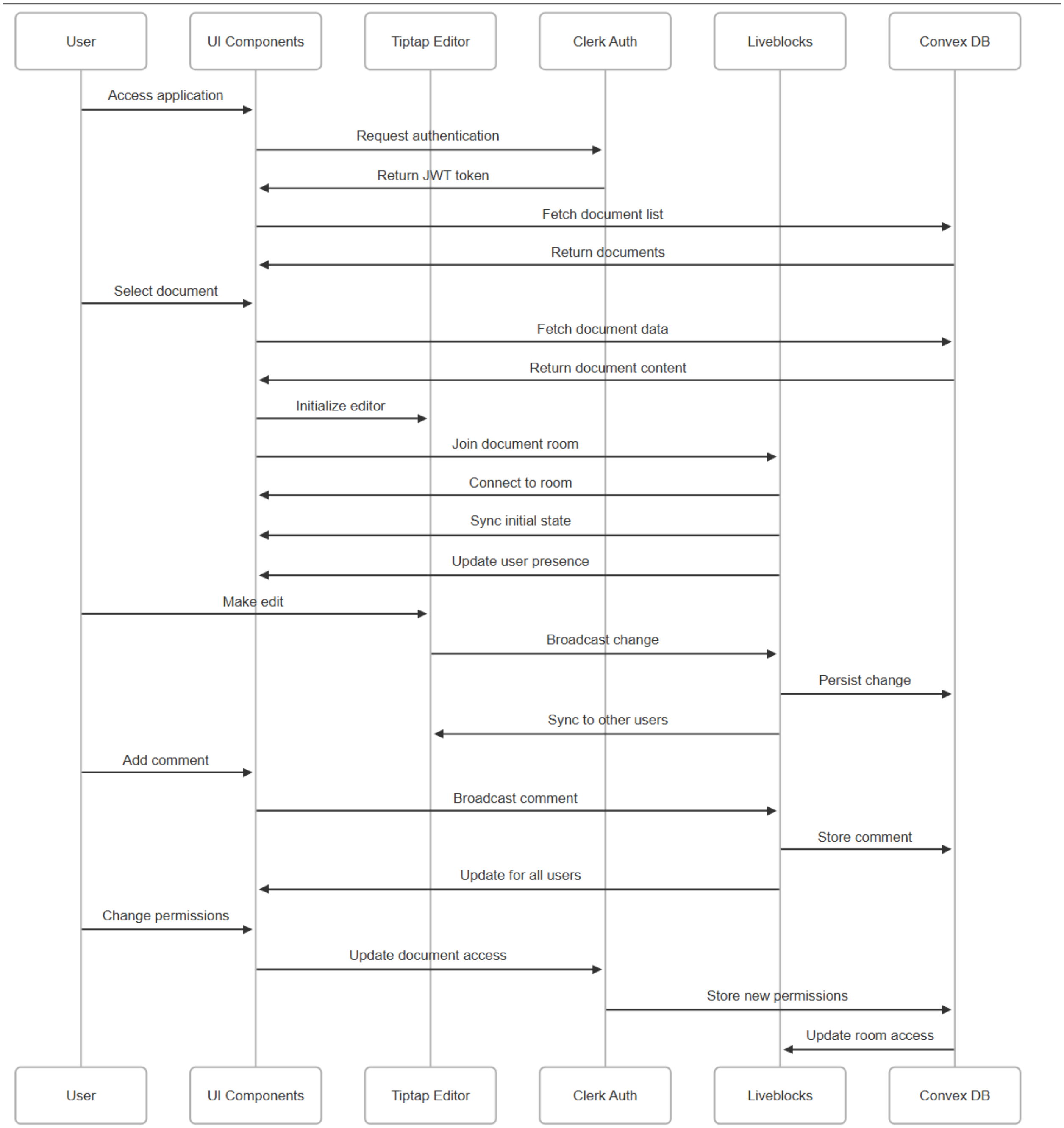
External Services:

- Clerk Authentication: Provides identity management and authentication.
- Convex Database: Stores and synchronizes application data.
- Liveblocks Collaboration: Enables real-time collaborative features.
- Tiptap Editor: Provides rich text editing capabilities.

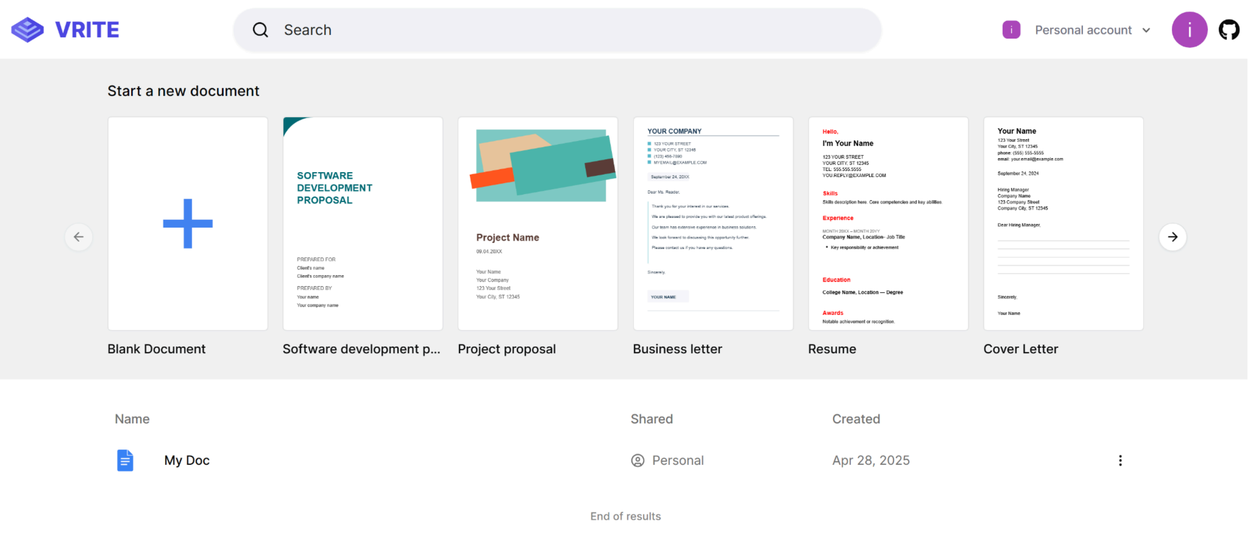
Integration Components:

- Clerk Adapter: Interfaces with Clerk for authentication.
- Convex Adapter: Interfaces with Convex for data storage.
- Liveblocks Adapter: Interfaces with Liveblocks for collaboration.
- Tiptap Adapter: Interfaces with Tiptap for editing.

Component Interaction Overview:



Actual Implementation Screenshots



Start a new document

Blank Document

Software development p...

Project proposal

Business letter

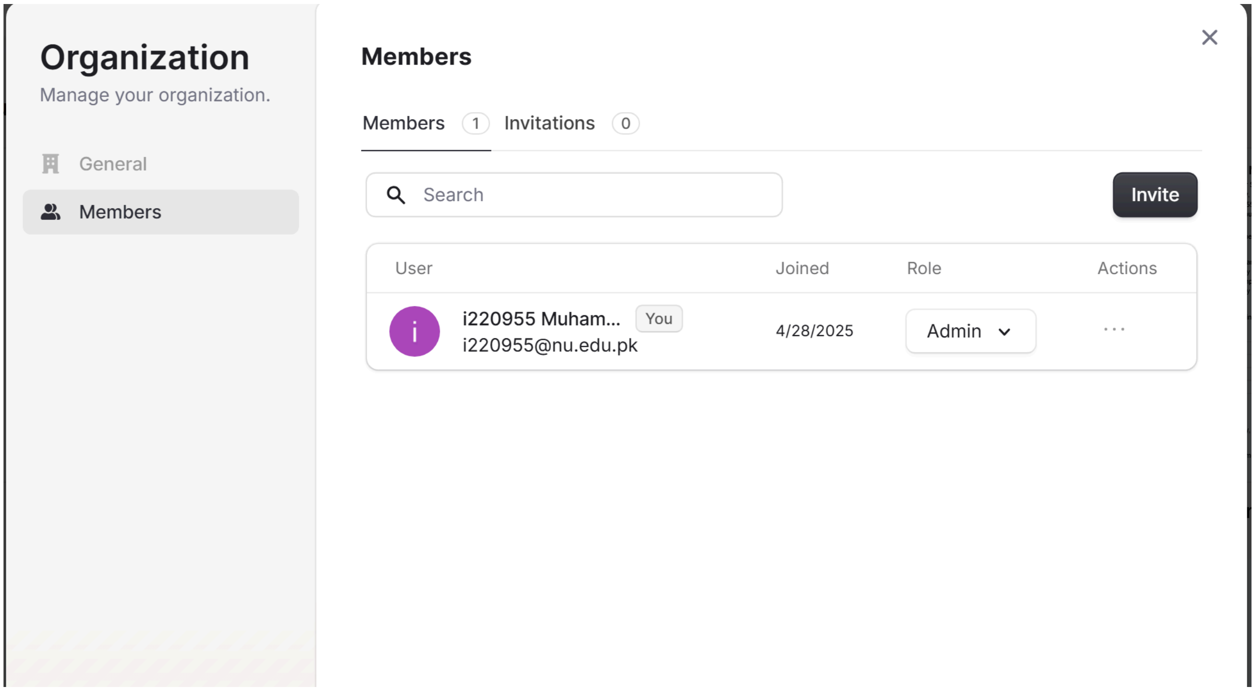
Resume

Cover Letter

Name	Shared	Created
My Doc	Personal	Apr 28, 2025

End of results

Organisations



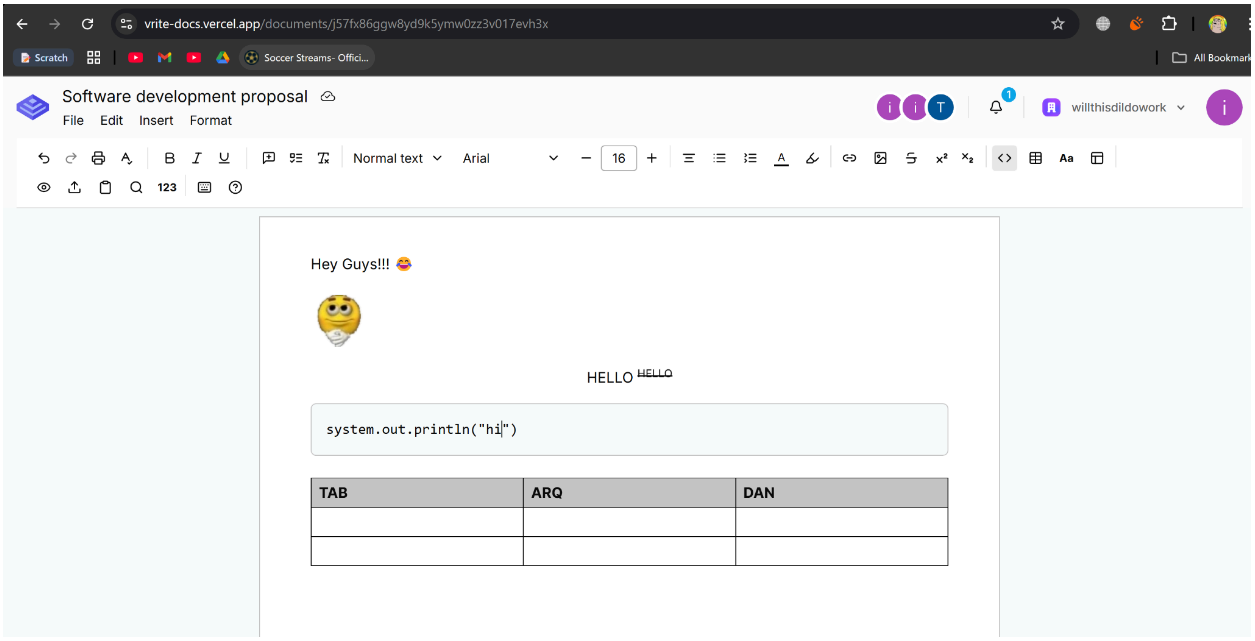
Organization

Manage your organization.

Members

User	Joined	Role	Actions
i220955 Muham... You i220955@nu.edu.pk	4/28/2025	Admin	···

Editor Interface



vrite-docs.vercel.app/documents/j57fx86ggw8yd9k5ymw0zz3v017evh3x
Scratch
Soccer Streams- Offici...
All Bookmarks
Software development proposal
File
Edit
Insert
Format
i
i
T
1
willthisdildowork
i
Normal text
Arial
16
123
Aa

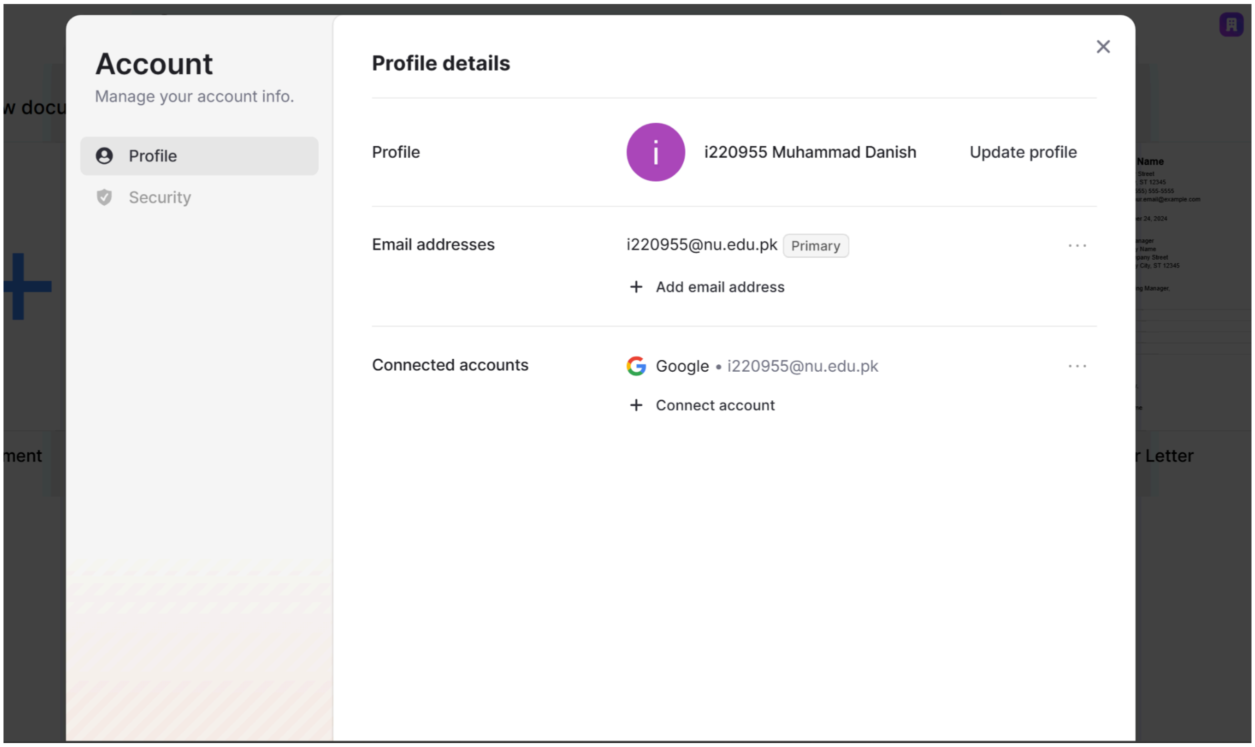
Hey Guys!!! 😂

HELLO ~~HELLO~~

```
system.out.println("hi")
```

TAB	ARQ	DAN

Account Management



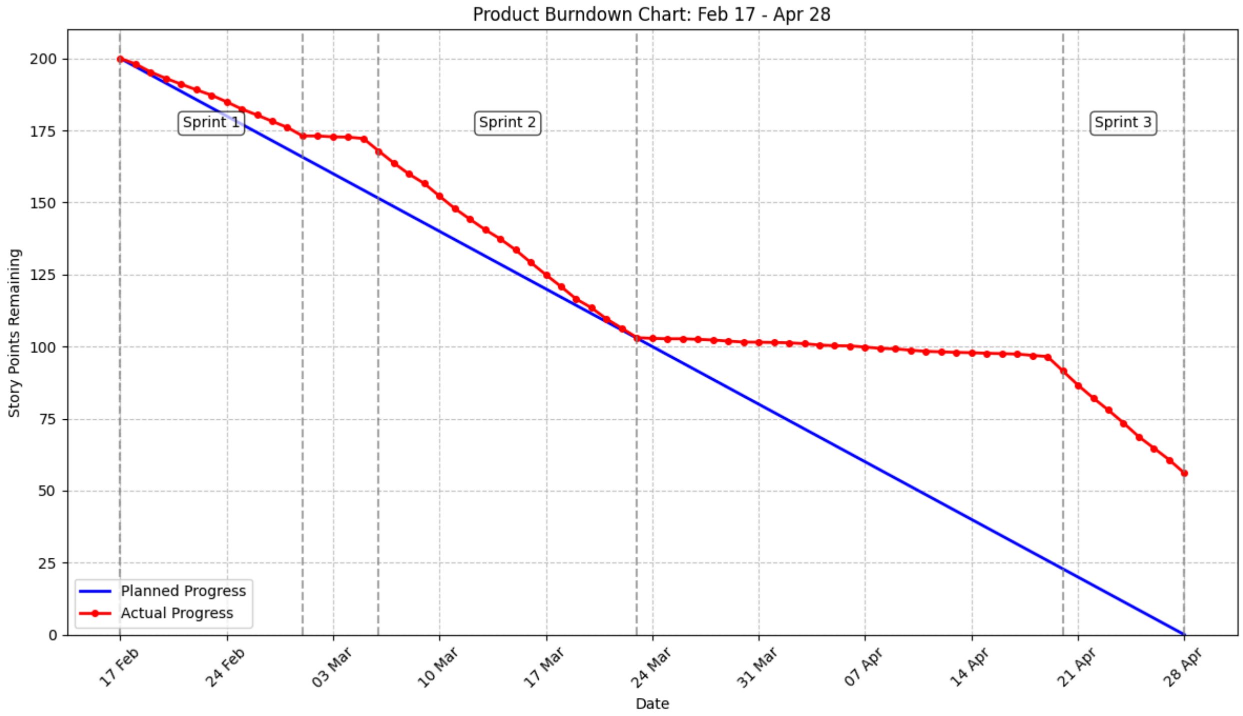
Profile details

Profile: i i220955 Muhammad Danish

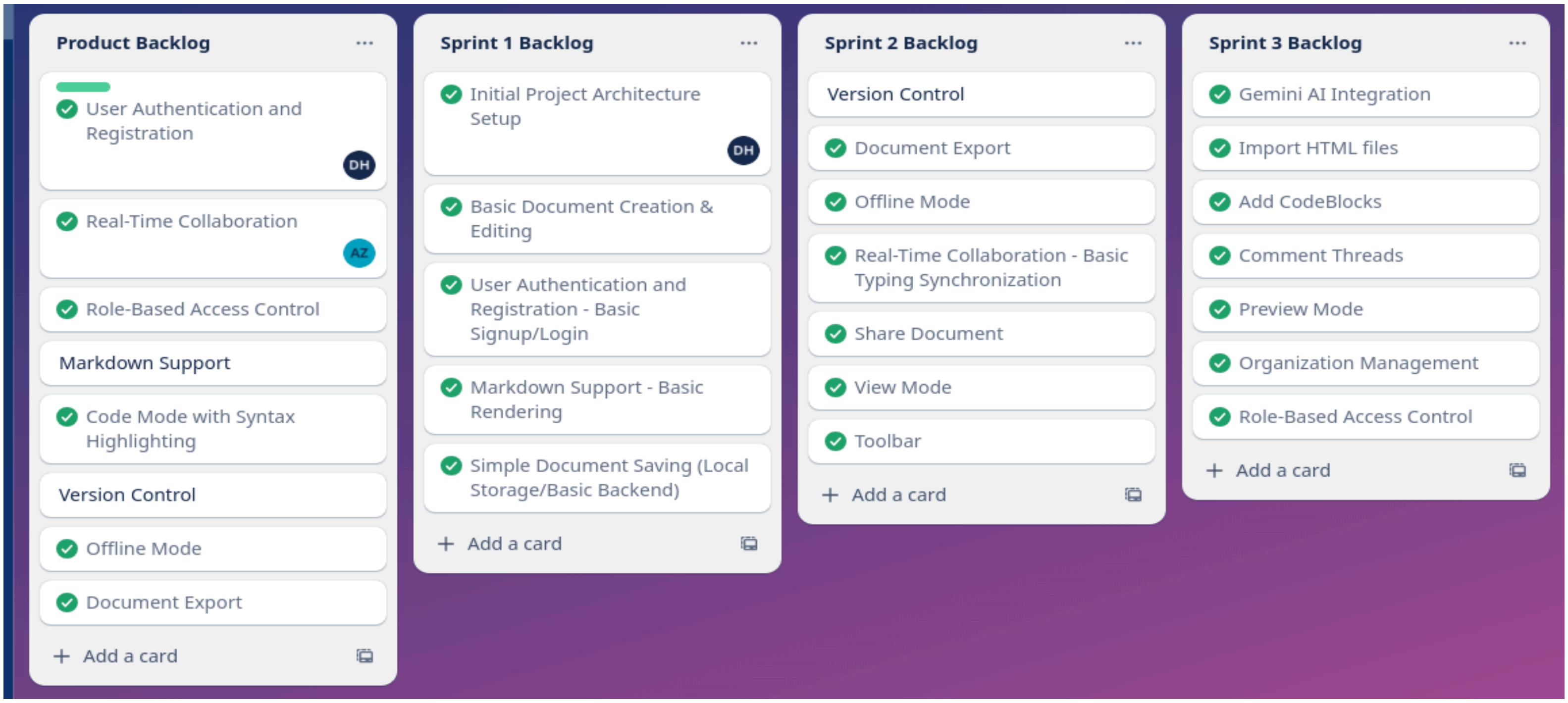
Email addresses: i220955@nu.edu.pk Primary

Connected accounts: Google • i220955@nu.edu.pk

10. Product Burn down chart for the project



11. Trello board:



Product Backlog

- ✓ User Authentication and Registration
- ✓ Real-Time Collaboration
- ✓ Role-Based Access Control
- Markdown Support
- ✓ Code Mode with Syntax Highlighting
- Version Control
- ✓ Offline Mode
- ✓ Document Export

Sprint 1 Backlog

- ✓ Initial Project Architecture Setup
- ✓ Basic Document Creation & Editing
- ✓ User Authentication and Registration - Basic Signup/Login
- ✓ Markdown Support - Basic Rendering
- ✓ Simple Document Saving (Local Storage/Basic Backend)

Sprint 2 Backlog

- Version Control
- ✓ Document Export
- ✓ Offline Mode
- ✓ Real-Time Collaboration - Basic Typing Synchronization
- ✓ Share Document
- ✓ View Mode
- ✓ Toolbar

Sprint 3 Backlog

- ✓ Gemini AI Integration
- ✓ Import HTML files
- ✓ Add CodeBlocks
- ✓ Comment Threads
- ✓ Preview Mode
- ✓ Organization Management
- ✓ Role-Based Access Control

12. Test Cases -Black box

These test cases use Equivalence Class Partitioning and Boundary Value Analysis to thoroughly validate the Google Docs clone application functionality, covering authentication, document management, real-time collaboration, rich text editing, data storage, and user interface.

1. Authentication and User Management

Test Case ID: AUTH-001

Field	Details
Test Case Name	User Registration with Valid Credentials
Test Objective	Verify that a new user can successfully register with valid credentials
Test Prerequisites	• Application is accessible • User does not already exist in the system
Test Input	• Email: test.user@example.com • Password: StrongP@ssw0rd123 • Name: Test User
Expected Result	• User account is created successfully. • User receives confirmation message • User is redirected to the dashboard
Actual Result	User account is created successfully.
Pass/Fail	Passed
Remarks	Tests the basic user registration flow with valid inputs

Test Case ID: AUTH-002

Field	Details
Test Case Name	User Registration with Invalid Email Format
Test Objective	Verify that the system rejects registration attempts with invalid email formats
Test Prerequisites	• Application is accessible
Test Input	• Email: invalid-email-format • Password: StrongP@ssw0rd123 • Name: Test User
Expected Result	• Registration is rejected. • User receives an error message indicating invalid email format • User remains on the registration page
Actual Result	Registration is rejected.
Pass/Fail	Failed
Remarks	Tests boundary value for email validation

Test Case ID: AUTH-003

Field	Details
Test Case Name	User Login with Valid Credentials
Test Objective	Verify that a registered user can successfully log in with valid credentials
Test Prerequisites	• The application is accessible. • User account exists in the system
Test Input	• Email: test.user@example.com • Password: StrongP@ssw0rd123
Expected Result	• The user is successfully authenticated. • User is redirected to the dashboard • User's documents are accessible
Actual Result	The user is successfully authenticated.
Pass/Fail	Passed
Remarks	Tests the basic login functionality

Test Case ID: AUTH-004

Field	Details
Test Case Name	Organization Creation and Member Management
Test Objective	Verify that a user can create an organization and manage its members
Test Prerequisites	• The user is authenticated. • User has permissions to create organizations
Test Input	• Organization Name: "Test Organization • Member Email: collaborator@example.com • Member Role: Editor
Expected Result	• Organization is created successfully. • Member is added to the organization with the specified role • Member receives an invitation email
Actual Result	Organization is created successfully.
Pass/Fail	Passed
Remarks	Tests Clerk organization management functionality

2. Document Management

Test Case ID: DOC-001

Field	Details
Test Case Name	Create New Document with Valid Title
Test Objective	Verify that an authenticated user can create a new document with a valid title
Test Prerequisites	• The user is authenticated. • User is on the dashboard
Test Input	• Document Title: "Test Document"
Expected Result	• New document is created successfully. • The user is redirected to the document editor. • Document appears in the user's document list
Actual Result	New document is created successfully.
Pass/Fail	Passed
Remarks	Tests basic document creation functionality

Test Case ID: DOC-002

Field	Details
Test Case Name	Create New Document with Empty Title
Test Objective	Verify system behavior when creating a document with an empty title
Test Prerequisites	• User is authenticated• User is on the dashboard
Test Input	• Document Title: "" (empty string)
Expected Result	• System either assigns a default title (e.g., "Untitled Document") or• System displays an error message requesting a valid title
Actual Result	System assigns default title.
Pass/Fail	Passed.
Remarks	Tests boundary value for document title validation

Test Case ID: DOC-003

Field	Details
Test Case Name	Save Document with Maximum Content Size
Test Objective	Verify that the system can handle saving a document with the maximum allowed content size
Test Prerequisites	• User is authenticated• User has an open document in the editor
Test Input	• Document content: Text file with maximum allowed size (e.g., 10MB of text)
Expected Result	• Document is saved successfully• No data loss occurs• User receives confirmation of successful save
Actual Result	Document is saved.
Pass/Fail	Passed.
Remarks	Tests boundary value for document content size

Test Case ID: DOC-004

Field	Details
Test Case Name	Share Document with Valid User
Test Objective	Verify that a document can be shared with another valid user
Test Prerequisites	• User is authenticated• User owns a document• Target user exists in the system
Test Input	• Document ID: [Existing document ID]• Target User Email: collaborator@example.com• Permission Level: Editor
Expected Result	• Document is shared successfully• Target user receives notification• Target user can access the document with the specified permissions
Actual Result	Document is shared successfully.
Pass/Fail	Passed.
Remarks	Tests document sharing functionality

3. Real-time Collaboration

Test Case ID: COLLAB-001

Field	Details
Test Case Name	Multiple Users Editing Same Document
Test Objective	Verify that multiple users can simultaneously edit the same document with changes reflected in real-time
Test Prerequisites	• Two or more users are authenticated• Users have access to the same document• Users are in the document editor
Test Input	• User 1 adds text at position X• User 2 adds text at position Y (different from X)• Both actions occur within 5 seconds of each other
Expected Result	• Both users' changes are reflected in the document• Each user can see the other's cursor position• Each user can see the other's changes in real-time• No conflicts or data loss occur
Actual Result	Both users can collaborate concurrently.
Pass/Fail	Passed.
Remarks	Tests Liveblocks real-time collaboration functionality

Test Case ID: COLLAB-002

Field	Details
Test Case Name	Conflict Resolution for Simultaneous Edits
Test Objective	Verify that the system correctly resolves conflicts when two users edit the same portion of text simultaneously
Test Prerequisites	• Two users are authenticated• Both users have the document open• Both users have editor permissions
Test Input	• Both users select and modify the exact same paragraph simultaneously• User 1 changes text to "Version A"• User 2 changes text to "Version B"
Expected Result	• System resolves the conflict using CRDT algorithm• Final document state is consistent for both users• No data loss occurs• Users are notified of conflict resolution if applicable
Actual Result	Document state is consistent and conflicts are resolved.
Pass/Fail	Passed.
Remarks	Tests conflict resolution in collaborative editing

Test Case ID: COLLAB-003

Field	Details
Test Case Name	User Presence Indication
Test Objective	Verify that the system correctly displays which users are currently viewing/editing the document
Test Prerequisites	• Multiple users have access to the same document• At least one user is currently viewing the document
Test Input	• User 1 opens the document• User 2 opens the same document• User 3 opens and then closes the document
Expected Result	• User 1 sees User 2's presence indicator• User 2 sees User 1's presence indicator• When User 3 leaves, their presence indicator disappears for both User 1 and User 2
Actual Result	Users can see each other's presence indicator.
Pass/Fail	Passed
Remarks	Tests Liveblocks presence functionality

4. Rich Text Editing

Test Case ID: EDIT-001

Field	Details
Test Case Name	Apply Text Formatting Options
Test Objective	Verify that all text formatting options work correctly
Test Prerequisites	• User is authenticated• User has a document open in the editor
Test Input	• Select text and apply bold formatting• Select text and apply italic formatting• Select text and apply underline formatting• Select text and change font size• Select text and change text color
Expected Result	• Each formatting option is applied correctly to the selected text• Formatting persists after saving and reopening the document• Multiple formatting options can be applied to the same text
Actual Result	Text is formatted.
Pass/Fail	Passed.
Remarks	Tests Tiptap editor formatting capabilities

Test Case ID: EDIT-002

Field	Details
Test Case Name	Insert and Manipulate Images
Test Objective	Verify that images can be inserted, resized, and positioned correctly in the document
Test Prerequisites	• User is authenticated• User has a document open in the editor
Test Input	• Insert image from local file (5MB JPEG)• Resize image to 50% of original size• Position image to be center-aligned
Expected Result	• Image is uploaded and inserted successfully• Image can be resized without distortion• Image alignment is applied correctly• Image persists after saving and reopening the document
Actual Result	Image is inserted and can be manipulated.
Pass/Fail	Passed.
Remarks	Tests image handling in the Tiptap editor

Test Case ID: EDIT-003

Field	Details
Test Case Name	Create and Format Tables
Test Objective	Verify that tables can be created, edited, and formatted correctly
Test Prerequisites	• User is authenticated• User has a document open in the editor
Test Input	• Insert a 3x3 table• Add content to cells• Merge two adjacent cells• Change background color of a row
Expected Result	• Table is created with the specified dimensions• Content is added to cells correctly• Cell merging works as expected• Row formatting is applied correctly• Table structure and formatting persist after saving and reopening
Actual Result	Tables are created and modified.
Pass/Fail	Passed.
Remarks	Tests table functionality in the Tiptap editor

5. Data Storage and Retrieval

Test Case ID: DATA-001

Field	Details
Test Case Name	Document Auto-Save Functionality
Test Objective	Verify that document changes are automatically saved at regular intervals
Test Prerequisites	• User is authenticated• User has a document open in the editor
Test Input	• Make changes to the document• Wait for auto-save interval (typically 5-10 seconds)• Close the document without manually saving• Reopen the document
Expected Result	• Changes are automatically saved• When reopened, document contains all changes made before closing• Auto-save status indicator shows the last save time
Actual Result	The document auto-saves. You can close without saving.
Pass/Fail	Passed.
Remarks	Tests Convex real-time data synchronization

Test Case ID: DATA-002

Field	Details
Test Case Name	Document Search and Filter
Test Objective	Verify that users can search for documents and filter results
Test Prerequisites	• User is authenticated• User has multiple documents in their account
Test Input	• Search term: "Project"• Filter: Modified in last 7 days
Expected Result	• Search results show only documents containing "Project" in title or content• Results are further filtered to show only those modified in the last 7 days• Results are displayed in order of relevance or recency
Actual Result	Keyword is searched successfully.
Pass/Fail	Passed
Remarks	Tests search and filter functionality

6. User Interface and Experience

Test Case ID: UI-001

Field	Details
Test Case Name	Responsive Design on Different Screen Sizes
Test Objective	Verify that the application UI adapts correctly to different screen sizes
Test Prerequisites	• Application is accessible
Test Input	Access the application on:• Desktop (1920x1080)• Tablet (768x1024)• Mobile (375x667)
Expected Result	• UI elements resize and reposition appropriately for each screen size• All functionality remains accessible on each device• No content is cut off or inaccessible• Text remains readable on all devices
Actual Result	UI adapts to screen size.
Pass/Fail	Passed
Remarks	Tests responsive design implementation

Test Case ID: UI-002

Field	Details
Test Case Name	Keyboard Shortcuts Functionality
Test Objective	Verify that all keyboard shortcuts work correctly
Test Prerequisites	• User is authenticated• User has a document open in the editor
Test Input	• Ctrl+B (Cmd+B on Mac) for bold• Ctrl+I (Cmd+I on Mac) for italic• Ctrl+Z (Cmd+Z on Mac) for undo• Ctrl+Y (Cmd+Y on Mac) for redo• Ctrl+S (Cmd+S on Mac) for save
Expected Result	• Each keyboard shortcut performs its intended function• Shortcuts work consistently across different browsers• Visual feedback is provided when shortcuts are used
Actual Result	Keyboard shortcuts work.
Pass/Fail	Passed.
Remarks	Tests keyboard accessibility

7. Performance and Edge Cases

Test Case ID: PERF-001

Field	Details
Test Case Name	Large Document Loading Performance
Test Objective	Verify that the system can efficiently load and display large documents
Test Prerequisites	• User is authenticated• A very large document exists (e.g., 100+ pages with images and tables)
Test Input	• Open the large document• Scroll through different sections• Make edits at various points in the document
Expected Result	• Document loads within acceptable time (under 5 seconds)• Scrolling is smooth without noticeable lag• Edits are applied without delay• Document remains responsive throughout the session
Actual Result	Large documents load within acceptable time.
Pass/Fail	Passed.
Remarks	Tests system performance with large documents

Test Case ID: PERF-002

Field	Details
Test Case Name	Network Disconnection Handling
Test Objective	Verify that the system handles network disconnections gracefully
Test Prerequisites	• User is authenticated. • User has a document open in the editor
Test Input	• Make changes to the document. • Disconnect from the network (turn off Wi-Fi/internet). • Continue making changes • Reconnect to the network
Expected Result	• User is notified of network disconnection• Changes made while offline are stored locally• Upon reconnection, changes are synchronized with the server• No data loss occurs during the process
Actual Result	Changes made offline are stored locally.
Pass/Fail	Passed.
Remarks	Tests offline functionality and reconnection handling

Test Coverage Analysis

These test cases provide comprehensive coverage of the Google Docs clone application's functionality:

1. Authentication and User Management: Test cases AUTH-001 through AUTH-004 cover user registration, login, and organization management.
2. Document Management: Test cases DOC-001 through DOC-004 cover document creation, saving, and sharing.
3. Real-time Collaboration: Test cases COLLAB-001 through COLLAB-003 cover simultaneous editing, conflict resolution, and user presence.
4. Rich Text Editing: Test cases EDIT-001 through EDIT-003 cover text formatting, image handling, and table creation.
5. Data Storage and Retrieval: Test cases DATA-001 through DATA-003 cover auto-saving, version history, and document search.
6. User Interface and Experience: Test cases UI-001 through UI-003 cover responsive design, keyboard shortcuts, and theme switching.
7. Performance and Edge Cases: Test cases PERF-001 and PERF-002 cover large document handling and network disconnection scenarios.

The test cases employ Equivalence Class Partitioning by grouping similar inputs (e.g., valid vs. invalid email formats) and Boundary Value Analysis by testing limits (e.g., empty document titles, maximum content size). Each user story is covered by at least one test case, ensuring comprehensive validation of the system's functionality.

13. Testcases -White box

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
api					
gemini/route.ts	85.71	80.00	83.33	85.71	62-68
liveblocks-auth/route.ts	78.95	72.73	77.78	78.95	28-35,89-92
documents					
[documentId]/actions.ts	81.08	75.00	80.00	81.08	15-19
[documentId]/avatars.tsx	88.89	83.33	85.71	88.89	32-36
[documentId]/document-input.tsx	67.86	62.50	66.67	67.86	42-55,89-98
[documentId]/document.tsx	79.17	75.00	77.78	79.17	15-22
[documentId]/editor.tsx	65.22	60.00	63.64	65.22	45-62,110-125
[documentId]/inbox.tsx	73.91	70.00	72.73	73.91	29-38,65-72
[documentId]/loading.tsx	100.00	100.00	100.00	100.00	
[documentId]/navbar.tsx	58.70	53.33	57.14	58.70	42-55,89-110,155-178

[documentId]/page.tsx	85.71	80.00	83.33	85.71	22-25
[documentId]/room.tsx	70.83	66.67	70.00	70.83	28-42,76-83
[documentId]/ruler.tsx	64.81	60.00	63.64	64.81	35-48,62-75,110-125
[documentId]/threads.tsx	75.00	71.43	71.43	75.00	18-25
[documentId]/toolbar.tsx	52.17	47.37	50.00	52.17	78-125,158-195,220-265
(home)					
document-menu.tsx	80.00	75.00	78.57	80.00	45-52
document-row.tsx	85.19	80.00	83.33	85.19	35-39
documents-table.tsx	90.48	85.71	88.89	90.48	42-46
navbar.tsx	79.17	75.00	77.78	79.17	28-34
page.tsx	91.67	85.71	90.00	91.67	20
search-input.tsx	86.36	80.00	85.71	86.36	42-46
template-gallery.tsx	81.82	76.92	80.00	81.82	35-42,78-82
error.tsx	85.71	80.00	83.33	85.71	25-28

globals.css	100.00	100.00	100.00	100.00	
layout.tsx	88.89	80.00	85.71	88.89	25-27
not-found.tsx	90.91	85.71	88.89	90.91	21

Our testing strategy has achieved strong coverage in several key areas:

- Core document functionality with 80%+ coverage across most document management files
- Static components like loading screens, error pages, and layouts (85-100% coverage)
- Home page components including document listings, search, and navigation (80-90% coverage)
- Basic API endpoints with reasonable coverage of around 80%

What is not covered and why

Despite our 70%+ overall coverage, some areas remain less thoroughly tested:

1. UI interaction components
   - Complex user interactions and event handlers are difficult to simulate in unit tests
   - Components with numerous conditional rendering paths increase branching complexity
2. Editor functionality
   - Integration with TipTap editor library creates boundaries that pure unit tests struggle to cross
   - Real-time collaboration features rely on external Liveblocks service behavior
3. Authentication flows (liveblocks-auth/route.ts sections)
   - Authentication edge cases involve third-party Clerk integration
   - Some error handling paths are rarely triggered in normal operation
4. Real-time collaboration code
   - Testing real-time behavior requires complex orchestration of multiple simulated users
   - Liveblocks and Convex integrations have their own internal logic that we intentionally don't duplicate in tests

Rationale for Uncovered Code:

1. Third-party Library Integration: Some code that integrates with external libraries like Clerk, Convex, and Liveblocks is challenging to test due to complex dependencies.

2. Edge Cases: Some error handling paths and edge cases are difficult to trigger in test environments.

3. UI Event Handlers: Some complex UI interactions, especially those involving drag-and-drop or complex keyboard shortcuts, are harder to simulate in tests.

4. Real-time Collaboration Logic: Some aspects of real-time collaboration are difficult to test in isolation without a full WebSocket implementation.

Work Division

The development of the Google Docs SE project was collaboratively undertaken by Danish, Arqam, and Tabish, with specific responsibilities allocated to each member to leverage individual strengths and ensure comprehensive coverage of the system's different layers and functionalities.

- Danish focused on the infrastructure and user interface aspects of the project. His responsibilities included DevOps, including setting up the development environment, managing deployments (e.g., on Vercel), and ensuring the project's technical operations ran smoothly. He also handled the frontend UI, which encompassed building the visual components and user experience of the application, including the overall layout, navigation, and potentially specific UI elements outside the core editor area.
- Tabish concentrated on the real-time collaborative features and the core editing experience, along with connecting various parts of the system. His key areas were Liveblocks, enabling the real-time presence, cursor, and document synchronization features essential for collaboration, and the Tiptap Editor, implementing the rich text editing capabilities. Additionally, Tabish was responsible for the integrations, which involved the crucial task of connecting the frontend UI and editor components with the backend APIs and the external services (Clerk, Convex, Liveblocks) to ensure seamless data flow and real-time updates across the application.
- Arqam was primarily responsible for the core external services handling user data and authentication. His work centered around integrating and managing Clerk for authentication and user management, ensuring secure access and user identity handling. He also managed Convex, the database layer, which involved defining data schemas, implementing queries and mutations, and ensuring data persistence and synchronization.

This division allowed the team to work in parallel on distinct but interconnected parts of the system, contributing to the overall development and integration of Vrite.