SELECTED TOPICS IN SOFTWARE ENGINEERING ASSIGNMENT

HARAMAYA UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF SOFTWARE ENGINEERING

Group Members

1.	**Amanuel Daniel**	**0365/13**
2.	**Astatike Amanuel**	**0485/13**
3.	**Bethlehem Zeray**	**0628/13**
4.	**Hossam Mohammed**	**1480/13**
5.	**Molla Kibret**	**2055/13**

Date: 5/23/2025
Instructor: Gizachew B.

What is Code Collab?

Code Collab is a **unified, real-time collaborative coding platform** designed to streamline and modernize how software development teams work together. It serves as an all-in-one solution that eliminates the need to juggle multiple tools for code editing, communication, task management, version control, and secure code execution.

The platform supports **developers, project managers, and organizations** by providing a centralized environment where collaborative coding and project management happen seamlessly in real-time. Whether working in a remote, hybrid, or in-person setup, Code Collab ensures that development teams remain synchronized, efficient, and productive.

Key Benefits:

- Eliminates tool-switching by combining editing, communication, and management tools.
- Supports both independent developers and organizations with tailored feature sets.
- Enhances workflow visibility and reduces miscommunication.
- Provides a secure, cloud-based, scalable environment for teams of all sizes.

Modules of Code Collab & Their Usage

The platform is built around several integrated modules, each designed to address a core need of collaborative software development:

1. Real-Time Code Collaboration

- **Purpose:** Enables multiple developers to edit code together with **instant live updates**.
- **Features:** Shared code workspace, synchronized views, user cursor tracking, inline commenting, and live change history.
- **Impact:** Promotes efficient pair programming, code reviews, and debugging sessions.

2. Integrated Communication Tools

- **Purpose:** Keeps teams connected during coding sessions.
- **Features:** In-app **text chat**, **video calls**, and **audio calls**.
- **Impact:** Reduces friction of switching between separate apps for communication, fostering instant feedback and faster decision-making.

3. Task and Project Management

- **Purpose:** Helps manage development workflows.
- **Features:** Task creation, assignment, tracking tools like **Kanban Boards** and **Gantt Charts**.
- **Impact:** Keeps teams aligned on priorities, deadlines, and responsibilities.

4. Version Control Integration

- **Purpose:** Manages code history and collaboration at scale.
- **Features:** Git integration for repository management, branch creation, commits, and merge conflict resolution.
- **Impact:** Supports a disciplined, version-controlled development process.

5. File Management

- **Purpose:** Organizes project files in a structured and secure manner.
- **Features:** File upload, versioning, folder management, permission settings, and storage limits.
- **Impact:** Keeps project assets organized and accessible to authorized team members.

6. Docker-Based Execution Environment

- **Purpose:** Ensures secure, consistent code execution and testing.
- **Features:** Runs code inside **isolated Docker containers**, supporting multiple tech stacks.
- **Impact:** Prevents environmental discrepancies, making development and testing environments predictable and reliable.

7. User Authentication and Security

- **Purpose:** Protects platform data and manages access.
- **Features:** Secure login, **role-based access control (RBAC)**, encryption, and activity logging.
- **Impact:** Safeguards sensitive project information and enforces structured access controls.

Introduction

As part of our senior project titled **Code Collab: A Unified Collaborative Platform for Real-Time Software Development**, we were tasked with selecting **one key module** and developing a detailed **Product Backlog** and **Sprint Backlog** for it.

This activity aligns with **Agile software development practices**, where requirements are expressed from the user's perspective (as user stories) and prioritized for development through iterative sprints.

Why Choose Real-Time Code Collaboration?

As part of our senior project, **Code Collab: A Unified Collaborative Platform for Real-Time Software Development**, we had the opportunity to select one core module to center our Agile-based backlog preparation and iterative development process around. After careful consideration, we selected the **Real-Time Code Collaboration** module as the focus for several strategic and technical reasons. This decision aligns not only with the platform's overall vision but also with modern software development trends and the pressing needs of remote and distributed development teams.

Importance to the Platform

The **Real-Time Code Collaboration module** is the **heart and defining feature of Code Collab**. It is the functionality upon which the platform's value proposition rests. The entire system is designed to enable and enhance collaborative software development by providing a shared, synchronized environment where users can work together seamlessly. Without this module, other complementary modules such as project management, task tracking, file management, and communication would lose much of their contextual relevance.

Key Functionalities and Features

This module introduces and integrates the following core capabilities that make it indispensable:

- **Write and Edit Code Together in Real-Time**
    - Multiple developers can work on the same code files simultaneously.
    - Code changes are reflected instantly in the shared environment for all participants.
    - Live updates reduce delays, miscommunication, and redundant work.
- **Communicate Instantly through Chat and Video Calls**
    - Integrated real-time chat allows team members to discuss ideas, ask questions, and give feedback without leaving the coding workspace.
    - Embedded video call functionality makes it easy to hold stand-ups, code reviews, and pair programming sessions within the same application.
- **See Who is Editing What at Any Moment**
    - User cursors are visually distinguishable with user names, making it easy to identify who is working where in the code.

    - This improves awareness, prevents conflicts, and fosters better team coordination.
- **Seamless Synchronization in a Shared Coding Workspace**
    - Code editors remain fully synchronized across all participants' screens.
    - Changes, comments, and file switches occur in real-time without requiring manual refreshes or external commits.

How It Supports the Vision of Code Collab

The vision behind Code Collab is to **eliminate the friction of working remotely or in distributed teams** by combining all essential software development tools into a single, unified platform. The Real-Time Code Collaboration module directly supports this vision by:

- **Integrating live code editing and communication tools** into one seamless experience.
- **Minimizing tool-switching** between external editors, messaging apps, video conferencing software, and version control platforms.
- **Reducing misunderstandings and coordination delays** by allowing developers to see each other's work and communicate contextually in real time.
- **Creating a truly collaborative, cloud-based development environment** that is accessible to individuals and teams regardless of their physical location.

Strategic Justification for Backlog Focus

In Agile software development, it's crucial to prioritize modules that deliver the highest value and address the most significant user needs first. By focusing our product backlog preparation and sprint planning on the Real-Time Code Collaboration module, we aim to:

- **Deliver the most essential and high-impact functionality early** in the development cycle.
- **Establish a solid foundation upon which other platform modules can be built and integrated**.
- **Test and validate the collaborative coding environment with users quickly**, gathering feedback that can shape the refinement of the overall platform.
- **Ensure that the core value proposition of Code Collab — seamless, real-time software development collaboration — is effectively realized**.

1. Product Backlog

Product Vision

The vision of this module is to **design, develop, and deliver a powerful real-time collaborative coding environment** that empowers multiple users, including developers, team leads, QA engineers, and project managers, to work on the same codebase **simultaneously in a cloud-based environment**.

This module seamlessly integrates essential collaboration features such as:

- **Live code synchronization**
- **User cursor and selection tracking**
- **Inline code commenting**
- **Integrated real-time communication tools (chat, audio, video calls)**

By **removing the inefficiencies caused by tool-switching and asynchronous workflows**, this module enables fully interactive, real-time software development and review experiences for distributed teams working from different locations and time zones.

Ultimately, it delivers a **unified, intuitive, and scalable online code editor** that improves development velocity, team coordination, and code quality.

Product Goal

The primary goal of the **Real-Time Code Collaboration module** is to:

- Provide a **cloud-based, secure, and scalable online code editor**
- Support **live collaboration with instant code synchronization**
- Integrate **contextual, in-editor communication tools** such as chat and video calls
- Offer **visibility into active collaborative activity** (cursors, user edits, inline comments, and session logs)
- Enable **session management** (saving, reopening, and tracking collaborative coding sessions)

This module aims to:

- Increase developer productivity and efficiency
- Reduce communication breakdowns and misunderstandings
- Eliminate the need for multiple disjointed tools
- Serve as the **core and most valuable feature of the Code Collab platform**

By achieving this, Code Collab can position itself as an **all-in-one, frictionless collaboration environment for modern, remote-first software teams.**

Product Backlog Items (User Stories)

ID	User Story	Priority	Acceptance Criteria
US1	As a Developer, I want to collaborate on code in real time so that all changes are instantly reflected to my teammates.	High	Code edits are broadcast live and instantly visible to all users without lag or refresh.
US2	As a User, I want to see other users' cursors and selections, so I can know who is editing where in the code.	High	Unique cursors with usernames and color coding appear in the editor for each user.
US3	As a Team, we want to chat within the editor so we can communicate without switching apps.	High	An embedded chat window allows real-time message exchange within the coding environment.
US4	As a User, I want to start video calls inside the editor so I can discuss issues face-to-face while coding.	Medium	Video call functionality can be initiated from the editor and supports multiple participants.
US5	As a Developer, I want to add inline comments to specific lines of code so I can suggest improvements or raise concerns.	Medium	Comments are attached to individual code lines and visible to all participants.
US6	As a User, I want syntax highlighting for multiple languages so I can code effectively without errors.	High	Editor supports syntax highlighting for at least five languages, including JS, Python, Java, C++, and PHP.
US7	As a Team Lead, I want to track edit history so I can see who made what changes and when.	Medium	Change log displays user actions with timestamps in a session-based history.
US8	As a Developer, I want to save and reopen collaborative sessions so our work isn't lost if we disconnect.	Medium	Sessions can be automatically or manually saved and reopened later.

2. Sprint Backlog — Sprint 1

Sprint Goal

Objective:
To design, develop, and deliver the **minimum viable version of the Real-Time Code Collaboration module**, establishing the core collaborative functionalities essential for enabling multiple users to interact within a shared coding environment.

This first sprint will focus on laying down the **technical foundation** of the editor and the collaborative infrastructure. The goal is to ensure that developers can:

- **Edit the same codebase simultaneously** with **live code synchronization** across all participants' editors.
- **See each other's cursor positions and text selections** in real time, improving collaboration awareness and reducing conflicts.
- **Communicate within the platform via an embedded chat window**, eliminating the need for external communication tools during coding sessions.

By the end of this sprint, the platform should deliver a **functional, working prototype** where:

- The **real-time collaborative code editor is fully integrated** and operational.
- **Bi-directional communication using WebSockets** is implemented for both code synchronization and chat messaging.
- **Multi-user cursor tracking** is active, accurately displaying each participant's cursor location and selection in the editor.

This sprint aims to validate the feasibility of real-time collaboration and provide the groundwork for advanced collaborative features like inline comments, video calls, and session history in subsequent sprints.

Sprint Tasks for Sprint 1 (Organized by User Story)

US1 — Real-Time Code Synchronization

Task ID	Task Description	Assignee	Est. Hours	Est. Days
T1.1	Set up WebSocket server infrastructure for handling real-time connections.	Backend Dev	4	0.67
T1.2	Integrate WebSocket connections into the client-side code editor.	Frontend Dev	3	0.5
T1.3	Develop broadcasting logic	Backend Dev	5	0.83

	for code changes to all connected users.			
T1.4	Implement event listeners to detect and transmit code changes.	Frontend De	3	0.5
T1.5	Perform integration testing with multiple clients for sync validation.	QA / Dev	3	0.5
Subtotal US1			18 hrs	3.0 days

US2 — Multi-User Cursor Tracking

Task ID	Task Description	Assignee	Est. Hours	Est. Days
T2.1	Define cursor metadata structure (user ID, color, position, selection).	Backend Dev	2	0.34
T2.2	Implement real-time cursor position broadcasting via WebSockets.	Backend Dev	3	0.5
T2.3	Integrate cursor rendering on frontend with unique colors and labels.	Frontend Dev	4	0.67
T2.4	Test multi-user cursor visibility within collaborative sessions.	QA / Dev	3	0.5
Subtotal US2			12 hrs	2.0 days

S3 — Embedded Chat Window (Minimum Chat Integration for Prototype)

Task ID	Task Description	Assignee	Est. Hours	Est. Days

T3.1	Design and implement simple chat UI in editor layout.	Frontend Dev	3	0.5
T3.2	Establish WebSocket channels for chat message exchange.	Backend Dev	3	0.5
T3.3	Implement frontend event listeners for real-time messaging.	Frontend Dev	3	0.5
T3.4	Conduct functional testing of chat with multiple clients.	QA / Dev	2	0.34
Subtotal US3			11 hrs	1.84 days

US6 — Syntax Highlighting for Multiple Languages

Task ID	Task Description	Assignee	Est. Hours	Est. Days
T6.1	Integrate code editor library (Monaco/CodeMirror) into the project.	Frontend Dev	4	0.67
T6.1	Configure syntax highlighting for 5 programming languages (JS, Python, Java, C++, PHP).	Frontend Dev	5	0.83
T6.1	Test syntax highlighting accuracy across supported languages.	QA / Dev	3	0.5
Subtotal US6			12 hrs	2.0 days

Sprint 1 Effort Summary

Backlog Item	Total Hours	Total Days
US1	18	3.0
US2	12	2.0
US3	11	1.84
US6	12	2.0
Total	53	8.84 days

Sprint 1 Expected Deliverables:

- A working **real-time collaborative code editor prototype**.
- **Multi-user cursor tracking** with unique user indicators.
- **Basic embedded chat feature** integrated into the editor.
- **Syntax highlighting** for at least 5 programming languages.

Final Sprint Summary

Section	Details
Module Chosen	Real-Time Code Collaboration
Product Goal	Seamless team collaboration with code editing, real-time synchronization, and communication tools integrated into a single platform.
Product Backlog	8 user stories prioritizing the collaborative editor, live sync, cursor tracking, chat, video calls, inline comments, syntax highlighting, edit history, and session management.
Sprint 1 Goal	Implement the core real-time collaborative functionalities — including code editor integration, live code synchronization via WebSocket, multi-user cursor tracking, syntax highlighting, and embedded chat. Deliver a functional prototype.
Sprint 1 Tasks	14 technical tasks mapped to 4 user stories (US1, US2, US3, US6), covering WebSocket setup, synchronization, cursor tracking, chat integration, syntax highlighting configuration, and multi-user testing.

Sprint 1 Effort	53 hours / 8.84 days for 1 sprint team