CollabNote: Collaborative Course Note-Sharing Platform

Xiaoyu Luo 50087474, Chen Zhuoxin 50087472, Juntong Luo 50087459,

Zihang Liang 50087475, Chenwei Lin 50087489, Qian Yumo 50087464

CollabNote is a vertically integrated academic collaboration platform for higher education. It integrates real-time collaborative editing, intelligent version control, and gamified incentive mechanisms to address academic adaptation gaps and inefficient workflows in existing tools. The platform aims to facilitate cross-generational knowledge transfer and sustainable digital asset development.

Problem：

University students are facing the problems of inefficiencies in collaborative learning and course knowledge management due to the following interconnected challenges:

1. **Mismatch Between Generic Tools and Academic Workflows**:
   Traditional collaboration platforms like Google Docs lack academic-specific features which can easily lead to confusions for users between projects. This can cause:
   i. Students are unable to organize notes by course or utilize academic formatting standards like Markdown for equations and code snippets.
   ii. Disjointed version histories that obscure contributions and make iterative improvements difficult to track.

   Users are demanded to get familiar with some foundational technical skills like version control. Which prevents users from focusing on the specific domain of their projects.

2. **Fragmented Knowledge Preservation**:
   Course notes are usually scattered across personal devices or ephemeral shared links, leading to:
   i. Redundant efforts by new student cohorts to reconstruct foundational course materials annually.
   ii. Gradual loss of valuable insights (e.g., exam preparation strategies) that could benefit future learners.

   Traditional methods of notes updating is not suitable for today's students' study pace.

3. **Low Incentives for Sustained Collaboration**:
   Current tools fail to recognize or reward contributions, causing:
   i. Over-reliance on individual initiative for knowledge sharing, with no mechanism to acknowledge consistent contributors.
   ii. Accessibility barriers for non-technical students, as complex version control systems (e.g., Git) remain intimidating for casual users.

   This may substantially erode team morale throughout project execution, diverting focus from core collaborative objectives.

4. **Limited Feedback Channels for Educators**:
   Teaching assistants struggle to identify recurring knowledge gaps across student

groups due to the decentralized nature of study materials, hindering targeted academic support.

Solution：

1. Mismatch Between Generic Tools and Academic Workflows
Problem: Generic tools like Google Docs lack academic-specific features, leading to disorganization and difficulty in tracking contributions.
Solution:

- **Develop or Adopt Academic-Focused Collaboration Platforms:** Use or create tools specifically designed for academic workflows, such as **Overleaf** for LaTeX-based document editing or **Notion** for organizing notes by course.
- **Integrate Version Control for Non-Technical Users:** Implement user-friendly version control systems (e.g., GitHub with a simplified interface) that allow students to track changes without requiring advanced technical skills.
- **Templates for Academic Workflows:** Provide pre-designed templates for course notes, project reports, and collaborative assignments. These templates can include sections for equations, code snippets, and references, reducing confusion and saving time.

---

2. Fragmented Knowledge Preservation
Problem: Course notes and materials are scattered, leading to redundant efforts and loss of valuable insights.
Solution:

- **Centralized Knowledge Repository:** Create a centralized, cloud-based repository (e.g., a university-hosted platform or a shared Notion workspace) where students can upload and organize course materials. This repository should be accessible to current and future cohorts.
- **Structured Note-Taking System:** Encourage students to use a standardized note-taking format (e.g., Markdown or LaTeX) and categorize notes by course, topic, and year. This ensures consistency and makes it easier for future students to build on existing materials.
- **Peer-Reviewed Knowledge Base:** Establish a system where students can contribute to a shared knowledge base, with older students or teaching assistants reviewing and curating the content. This ensures quality and reduces redundancy.

---

3. Low Incentives for Sustained Collaboration

Problem: Lack of recognition for contributions and accessibility barriers for non-technical students.

Solution:

- **Gamification and Recognition:** Introduce a gamified system where students earn points or badges for contributing to shared resources (e.g., uploading notes, answering questions, or improving existing materials). These contributions could be recognized in course evaluations or even count toward participation grades.
- **Peer Accountability:** Assign roles within study groups (e.g., note-taker, editor, reviewer) to ensure equitable participation.

---

4. Limited Feedback Channels for Educators

Problem: Decentralized study materials make it difficult for educators to identify knowledge gaps.

Solution:

- **Analytics-Driven Insights:** Use platforms that provide analytics on student engagement and content usage (e.g., which sections of notes are most viewed or edited). This data can help teaching assistants identify recurring knowledge gaps.
- **Structured Feedback Loops:** Implement regular **Questionnaire** where students can highlight areas of difficulty.

Objectives:

In this project design, we will develop a collaborative learning platform. Our final deliverables aim to achieve the following goals:

1. Complete Platform Development

The primary objective is to build the collaborative platform. We will enhance student efficiency through functional development:

Ⅰ Help students effortlessly distinguish notes across multiple courses: The platform shall implement course categorization management, such as organizing note documents for different courses using tree diagrams.

Ⅱ Create academic-formatted editor: Integrate Markdown formula OCR recognition to cover 90% STEM discipline requirements

Ⅲ Facilitate historical version tracking: Implement a version history backtracking feature.

2. Construct a Course Knowledge Base

Build an uploadable course knowledge base to streamline note management for current students and systematically assist future students in avoiding redundant work:

Ⅰ Build cloud-based shared storage: Support 1,000+ concurrent collaborations with cross-device access latency <200ms

Ⅱ Deploy intelligent tag search system: Enable "course code + concept" combo queries to reduce content localization time from 15min to 3min

Ⅲ Eliminate repetitive work for new students: Establish an archival system for past years' notes, categorized and formatted for easy reference.

3. Enhance Collaborative Incentives

Member engagement significantly impacts collaborative projects. To boost team morale, the platform shall incorporate incentive mechanisms:

Ⅰ Develop contribution metrics: Integrate with university academic systems via API, allowing contribution points to redeem up to 10% course credit

Ⅱ Simplify operations for collaboration: Create role-based templates with predefined roles (e.g., Team Lead/Backend/Testing) and corresponding permissions.

4. Enhance Teaching Feedback Channels

The platform will also engage instructors by developing targeted feedback mechanisms to improve pedagogical support:

Ⅰ Generate learning heatmaps: Automatically tag top 3 high-difficulty chapters through concept access frequency analysis

Ⅱ Enable instructors to proactively identify learning challenges: Develop a survey tool with predefined templates for common questions, such as "Select the chapter you find most difficult."

Benefits

1. Simplified Collaboration

• Clear Course Organization:

Each course has a dedicated notebook space, making it easy to separate programming notes from math notes—like organizing books on different shelves.

• Auto-Formatted Content:

Use built-in buttons in the editor to insert code blocks or equations with proper formatting (e.g., automatic indentation for code, clean equation layouts), eliminating manual adjustments.

2. Seamless Knowledge Transfer

• Access Past Notes Instantly:

Search for materials using course codes (e.g., JC3506) and keywords (e.g., “Lectures WEEK1 Notes") to find high-quality notes uploaded by seniors.

• Avoid Redundant Work:

New students can download verified notes from previous years, such as summaries of common lab mistakes in "Machine Learning," instead of starting from scratch.

3. Transparent Contribution Tracking

• Visualize Contributions:

The group project interface displays each member’s edits (e.g., Jason wrote the

experiment steps, David organized references) with clear metrics like word count.
• Easy Role Assignment:
Group leaders can assign roles in one click (e.g., assign Jame to check formatting and Peter to update content), preventing task overlap.

4. Early Problem Detection
• Highlight Confusing Content:
The system flags sections revised repeatedly (e.g., a code explanation edited 10 times) in red to alert instructors about potential misunderstandings.
• Quick Feedback Collection:
Teachers can launch polls (e.g., "Which topic in Chapter 2 is hardest?") with one click, and results auto-generate into charts—no manual data sorting.

5. Personal Achievement Records
• Archive Learning Milestones:
Upon graduation, students can apply to turn their best notes into "Officially Recommended" materials and generate a contribution report (e.g., "Helped 50 classmates solve database issues").

Time Line

Phase	Description / Subtasks	End Date
Phase 1: Requirements Analysis & Prototype Design	1. Finalize core functional requirements documentation (including JWT-based user permission system). 2. Design prototype interfaces supporting Markdown note preview/editing. 3. Database schema design: • User Table: Fields for JWT tokens. • Note Table: Stores Markdown content and version history. • Interaction Log Table: Records user actions (likes, comments).	April 10, 2025
Phase 2: Core Feature Development	Subtask 1: User System • Implement JWT authentication (registration/login) with role-based permissions (student, TA, admin). Subtask 2: Note Functionality • Integrate CodeMirror rich text editor (Markdown syntax highlighting). • Develop WebSocket-based real-time collaboration and version history tracking. Subtask 3: Basic Interaction Features • Use Redis to cache likes/favorites data and enable comment functionality. Subtask 4: Note Search & Filtering • Implement full-text search via PostgreSQL indexing. • Enable tag filtering with many-to-many relational tables.	May 1, 2025

	Subtask 5: Contribution Scoring Systems • Implement the basic function of contribution scoring.	
Phase 3: Incentives & Additional Features	Subtask 1: Points System • Track user contributions (edits, collaborations) using Redis for real-time calculations. Subtask 2: Reward Redemption • Define rewards (custom fonts, backgrounds, titles). • Design redemption center UI and user personalization settings. Subtask 3: Notification System • Deliver real-time alerts (points changes, comments, note updates) via WebSocket.	May 19, 2025
Phase 4: Security Testing & Optimization	1. Stress-test WebSocket stability under high concurrency. 2. Optimize PostgreSQL query performance (index critical fields). 3. User experience improvements: • Refine interaction workflows. • Optimize loading speeds.	May 26, 2025
Phase 5: Deployment & Beta Testing	1. Deploy servers and complete domain registration. 2. Conduct closed testing with invited users. 3. Collect feedback and prioritize optimizations.	TBD

Action Plan

Objective	Action	Assigned To	End Date	Progress Tracking
1.Requirements & Prototype	Finalize user permissions & editing workflows	Project Manager	03/28/2025	Review via Shared Docs
2.Database Design	Design PostgreSQL tables (User/Note/Interaction)	Backend Developer	04/10/2025	Share schema in Shared Docs
3. User System	Develop JWT authentication + role-based access control Backend Developer	Backend Developer	04/15/2025	Gitee commit records + daily sync
4. Editor & Collaboration	Integrate CodeMirror (Markdown support)	Frontend Developer	04/30/2025	Weekly demo via screen share
	Develop WebSocket real-time sync	Full-stack Developer	05/05/2025	Test with Postman scripts
5. Search & Filter	Implement PostgreSQL full-text search + tag filtering	Backend Developer	05/10/2025	Submit test cases to Gitee Issues
6. Points & Notifications	Redis-based points calculation logic	Full-stack Developer	05/16/2025	Validate via Redis CLI
	WebSocket alerts (comments/points)	Full-stack Developer	05/19/2025	Share real-time message screenshots in WeChat
7. Security & Testing	Simulate 50-user concurrency (using Locust)	Team Rotation	05/23/2025	Upload test reports to WeChat group

8. Deployment & Beta	Docker deployment (Nginx + PostgreSQL)	Full-stack Developer	05/26/2025	Share deployment logs in Shared Docs
	Invite 20 users for closed testing	Project Manager	To be determined	Collect feedback via WeChat

Technical Specifications

Frontend:

The frontend will utilize Vue3 (with TypeScript support) as the core framework. Real-time collaboration is achieved through the Yjs collaboration protocol and native WebSocket communication. The academic editor integrates CodeMirror 6 for Markdown editing and KaTeX for mathematical formula rendering, addressing mixed-content creation needs in academic scenarios.

1. Core Framework
• Vue3 (with TypeScript support)
2. Real-Time Collaboration
• Yjs (collaboration protocol) + WebSocket (communication)
3. Academic Editor
• CodeMirror 6 (core editing) + KaTeX (formula rendering)

Backend:

The backend will utilize Spring Boot (Java) as the core framework. Real-time collaboration is enabled via WebSocket (STOMP) and PostgreSQL storage. Authentication relies on Spring Security + JWT, while Redis drives real-time scoring and anti-abuse mechanisms.

1. Core Framework
• Spring Boot (Java)
2. Real-Time Collaboration
• WebSocket (STOMP protocol)
• PostgreSQL (document snapshots)
3. Authentication & Authorization
• Spring Security + JWT + OAuth2
4. Data & Storage
• PostgreSQL (relational data)
• Redis (caching/rate limiting)
• Elasticsearch (search engine)
5. Contribution Scoring
• Event-Driven Architecture (Spring Events)
• Redis (real-time leaderboards)

Development Focus:
Shared document editing and contribution scoring systems will be prioritized during development, with additional features implemented subject to timelines. Final feature

availability may vary based on product iterations.