Note-taking app – Notely

Software Engineering Project

Project development process

Metropolia University of Applied Sciences
Information and communication technologies
Software engineering
Software project
27.08.25

1. Project overview

This document covers the purpose, chosen methodologies, development process, and final outcomes of the “Notely” , our note-taking application for software engineering students and people who work in the IT- industry.

1.1. Problem Summary

ICT and Computer Science students often find it challenging to take notes in a clear and efficient way. The existing features of general note-taking applications are not well suited for creating programming notes. Code snippets attached to the notes are often poorly formatted, which makes them look messy and reduces writing efficiency.

1.2. Target audience

Primary users of the app are ICT and Computer Science students in universities, and other educational centers. Also, educational staff and lecturers support the students' learning. Also skilled technology users who prefer effective and performance efficient applications, with clear and intuitive user interface design.

1.3. Main Features

- Core note management features for adding, editing and deleting notes, with included text formatting options.
- Organization tools for custom categories of notes.
- Search functionality for filtering by keywords.
- Multi-tab interface for having multiple pages of notes open at the same time.
- Collaborative features for real time sharing with an invitation.
- Code integration for formatting pasted code into notes, with proper syntax highlighting.
- Multimedia support for Images and diagrams.

2. Objectives

Our objectives are for at least 40% usage among the test group within 2 weeks of deployment. And to maintain user satisfaction scores of 70% or higher through the testing phase. We want to deliver a responsive application with low load times under 2 seconds. For online and collaborative features, we want to provide editing with multiple users at the same time.

While the application is developed, we want to keep the application as open source for broader community adoption. We also want to see improved note taking for programming students by 50% compared to traditional methods.

3. Project scope and deliverables

3.1. Included

- Web-based note taking
- Collaborative features
- Code pasting integration
- Image attachment
- User authentication
- Search
- Responsiveness
- Basic documentation

3.2. Deliverables

1. Functional prototype
2. Complete application
3. User documentation
4. Technical documentation
5. Testing reports

4. Timeline

4.1.Phase 1

Planning and analysis will be mostly done in sprint 1 and 2 between 18.08.25 – 31.08.25. Mostly focusing on planning of the project and setup of necessary technology.

- Requirements gathering and analysis.
- Technology stack finalization
- Project setup
- Team role assignments

4.2. Phase 2

Implementing frontend and unit tests will be implemented in sprint 3 and 4 between 01.09.25 – 14.09.25. In this part we will mostly focus on implementation of the application in the frontend and building the application.

- UI and UX implementations for frontend
- Database schema design
- System architecture planning
- API endpoint planning
- Unit tests

4.3. Phase 3

Implementation of backend services and Jenkins in sprint 5 and 6 15.09.25 – 28.9.25. After a well established frontend we will implement the logic of the application.

- Backend API implementation
- Basic authentication
- Jenkins implementation
- Basic connection with frontend

4.4. Phase 4

Implementation of database and extending product functionality, docker implementation and finishing of documentation in sprint 7 and 8 between 29.09.25 – 12.10.25.

- Database implementation
- Docker implementation
- Documentation and repository finalization
- Final fixing and testing
- Analysis and reporting

4.5. Phase 5

Testing implementation and debugging of the application is in sprint 9 and 10 between 13.10.25 – 26.10.25.

- Unit tests
- Integration testing
- Debugging and fixing of issues
- UI and DB localization

4.6. Phase 6

Documentation and preparing the application for deployment is in sprint 11 and 12 between 27.10.25 – 09.11.25.

- Documentation for project, user guides and developer documentation
- Preparing application for deployment
- Cloud Services
- Quality assurance

4.7. Key milestones

1. Project plan submission
2. Design specifications
3. Functional prototype
4. Testing phase completion
5. Final product delivery

5. Resource Allocation

5.1. Team members and roles

The product will be developed by a four member team: Emil Blumenthal, Verner Etola, Emmi Korhonen and Musa Saqid Musa. Team members rotate between different roles, including Scrum Master, developer, UI/UX designer and a tester. The development process will follow the Agile Scrum methodology. The roles shift across sprints and responsibilities are assigned according to each team member’s strengths. The key responsibilities for each role are described below.

- **Scrum Master:** Guides the team and facilitates Scrum process, including sprint planning, daily meetings, sprint reviews and retrospectives.
- **Developer:** Handles backend development.
- **UI/UX Designer & Developer:** Creates user-friendly interface and implements frontend features.

- **Tester:** Manages testing process to ensure quality. Maintains technical documentation.

The Scrum Master role will rotate through the sprints as follows: Sprint 1 - Verner Etola, Sprint 2 - Emmi Korhonen, Sprint 3 - Musa Saqid Musa and Sprint 4 - Emil Blumenthal.

5.2. Software and tools

Various software and tools are used in the development of the product. Discord is used for communication and collaboration among the team. The development environments include Visual Studio Code and IntelliJ IDEA code editors. GitHub is used for version control, allowing multiple developers to work on the project at the same time. The product's user interface is designed using Figma. We will also be utilizing Trello for the team management.

5.3. Technology stack

The product will be developed as a web-based application. The following technology stacks are used to ensure performance and support the product's main features.

- **Frontend:** React + Tailwind CSS
- **Backend:** Node.js + Express
- **Database, storage and authentication:** Supabase
- **Package Manager/Runtime**: Bun
- **Docker**: Compartmentalization

5.4. External resources

The product development will be supported by external resources, such as course materials and lecturer supervision. The reference materials are based on reliable educational and professional sources.

6. Risk management

6.1. High priority risks

1. **Real-time Collaboration Conflict**
   - **Risk:** Simultaneous edits may lead to overwriting or data inconsistency.
   - **Mitigation:** Implement operational transforms (e.g., using **yjs** with Supabase Realtime) and maintain version history for rollback.

2. **Data Security and Privacy**
   - **Risk:** Unauthorized access to student notes or personal data.
   - **Mitigation:** Use Supabase authentication and role-based access control; ensure HTTPS and JWT validation.

3. **System Scalability**
   - **Risk:** Performance degradation when many users collaborate on large notes.
   - **Mitigation:** Optimize database queries, enable Supabase row-level policies, and plan for horizontal scaling if usage grows.

6.2. Medium priority risks

1. **Frontend Performance**
   - **Risk:** Large notes with images and annotations could slow down rendering.
   - **Mitigation:** Use lazy loading, virtualized lists, and efficient state management in React.

2. **Collaboration Latency**
   - **Risk:** Network lag or dropped WebSocket connections disrupt realtime updates.
   - **Mitigation:** Implement reconnection strategies and local buffering until sync is restored.

7. Testing and quality assurance

7.1. Testing strategy

- **Unit Testing:** Validate individual functions, components, and services using Bun’s built-in test runner.

- **Integration Testing:** Test the interaction between React frontend, Node backend, and Supabase.

- **End-to-End Testing:** Simulate real user flows (note creation, editing, collaboration) using automated tools.

- **Continuous Integration (CI):** Jenkins pipeline to run all tests on each commit/PR to prevent regressions.

7.2. Success criteria

- 90% unit test coverage on critical backend and frontend functions.

- Real-time updates propagate to all clients with <300ms latency under normal conditions.

- Zero unresolved **critical bugs** at the end of each sprint.

- Authentication and authorization verified with no data leaks.

7.3. Testing tools and frameworks

- **Bun Test Runner:** For unit and integration tests (fast, built-in with Bun).

- **Jenkins:** For CI/CD automation, running test suites, and generating reports.

- **Cypress (Maybe):** For end-to-end browser-based testing of collaborative features.

8. Documentation and reporting

8.1 Documentation plan

- **Code Documentation:** Inline comments (minimal, code should be self-documenting), JSDoc/TSDoc style annotations for maintainability.

- **API Documentation:** Auto-generated from Express routes (Postman collections).

- **User Documentation:** A lightweight user guide for note-taking, annotation, and collaboration features.

- **Architecture Documentation:** High-level system diagram (React, Node, Supabase, Bun) and explanations for future contributors

8.2 Progress reporting

- **Sprint Reports:** A report generated after each sprint summarizing completed tasks, blockers, and upcoming goals.

- **Jenkins Reports:** Automatically generated test and build reports shared with the team.

- **Daily Check-ins:** Quick team updates on progress, risks, and any support needed.

9. Conclusion

This project plan gives a roadmap for developing a note-taking application designed for programming students. With a timeline, defined roles and thorough risk management, the team is prepared to deliver a valuable educational tool. The combination of modern technology, agile development, and user-centered design approach, ensures a successful project with a well defined scope.