EpiTrello

Project Specifications

Project: Project Management Application

Period: October 20, 2025 - January 28, 2026

Team: Corentin Wolff & Basile Trebus-Hamann

1 Project Overview

Context

Development of a Trello-inspired project management web application as part of a simulated professional work project.

Duration: October 20, 2025 to January 28, 2026 (14 calendar weeks)

Schedule: 3 days per week (Monday, Tuesday, Wednesday) = ~39 work days total

Team

👨‍💻 Corentin Wolff

👨‍💻 Basile Trebus-Hamann

Objective

Create **EpiTrello**, a collaborative task management application based on the Kanban method, allowing users to organize projects through boards containing lists of cards representing tasks.

2 Features

👤 User Management

- ✓ Secure registration and authentication
- ✓ User profile management
- ✓ Logout

📋 Board Management

- ✓ Create, edit, and delete boards
- ✓ View list of boards (personal and shared)
- ✓ Share boards with other users
- ✓ Permission management (owner, member)

📝 List Management

- ✓ Create lists within a board
- ✓ Edit list titles
- ✓ Reorganize via drag-and-drop
- ✓ Delete and archive lists

🎴 Card Management

- ✓ Create and edit cards
- ✓ Detailed descriptions
- ✓ Colored labels and due dates
- ✓ Checklists and comments
- ✓ Move cards between lists
- ✓ Assign members to cards

🔄 Real-Time Collaboration

- ✓ Instant synchronization of changes
- ✓ Notifications for actions
- ✓ Active user presence indicators

✨ Additional Features (Bonus)

- ✓ Card search and filtering
- ✓ Board action history
- ✓ Data export
- ✓ Dark mode

3 Technical Specifications

Architecture

- Client-server architecture with REST API
- Real-time communication via Socket.io
- Frontend/Backend separation

Technologies

Frontend **React**	Styling **Tailwind CSS**	State **Redux**
Drag & Drop **dnd-kit**	Backend **Node.js**	Framework **Express**
Real-time **Socket.io**	Database **MongoDB**	Auth **JWT**

4 Development Tools

🧰 Version Control

- ✓ Git with GitHub
- ✓ Structured commit conventions

📊 Project Management

- ✓ GitHub Projects
- ✓ Agile methodology
- ✓ 2-week sprints

✅ Code Quality

- ✓ ESLint for linting
- ✓ Prettier for formatting
- ✓ Husky for pre-commit hooks

🚀 CI/CD & Deployment

- ✓ GitHub Actions
- ✓ Docker & Docker Compose
- ✓ Automated tests

5 Methodology

Agile Approach

- Iterative development with 2-week sprints
- Sprint planning with defined objectives
- Daily stand-ups for synchronization
- Sprint retrospectives for continuous improvement
- GitHub Projects for backlog management

Test-Driven Development

70%	3	100%
Minimum coverage	Types of tests	Automated tests

- Unit tests for functions and components
- Integration tests for API endpoints
- End-to-end tests for critical user journeys

Code Review

- Mandatory pull request system
- Review by teammate before merge
- Adherence to established conventions

6 Non-Functional Requirements

⚡ Performance

- ✓ Initial load time under 3 seconds
- ✓ Real-time sync latency under 500ms
- ✓ Support for at least 10 concurrent users

🔒 Security

- ✓ Secure authentication with tokens
- ✓ Injection protection (input validation)
- ✓ CSRF and XSS protection
- ✓ Encrypted communications (HTTPS)

📱 Usability

- ✓ Intuitive and responsive interface
- ✓ Support for major modern browsers
- ✓ Mobile, tablet, and desktop compatibility
- ✓ Visual feedback for user actions

♿ Accessibility

- ✓ Sufficient color contrast
- ✓ Alternative text for visual elements
- ✓ WCAG best practices compliance

🔧 Maintainability

- ✓ Structured and modular code
- ✓ Up-to-date technical documentation
- ✓ Comments for complex logic
- ✓ Scalable architecture

7 Project Schedule

14	39	7
Calendar weeks	Working days	Sprints

Sprint 1: Setup

October 21-22-23 & 28-29-30 (6 days)

Development environment setup

Repository and project structure setup

Docker, CI/CD, and quality tools configuration

Architecture and data model design

Sprint 2: Authentication and Boards

November 4-5-6 & 11-12-13 (6 days)

Full authentication system

Board creation and management

Basic user interface

Sprint 3: Lists and Cards

November 18-19-20 & 25-26-27 (6 days)

CRUD operations for lists

CRUD operations for cards

Board navigation and interface

Sprint 4: Drag & Drop

December 2-3-4 & 9-10-11 (6 days)

Drag-and-drop system for lists and cards

Smooth UI interactions

Sprint 5: Collaboration

December 16-17-18 & 23 + January 6-7-8 (6 days)

Board sharing and member management

Assigning members to cards

Permissions system

Sprint 6: Real-Time Features

January 13-14-15 & 20-21-22 (6 days)

Real-time synchronization with Socket.io

Notifications

Active user presence indicators

Sprint 7: Testing and Finalization

January 27-28-29 (3 days)

Final testing and bug fixes

Documentation completion

Presentation preparation

Project delivery

Note: This condensed schedule prioritizes essential MVP features. Advanced features (comments, labels, checklists, search, history, dark mode) are deprioritized and may be implemented if time allows during later sprints.

8 Deliverables

Source Code

- GitHub repository with complete commit history
- Structured and documented code
- Detailed README.md with installation instructions

Application

- Functional application accessible locally via Docker
- Demo data for testing
- Test user accounts

Technical Documentation

- System architecture and diagrams
- Database schema
- REST API documentation
- Deployment guide

Project Documentation

- Project specifications (this document)
- User stories and backlog
- Sprint reports
- Technical decision justifications
- User guide

Tests

- Automated test suite
- Code coverage report
- Test scenario documentation

9 Risks and Constraints

Identified Risks

- Complexity of multi-user drag & drop
- Managing real-time synchronization conflicts
- Meeting deadlines despite unforeseen events
- Learning curve for technologies used

Constraints

- Development mainly done locally
- Hosting limited to free solutions
- Time budget of 14 weeks (39 working days)
- Small team (2 people)
- Work rhythm: 3 days per week

Mitigation Strategies

- Strict prioritization of features (MVP first)
- Regular testing and continuous integration
- Constant team communication
- Documentation throughout development
- Use of proven and well-documented technologies
- Efficient use of available working days

10 Objectives

The project will be considered successful if:

- Main Kanban features are implemented and functional
- The application is stable and usable
- Agile methodology and TDD have been applied
- The code is high quality, tested, and documented
- Collaboration and CI/CD tools are configured and used
- The project is delivered on time with complete documentation