Feasibility Report

CTRL ALT Elite

Huynh Le, Ethan Gray, Laken Hollen, Kevin Ekart

Product

This project is a digital task board that helps teams organize and manage their work together. It works like a virtual whiteboard with lanes (like "To Do," "In Progress," and "Done") where users can add, move, and track tasks. Each task can show who it's assigned to, when it's due, notes, comments, and a history of changes.

The system makes teamwork easier by giving everyone a clear view of progress, allowing real-time collaboration, and supporting multiple projects in one place. It meets business needs by improving organization, communication, and accountability in a simple, visual way.

Technical Feasibility

This system is technically feasible to build using modern, well-supported web technologies. The proposed architecture consists of a Next.js front end, a Python/Flask back end, and a PostgreSQL database for structured data storage. All of these tools are mature, open source, and supported by large developer communities, ensuring long-term maintainability and extensibility.

Frontend

- React with NextJS
    - Provides a fast, reactive, component-based interface
    - Supports real-time task updates and a drag and drop interface for moving tasks between lanes
- Tailwind CSS and ShadCN UI
    - Allow rapid development of polished, responsive user interfaces
    - Offer accessible, prebuilt components for consistency
    - Facilitate rich UI features such as task detail popovers, tooltips, and modal dialogs

Backend

- Python with Flask
    - Integrates easily with PostgreSQL
    - Suited for real-time features using WebSockets
    - Supports secure user authentication and session management

Database

- PostgreSQL
    - Supports relational schemas for users, projects, tasks, comments, and activity logs

Social Feasibility

The system will be designed to be intuitive and require minimal training for end users. Since it will mirror familiar concepts from tools like Trello, Asana, and Jira, users can quickly adapt to the workflow. Users will be able to perform tasks effectively with minimal need for onboarding. Having a low learning curve, users should be able to adapt to the interface and usage of the platform with little to no friction. Since the application is web-based, users will be able to access it and be trained on it remotely.

Economic Feasibility

Our group is not currently sponsored by an organization, but we can still create a decent estimate of how much this project would cost a company. We must estimate how much the business would pay our group in work fees. There are about 6 months until next April, which is when the project is scheduled to end. A quick Google search led to the average pay of $125,000 per year that a software developer works. There are 4 of us in our group. We will work for about half a year, so that means the business will pay $62,500 per person, or $250,000 for the whole group. Once the project is done, the software won't require any membership to use, so there is no further cost. There may be other costs, such as hardware costs. However, those should be negligible compared to the labor cost, so they will not be considered. So, the final cost is $250,000.

Now, we need to estimate how much the business is currently paying for the existing software. Let's assume that the business uses Slack, as it is one of the most popular apps for task management and project coordination. We will assume that the company is of decent size, with about 500 employees. The Slack fee is usually $8.75 per person per month. So, we estimate that the current software costs the company $4,375 per month, or

$52,500 per year. So, if we divide 250,000 by 52,500, we get that the company will break even 4.76 years after the new software is complete. Thus, in the long run, the new software will help to cut down costs.

There are some other benefits the company will receive from this custom software. Firstly, the features of the software can be developed to better suit the needs of the company's employees. We can work with the employees the whole way to ensure that the software works how they want it to. Also, since the software is made in-house, any future changes or patches that need to be done can be done so in a timelier manner. Overall, the new software will be easier to improve and will be tailored to the needs of the employees.

Market Research

There is significant market demand for our product. For example, Slack, a task management/project collaboration tool similar to ours, is used by over 600,000 companies worldwide. Also, in general, there are over 6,000,000 companies worldwide that use project collaboration software. This data indicates that there are lots of companies that need software like this.

In terms of costs, these tools can be expensive to use. Using Slack as an example, companies need to pay $8.75 per user per month for the pro version, which is the one most companies use based on some research. This means that many companies will have to spend thousands, perhaps tens of thousands of dollars per month to use this software. This is a big deal for smaller companies that don't have as much money to spend. One reason why this software is priced as such is that it includes many extra, non-essential features.

The main way that we can compete in this market is by excluding some of the more advanced features such as AI so that we don't have to charge high prices for our product. We can fill a niche as an affordable task management tool that includes only the essential features. This also has the benefit of making our software simpler to use. In summary, we will compete in this market by appealing to small businesses with software that is relatively cheap and easy to use.

Sources Used:

https://6sense.com/tech/project-collaboration

https://slack.com/pricing

Alternative Solution

Option A: A ready-to-use SaaS Kanban option

Pros: Available now, attractive UI, available mobile apps, a quick start.

Cons: Monthly costs, limited customization, hosting data off-site, limited control over user privacy. Best usage: Teams who value speed of adoption over custom features.

Option B: A low code workspace (i.e. Notion or Airtable)

Pros: Fast startup can easily build template for workflow, simple permissions, no backend to maintain.

Cons: Limited real-time capabilities, limited drag and drop functionality, course specific fields and workflow would be difficult to enforce. Best usage: Very small teams and or short-term projects.

Option C: Custom build using React and FastAPI

Pros: Features exactly fit needs, significant learning opportunity from capstone, full control of data, privacy, exports, easy addition of course specific data fields/reports.

Cons: More engineering work, responsible for maintenance and security, and slower development time to first deployable version. Best usage: Teams that need to create features and control over the data.

Justification

The proposed custom build satisfies the goals of the capstone learning and while providing for a precision which supports graded assessments, privacy and project workflows. The idea of added engineering and development time is within the scope of a semester course when accepting the development of a minimum viable product.

Project Risks

Data loss

Risk: All or part of the data could get deleted accidentally or through server failure.

Mitigation: Expect nightly backups of information, rehearsal of a restore procedure, and have detailed migration scripts that are version controlled.

Performance bottlenecks

Risk: Very large boards will load slowly and updates will feel slow to users.

Mitigation: Create indexes in the database on the project and status fields; consider pagination; only load new data to display increments; and implement simple caching of frequent common queries.

Low adoption

Risk: Users will ignore the board and keep with their old habits.

Mitigation: Ensure basic UI comprehension; use a quick start guide and deliver ten-minute demos; import from csv to get tasks loaded for initial activities; and ensure time saved is shown too.

Accessibility gaps

Risk: There will be insufficient keyboard navigation for keyboard users or screen reader additional features for screen reader users.

Mitigation: Use checks against WCAG 2.2 AA; practice task movement using keyboard-only methods; adding ARIA labels to user interface features; and performing automated audits of your changes.

Team turnover

Risk: There will be too much knowledge focused on one team member for their competency in using the system.

Mitigation: Use shared documentation; conduct code reviews; provide clear READMEs with steps to setup; and have at least two contributors to even the various subsystems.