Scrum & Agile Task Management Tool

Student 1, Student 2, Student 3, Student 4, Student 5, Student 6
Code Repository: https://github.com/**********

***Abstract*—In modern software development processes, an increasing number of teams are adopting distributed development models. To better manage project development progress, project management tools are widely used throughout the entire development process. This paper introduces a Scrum & Agile task management tool developed by our team, based on an open-source project, aiming to provide a detailed account of our efforts over the past months in designing and developing such a tool. The entire development process went through stages such as requirement engineering, analysis and selection of potential technologies, code development, code reviews, and testing. Tools such as GitHub, JIRA, and Overleaf were utilized to assist us in managing the development process. The development phase was divided into three sprints, with the first sprint primarily focusing on requirement engineering, technology selection, and code porting and bug fixing. The second sprint mainly involved refining the tasks completed in sprint 1 and implementing new features. The third sprint was used to enhancing overall system stability and significantly improving the user experience. The final completed project has largely achieved our intended goals, but there are still some shortcomings. We will further enhance this project to provide a comprehensive and diverse software development management tool with improved functionality.**

I. Introduction

NOWADAYS, organizations worldwide strive to enhance their project managements in order to meet their demands. The Scrum framework, coupled with Agile methodologies, has merged as a highly effective method to manage projects. In this report, we propose a Scrum & Agile task management tool combined with the features of flexibility, stability to improve collaboration.

A. The Significance of Project Management

In software development, projects tend to be complex and multi-faceted in nature. To complete a project on time within budget, software team must be in control in efficient planning and organizational monitoring. Project management involves the use of various policies, procedures and principles to plan, implement and complete a project. To ensure a satisfactory result, projects must begin with specified parameters designed to produce the desired outcome. Every project that goes through the project management process follows the system development life cycle that essentially guides the project from start to completion.

Software project management focuses on developing a product that will have a positive effect on an organization. Without project management, a software development team may begin working on a project without any clear vision or guidance, resulting in increased instances of errors and confusion. Moreover, user requirements and other unforeseen events are susceptible to rapid changes during the software development process, so that project management is important for team members to know what happened and how to best handle those situations. Therefore, project management plays a critical role in ensuring product quality and customer satisfaction, addressing challenges in collaboration and communication within development teams.

B. Shortcomings of Conventional Tools

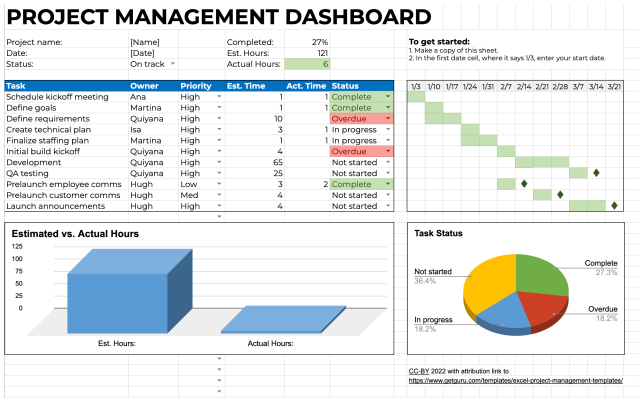


Fig. 1. Conventional task management tools (Excel)

Conventional tools such as Excel, despite their widespread adoption, exhibit considerable constraints with regards to project management applications. These tools lack the capabilities to effectively manage complex projects, leading to issues like miscommunications, scope creep, missed deadlines, and lower-quality outputs. The absence of a structured project management approach may cause inefficiencies and failures. Effective project management ensures clear project scope and objectives, efficient resource allocation, adherence to timelines, and enhanced communication and collaboration among team members. [1] It involves critical tasks like project planning, task management, resource allocation, time tracking, collaboration tools and risk management.

C. Shortcomings of Existing Modern Tools

Currently, there are numerous tools available for project management. One of the most well-known is Atlassian JIRA, which offers many advanced features but has high subscription fees (see Fig. 6). Another comparable tool is Wrike, which resembles JIRA but also utilizes automation and AI, although its subscriptions are similarly costly (see Fig. 6). Therefore, we opted to create our own tool based on user requirements, as we do not require the sophisticated capabilities of these business

platforms and the budget limitation. To meet deadlines, we investigated open-source code projects on GitHub and found options such as ”scrumboard” (yappkahowe) [2], ”scrumboard” (Vaibhav Mehta) [3], and ”scrumblr” (aliasaria) [4] , but these only offered basic task creation functionality, which did not satisfy our users’ needs.

D. How Scrum & Agile Methods Work

Agile and Scrum methodologies have significantly transformed project management in software development, emphasizing adaptability, collaboration, and efficiency. Agile is a broad approach that focuses on incremental delivery, where projects are divided into small, manageable segments, completed in short cycles called sprints. This methodology allows for rapid adjustments and continual learning, making it ideal for projects with evolving requirements.

Scrum, a popular Agile framework, structures project management into sprints, typically lasting two to four weeks. It involves key roles such as the Product Owner, Scrum Master, and Scrum Team, and employs practices like Sprint Planning, Daily Stand-Ups, Sprint Reviews, and Retrospectives. These components ensure continuous feedback and adaptation, fostering a collaborative environment where teams can efficiently deliver high-quality products. Scrum’s emphasis on flexibility and team empowerment makes it particularly effective in dynamic project environments where requirements and goals are subject to change.

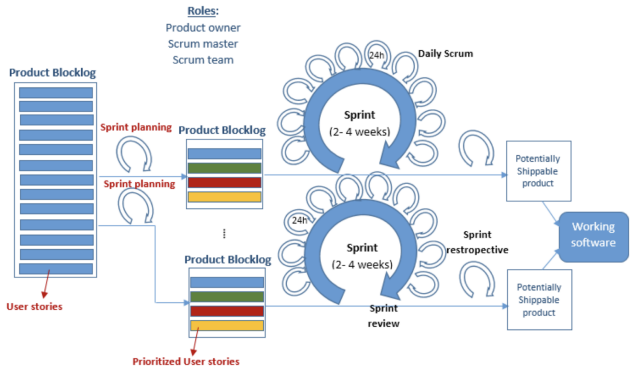


Fig. 2. Scrum method life cycle [5]

E. Our Ideas.

Due to the limitations of both conventional and existing tools, our tool should be more affordable than business platforms and offer more advanced features than these open-source projects. For instance, we did not encounter voting functionality embedded within these open-source tools, which could enable more democratic and collaborative decision-making regarding tasks. Furthermore, we are utilizing lightweight code to reduce resource requirements for both development and deployment.

F. Our Solution

In this report, we present our proposal for a task management tool that leverages the advantages of Scrum and Agile methodologies. Our solution revolves around constructing a task state management system that caters to diverse stories and roles, each with distinct permissions. To realize this, we have designed a web application that integrates various technologies: On the backend, we utilize Node.js with the Express.js framework. For the frontend, we employ TypeScript, supplemented with a small amount of JavaScript, and utilize the React.js framework. As for the database, we opt for a NoSQL database, specifically MongoDB. By combining these technologies, we aim to develop a robust and efficient task management tool that aligns with Scrum and Agile principles.

II. Related Work

A. A simple real-time scrum board

@yappkahowe/scrumboard

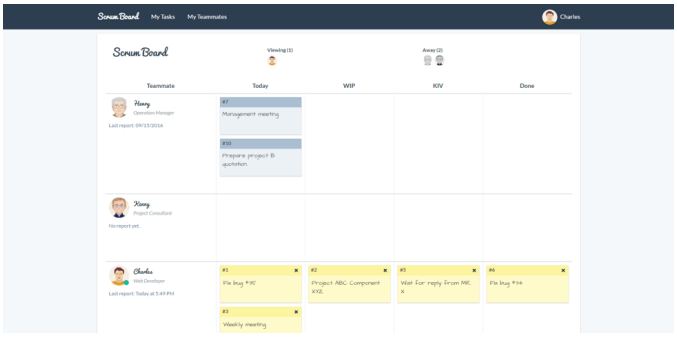


Fig. 3. A simple real-time scrum board

The project in Fig3 is using Laravel, and it is a PHP web application framework. There is lacks proficiency in PHP development experience among our team. By consider the learning curve and the workload may impact the whole project schedule, it will not be the first choice.

Additionally, the project’s design seems to be better for smaller teams, may not be capable if the project scales or the members of the project expands. The limitations of the project may caused more effort to expand, or required directly refactoring. This scalability concern would impact the development.

Furthermore, while the project may good at managing the daily tasks of a small team, it’s not suitable for larger-scale operations. We are aimed to build a scalable and flexible scrum management tools. Thus, it is not a good choice as the starting point of a project because of the efficiency.

B. Collaborative Online Scrum Tool

@aliasaria/scrumblr

The structure of this project in in Fig4 is not organised, the memo are without formats like title, content, task distributions, etc. It will be difficult to separate as different types of memo if later required to classify them or apply filters. It also retracted the extensibility.

It is difficult to extend the text based and stated module. With the unstructured data, it may required preprocessing the data on each extension. For example adding a story category filter, it need to retrieve the category from the text string before filtering.

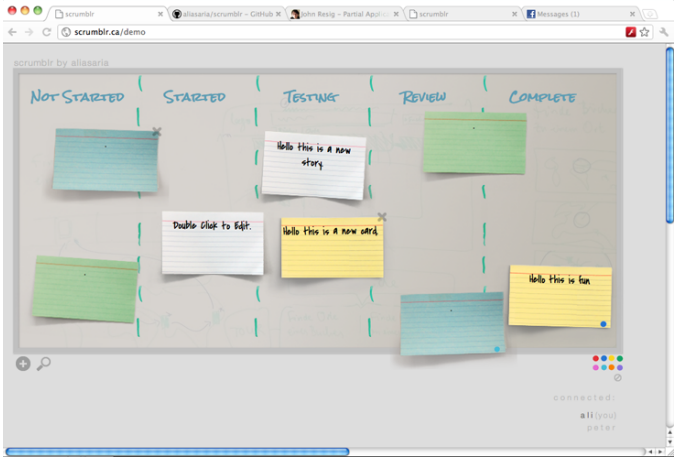


Fig. 4. scrumblr

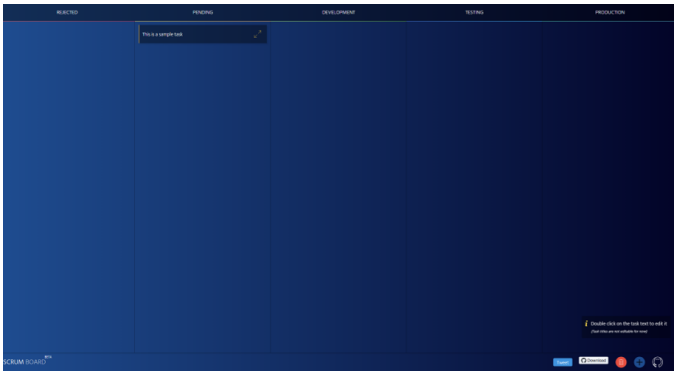


Fig. 5. scrum-board

C. scrum-board

@i-break-codes/scrum-board This is a simple scrum tool in Fig5with client side only, it store the data on local Storage only, that means the records can only see on the local device. It is not allow for a team work together online simultaneously. The project is built by HTML, JavaScript and CSS, with simple file structure and non modularization. For example, most of the actions handling functions are locate in a single JS file. Furthermore, which only work on small scale project because it has only a single story structure with tasks manipulation.

D. Outsourcing

1) Business Tools: We evaluated two popular tools such as Atlassian JIRA and Wrike, since they fulfil our user requirements, which both have board function to handle the backlogs of project management and monitor the progress of the whole project. However, we believe their subscription fees are relatively high, for instance in Figure 6, the subscription fee is around 800USD to 1000USD per month, and our project does not need most of their advanced functionalities such as workflows and dynamic request forms currently, which we just need board function and timeline , so the business tools are not an optimal option.

2) Open-source Tools:

Fig. 6. Atlassian JIRA and Wrike Price Plan

E. Solution

Based on the Final points. We choose Node JS / Express.js as the framework of backend. TypeScript / Express.js as the framework of frontend. Document-Oriented / MongoDB as the Database.

III. Preliminaries

A. Requirement Engineering

1) Requirement Elicitation: we have summarized the following requirements from the interviews. (See Requirements for more detailed)

- R1: Every user needs an account to login the platform.
- R2: Only certain users designated by the platform administrator are eligible to create project discussion groups.
- R3: The creator serves as the initial group leader, and the group leader position can be transferred with mutual agreement. Only the group leader has the authority to add or remove group members and edit the group announcement.
- R4: Discussion group members can create tasks and add information such as priority, expected completion time, and who is responsible for the task.
- R5: Each project has a project announcement, which will be displayed on the homepage of the project. Only the group leader has the ability to edit it.
- R6: Task information should be editable at any time and should be modifiable on both the project and task-specific pages.
- R7: Each task should have a discussion area where users can send text, images, files, and other information to discuss about that specific task.
- R8: The team leader can set anonymous/non-anonymous voting, and everyone can participate.
- R9: The progress of all tasks should be displayed on the timeline.
- R10: This tool should be easy to maintain and expand with new features.

2) Requirement Specification: After group discussion and analysis, we have made the following conclusions about the hierarchical dependencies among the requirements:

Firstly, R1 is the fundamental requirement because every user on the platform needs an account as an identity marker. Additionally, controlling the account issuance ensures that only legitimate users can log in to the platform, thus ensuring security.

Secondly, R2, R3, and R4 are second-level requirements that relate to the basic functionality of the platform, such as

creating project groups and setting development tasks within those groups. These requirements depend on R1, which is the user identity.

Thirdly, R5, R6, R7, R8 and R9 are third-level requirements that involve functionalities like announcements, timeline, maintaining task information, discussion areas and voting. These requirements depend on R1 to R4.

Finally, R10 is a non-functional requirement. This requirement primarily focuses on the maintainability of the software, requiring it to be easy to maintain and upgrade.

There are no conflicting relationships among the mentioned requirements.

3) Requirement Validation & Negotiation: After group discussion, we have concluded that there are no ambiguities, conflicts, or deficiencies in the mentioned requirements, and they are feasible.

We have preliminarily decided to develop each functionality in the order of R1 to R9 to meet the dependencies between the functionalities.

We initially assume the priority order among the requirements as R1 $>$ R2 = R3 = R4 $>$ R5 = R6 = R7 = R8 = R9 = R10, based on their dependency relationships.

After the interviews, the priority is adjusted as follows (TABLE I):

TABLE I
PRIORITY OF REQUIREMENTS

Priority	Requirements
Must Have	R1, R2, R3, R4
Should Have	R6, R8, R9
Could Have	R5, R7, R10

B. Attribute-Driven Design

Considering web applications, we need to think about the backend, the frontend of the application and the database system.

In the backend development of web applications, various combinations of programming languages and frameworks can be utilized (TABLE II). For instance, Node.js paired with Express.js offers a lightweight and efficient solution, particularly suitable for applications requiring a scalable and fast I/O model. Python with Django is another robust choice, known for its clean and pragmatic design, which is highly beneficial for rapid development and clean, pragmatic design. PHP with Laravel is a popular combination due to Laravel's elegant syntax and tools, making it a powerful choice for complex applications. Lastly, Java with Spring is renowned for its comprehensive infrastructure support for developing Java applications, offering a wide range of functionalities and a high degree of flexibility.

For the frontend, different technologies and frameworks cater to various aspects of user interface design and interaction (TABLE III). HTML is the backbone of web page structure, while CSS, enhanced by preprocessors like SASS, offers advanced styling capabilities. JavaScript, a key component of interactive web applications, can be used with frameworks like React.js, Angular JS, or jQuery, each providing unique advantages in terms of UI rendering, single-page application features, and simplicity, respectively. TypeScript, an extension of JavaScript, brings in strong typing and object-oriented features, enhancing code quality and maintainability.

TABLE II
BACKEND

The backend of a web application	Framework of backend
Node.js	Express.js
Python	Django
PHP	Laravel
Java	Spring

TABLE III
FRONTEND

The frontend of a web application	Framework of frontend
HTML CSS JavaScript TypeScript	SASS React.js / Angular JS / jQuery

The choice of database technology is crucial for data management and retrieval IV. Relational Database Management Systems like PostgreSQL and MySQL are widely used for their reliability and support for complex queries. Document-Oriented databases, such as MongoDB, offer flexibility in handling semi-structured data and are highly scalable. In-Memory Data Structure Stores like Redis provide exceptional speed and are ideal for scenarios requiring rapid access to data.

TABLE IV
DATABASE

Families of Technology in Database	Database Choices
Column-Family Relational Database Management System Document-Oriented In-Memory Data Structure Store	PostgreSQL, MySQL, MongoDB Redis

Based on the evaluation of final points focusing on security, maintainability, performance, scalability, and other critical factors, the decision to use Node.js with Express.js for the backend is justified by its high score of 26 points (TABLE V), indicating its efficiency and suitability for the project's needs. For the frontend, JavaScript with React.js, scoring 25 points (TABLE VI), is chosen for its robust ecosystem and efficient UI rendering capabilities. In the database category, the Document-Oriented MongoDB, with the highest score of 25.5 points ((TABLE VII)), is selected for its flexibility and performance in handling large and diverse data sets. These choices reflect a balanced approach to building a web application that is efficient, scalable, and cost-effective. (Go ADD for more detailed information)

The frontend refers to the part of the browser that users see and interact with it. The backend refers to the server-side part

TABLE V
BACKEND FRAMEWORK

Design Decision	Final points
Node JS / Express.js	26
Python /Django	22
PHP / Laravel	20
Java /Spring	22

TABLE VI
FRONTEND FRAMEWORK

Design Decision	Final points
JavaScript / React.js	25
JavaScript / AngularJS	23
JavaScript / jQuery	23
CSS / SASS	21.5

of the web application, which is responsible for processing and storing data, business logic, and interaction with the database. The following items are some brief introductions.

1) Express: Express is a Nodejs framework used for backend/server-side development. It is a popular web application framework for Node.js that provides a minimalistic and flexible set of features for building web applications and APIs. Express has a good performance, Real-time analysis and Express.js itself is open-source and free to use, contributing to cost savings.

2) Node.js: Node.js is an open-source, server-side runtime environment that allows you to run JavaScript code outside of a web browser. Node.js is commonly used for building web servers, APIs, real-time applications, streaming services, microservices, and command-line tools. It has gained popularity due to its performance, scalability, and the ability to leverage JavaScript skills for server-side development.

3) JavaScript: JavaScript is a high-level, interpreted programming language that enables dynamic and interactive behavior on web pages. It is commonly used for frontend development to add functionality, handle events, and manipulate web page elements. JavaScript has a relatively low learning curve, making it accessible to both beginner and experienced developers and it has a vast ecosystem of libraries, frameworks, and tools, providing developers with extensive resources and community support.

4) ReactJS: ReactJS is a declarative, efficient, and flexible JavaScript library for building user interfaces. ReactJS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook. Moreover, React Js makes Front-end development very easy. React.js is designed to optimize UI rendering by efficiently updating only the necessary components when there are changes in data and React.js itself is an open-source library, which means it is free to use and can contribute to cost savings.

TABLE VII
DATABASE

Design Decision	Final points
Document-Oriented / MongoDB	25.5
RDBMS / PostgreSQL	23
RDBMS / MySQL	22
In-Memory Data Structure Store / Redis	21.5

5) Document-oriented: Document-oriented databases store data in a semi-structured format like JSON or XML documents. This flexibility allows for dynamic schemas and makes it easier to store and retrieve hierarchical data structures. It Can distribute data across multiple servers, allowing for horizontal scaling.

6) MongoDB: MongoDB (from ”humongous”) is an open-source document database and the leading NoSQL database. It features JSON-style documents with dynamic schemas, providing simplicity and power. MongoDB supports indexes on any attribute, offers mirroring across LANs and WANs for scalability, and scales horizontally without compromising functionality. It also provides flexible aggregation and data processing capabilities. • MongoDB also has known durability issues (though being fixed), and there can be challenges with repairing databases. Implementing replication setup is necessary to ensure reliability. And it is commonly used for real-time analysis. It supports schema design, indexing, and sharding, making it suitable for real-time analytics workloads.

7) Frontend and Backend: The frontend and backend work collaboratively. The frontend is responsible for displaying and interactive interfaces, sending requests and receiving responses to the backend, while the backend is responsible for processing requests, executing logic and providing data. Through the collaboration of frontend and backend, a complete web application can be built.

IV. SOLUTION

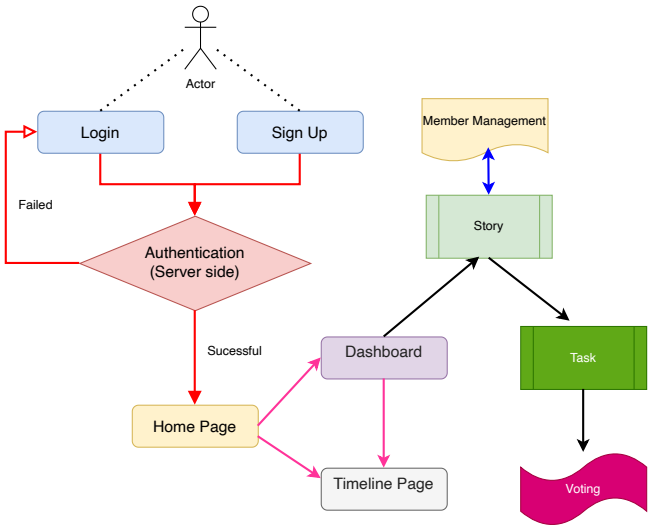


Fig. 7. Walkthrough of our solution

Here is the walkthrough of our solution, as illustrated in Fig. 7. Users are required to login and be authenticated by the

server before accessing the home page. The authentication process also checks the user's role and sends a token to the client side. Once successfully authenticated, users will be redirected to the home page. From there, they can choose between the dashboard and the timeline page for further operations. On the timeline page, a specific task timeline will be displayed. The dashboard enables users to perform create, read, update, and delete (CRUD) operations on stories and tasks. On task level, we have developed a voting function, which made the project more competitive. This voting function allows anonymous or non-anonymous voting in which others can participate. Additionally, for each story, we have implemented a member management feature to control authority.

A. Design Principles and Framework

In the development of our project, we have adhered to the Model-View-Controller (MVC) architectural pattern (Fig. 8), leveraging its advantages to create a well-structured and scalable web application. The backend is powered by Node.js, which acts as the 'Controller' layer, orchestrating the application's logic and handling client-server communication. MongoDB, our chosen database, forms the 'Model' component, responsible for managing the data and ensuring its integrity. This setup allows us to store and retrieve data efficiently, catering to dynamic content management needs. On the frontend, React is utilized to construct the 'View' layer, providing an interactive and responsive user interface. React's component-based architecture aligns seamlessly with the MVC model, enabling us to develop reusable UI components that can respond dynamically to changes in the application's state. This integration of React, Node.js, and MongoDB under the MVC framework not only streamlines the development process but also enhances maintainability and scalability, making it an ideal choice for our web application.

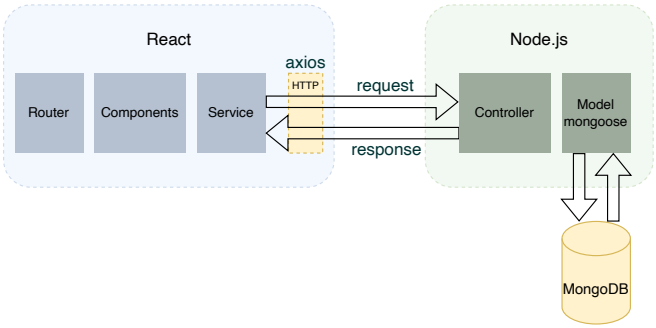
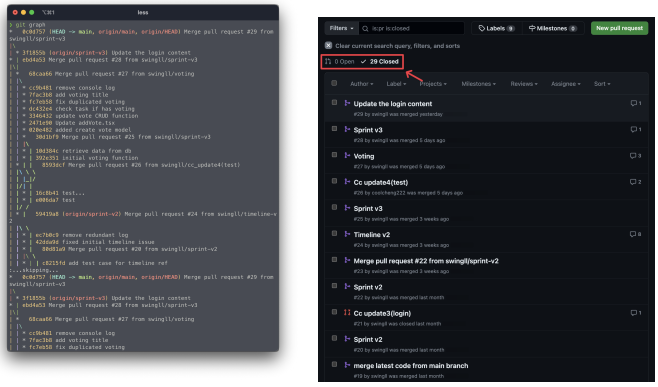


Fig. 8. Web application architecture

B. Modern Code Revieiw

We utilize Git as our version control system and GitHub as the code hosting platform for team collaboration, as illustrated in Figure 9.

Three sub-branches are associated with three sprints 1, 2, 3. Each developer creates their own branch based on the backlog or task and merges it into the respective sub-branches for



(a) Git log graph (b) Pull requests in GitHub

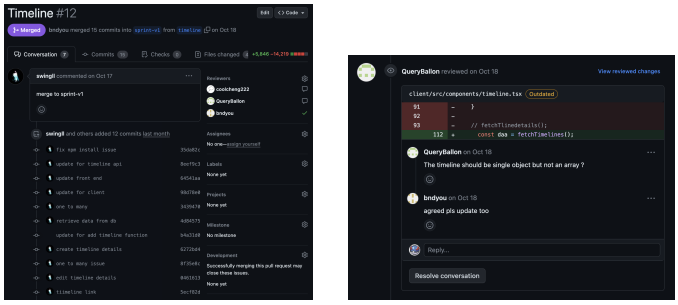
Fig. 9. Using Git as version control system and GitHub as code hosting platform

sprints 1, 2, 3. There are approximately 20 or more MCRs involved.

Our MCR process is as follows:

1) Developers create merge pull requests to merge into the sprint 1, 2, 3 branches, and then other developers review and approve them.
2) If reviewers identify any issues, they request code changes, and the developer needs to make the necessary changes to the code.
3) Once the reviewers are satisfied with the changes, they submit the merge.

Taking the timeline feature PR as an example, Figure 10 showcases our record of PRs conducted through GitHub.



(a) PR of Timeline #12 (b) Code review before merge

Fig. 10. Using Git as version control system and GitHub as code hosting platform

C. Testing

The testing process is a crucial phase that involves evaluating the quality and functionality of our web-based application. In this part, we validated the user interface elements to ensure they meet the desired standards and user expectations.

Figure 11 displays the test code within our project, where each file with the .test.js suffix corresponds to the unit test for its respective module. In this figure, we have designed tests to validate whether the HTTP Status Code meets our expectations.

Figure 12 represents the display of a single run result.

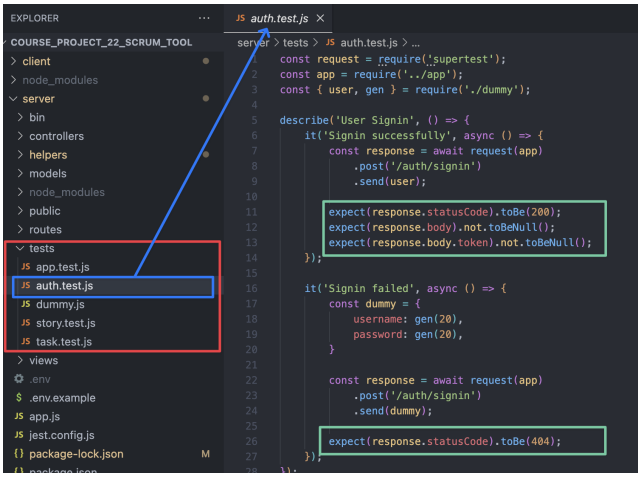


Fig. 11. Test code example

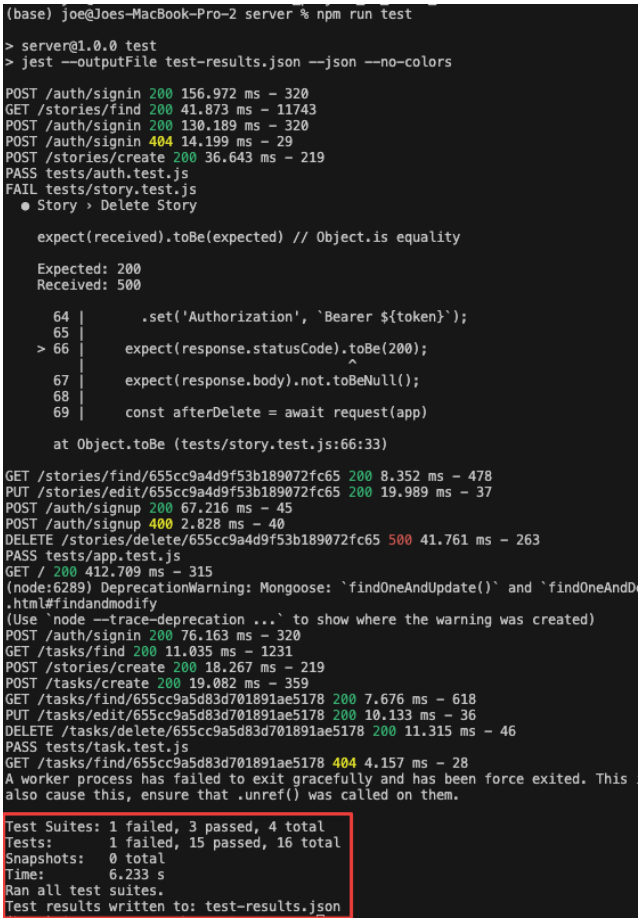


Fig. 12. Testing

D. Technical Debt

In the development of our web-based application, we encountered various challenges that resulted into our technical debt. It refers to the compromises we made in terms of design, implementation, and functionality due to constraints such as limited time.

One of the primary factors contributed to our technical debt was the impact of holidays and the pressure from other academic courses or job. Unexpected events and holidays may disrupt our development schedule.

Besides, another example of technical debt can be observed with the technical transition from React version 16 to version 18 [6]. After the upgrade, there is an issue with the npm command for the node_modules installation. To ensure successful installation, we have a temporary solution, which need to include the --force flag, as demonstrated by the command ”npm install --force”. This workaround is necessary due to the presence of outdated or conflicting dependencies that were not properly resolved during the upgrade process. Such temporary solutions contribute to technical debt and should be addressed in order to maintain a healthy and sustainable application.

V. Software Process

Given the constrained timeline of three months for development,we just define three sprints for the development. And define three sprint branches corresponding to three development phases, consisting of sprint 1, sprint2, sprint3 for a total of 27 backlogs.

TABLE VIII
SPRINT 1

Story ID	Sprint1	Hours
3	Requirement engineering	6
2	Design principle	6
13	Standardized backend modules in project structure	10
12	Standardized frontend modules in project structure	8
25	A poll feature	15
4	Fixed local deployment issue	8
1	Login function	8
7	Swagger API	15
23	Implement function to create new project	6
	Total:	82

TABLE IX
SPRINT 2

Story ID	Sprint2	Hours
14	Timeline visualization about different process	8
8	(Non-functional) Security - Password Encryption	8
9	Restricted story retrievability, only the members of the story can retrieve the story	8
16	MCR and Backend api auto unit test	10
18	Fix issues after login function migration, such as drag task,etx?	10
20	Migrate the timeline function	6
21	Adjust the login page	6
10	Member registration	10
11	Add Roles Permissions	6
	Total:	72

TABLE X
SPRINT 3

Story ID	Sprint3	Hours
6	Access control, such as which role can manage user or just add user for now	6
5	Multiple stories? Such as sprint 1, 2, 3	10
27	handled auth error message	6
19	Fix timeline issue	5
22	Fix some problems, improve user experience	10
17	Frontend auto unit test	10
26	Code Review and debug	10
15	Upgrade the node js version to 18	15
24	npm install issue for node version	15
	Total:	87

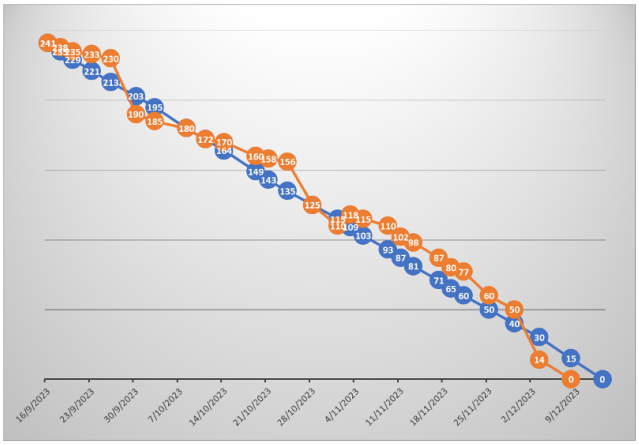


Fig. 13. Burndown chart 1

A. Sprint 1

The first step in the software development process was to conduct an interview meeting with key stakeholders for requirement engineering, such as information collection, elicitation, specification, validation, negotiation, and maintenance of software requirements. These initial requirements elicitation sessions took around three hours to complete as it was important to thoroughly understand the needs and constraints of the project with stakeholders. During the meeting, we asked stakeholders with 16 different questions about the features, functions, and limitations of the system that were wanted and got the information. And then we analyzed and validated those requirements, and negotiated with stakeholders, finally we summarized requirements in the document.

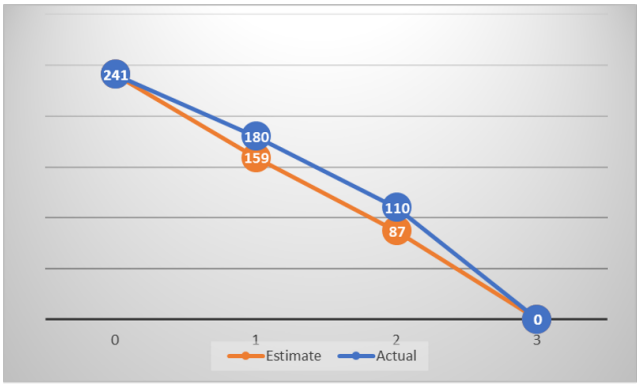


Fig. 14. Burndown chart 2

Requirement#	R1	Requirement Type	Event/BUC/PUC	Platform login
Description:	Every user needs an account to login the platform			
Rationale	1. We need to identify each user; 2. We need to prevent malicious attacks on this platform.			
Originator				
Fit Criterion				
Customer Satisfaction:		5	Customer Dissatisfaction	5
Priority	Must have	Dependences	None	Conflicts	None
Supporting Materials				
Version History				

Fig. 15. Requirement #R1

To begin the project planning, we arrange a higher-level planning to estimate the project structure and select which frameworks are compatible and appropriate with the project. We analyzed several widely backend programming language such as Node.js, Python, Java, and PHP, while analyzed popular frontend programming language such as JavaScript, CSS, TypeScript, and React JS, also included database both relational database non-relational database, such as Oracle DB, MonogoDB. We defined factors matrix tables to compare those technologies, after thorough assessments and discussions in meetings, we finally chose React JS as frontend, Node JS as backend with MongoDB to provide a balanced full-stack solution with good scaling properties and active communities.

Family Cards: Performance, Scalability, Availability, Ad-hoc Analysis

Design Decision	Driver points	Bonus Points	Comments
Column-Family	10		
RDBMS	10		
Document-Oriented	11		
Key-Value Store	9.5		

Technology Cards: Performance, Reliability, Real-time analysis, Cost economy

Design Decision	Driver points	Bonus Points	Comments
MongoDB	14.5	1	It features JSON-style documents with dynamic schemas, providing simplicity and power
PostgreSQL	13	1	
MySQL	12		
Redis	12		

Fig. 16. ADD (take DB as example)

And then we designed a lower-level plan to standardize the project structure. We assume that the end-user sends the related request to the server to login or manage the project in the web application. The server and then send the request to specify controller, and then controller assign the request to the model, which will query information from database and return to controller, finally view show the information to end-user, which retrieve from controller. With the above requirement and design principle, we finally choose one existing open-source code project in GitHub with some basic function and structure, which fulfil our requirement [7].

Based on the MVC model, we divided into three components in the project as follows:

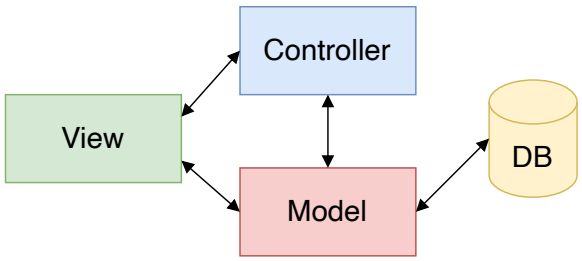


Fig. 17. MVC model structure

TABLE XI
COMPONENTS IN MVC MODEL

Client	View	React JS
Server	Controller	Node JS
	Model	

One of the innovative functions of this project, which allows users to create a poll based on task level (Fig.19), and other users can see a link to poll in the task board, and restrict each user to only vote once per task. We planned to implement it as soon as possible, however, we found that there are bugs that need time to fix when we initial the existing project such as local deployment issues, so we decided to put this function in sprint3.

When we clone the open-source code into local development and follow up the manual of this project to run the project but failed. After we analyzed this problem of deployment, we found the project was not completed, and some key codes were missing. And then we found the fundamental function also was not completed, such as cannot create a story. And we also found the JavaScript type of React JS was outdated.

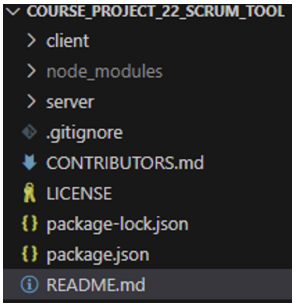


Fig. 18. Initial project code repository

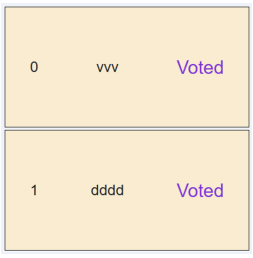


Fig. 19. Poll based on task level

As we know that TypeScript has better autocompletion, navigation, and refactoring services, which can make our further development faster and more efficient. TypeScript supports the static type checking, which makes refactoring safer and easier, TypeScript supports features from upcoming versions of ECMAScript. So, we decided to migrate client code from JavaScript to TypeScript.

Depending on user requirements, the scrum tools require access authentication, which means a login function needs to be implemented. This function provides two ways to log in consisting of email login and username login with a password (Fig.20). When the user clicks the login button, the frontend should check if the required fields are not empty and follow the certain format. A pop-up alert will be displayed if the validation fails, otherwise, this login request will be passed to the backend. Similarly, the backend will also check if the login information is valid, which includes querying if existing user information exists in the database, and then generating a JWT token containing role information to return to the frontend. The user will see the dashboard afterward.

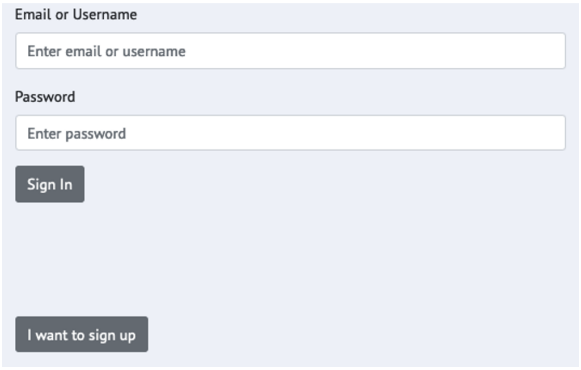


Fig. 20. Login UI

In order to make development efficient, we decided to implement a Swagger API (Fig. 21). As above backlog, we separate the project into both frontend and backend, so that we run the parallel development mode, which allows the frontend team to use the API specification documentation for backend generated by Swagger. As Swagger allows up-to-date information about the API in order to reduce the chances of misunderstandings or miscommunication. Front end team can use it to create the request and handle the response, while the backend can keep developing or testing the API. The frontend and backend team can work simultaneously, which significantly speeds up the development process. By importing the lib and configuring it in the Package.json, and then running the command “npm run swagger-autogen”, we are easy to get the API documentation.

For the client side (frontend), we used Axios to interact with the server side APIs services. Axios is a promise-based HTTP JavaScript library. It can be used to execute HTTP requests, for example GET, POST, DELETE, PUT, etc. It is simple, and flexible to use, it’s also compatible with Reactjs. For the data operation on the client side, we used Redux to handle the data. Redux works particularly well with the Reactjs framework. It describes the UI as a state function, and Redux corresponds

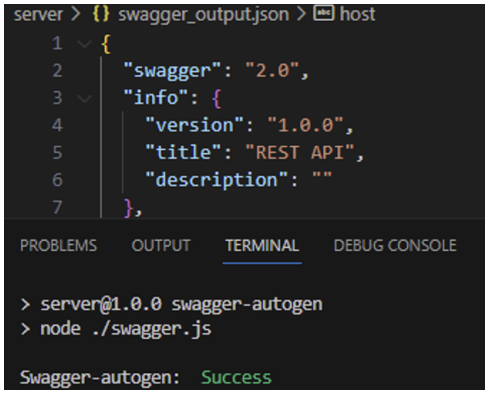


Fig. 21. Login UI

to actions to issue state updates. There are 3 core components in the Redux, namely Action, Reducer and Store.

The Action is JavaScript objects which describe a behaviour, for example click a button, call API, etc. which the behaviour may affect the states. The Store is a centralized container which stores all the state of the application. The state container is globalized, that means it can access all components. The state in the Store can only be updated by dispatching actions.

The Reducer is a function to determine the result of the state changes in application in response to actions sent to the Store. It takes the current state and the value from action to calculate the new state. Reducer is a pure function, it can return the same result for the same input.

When we fixed the local development of the open-source code, we found it was missing the multiple stories creation (Fig.22). Depending on the user requirements, we decided to implement this function in sprint1, since it is a core function in this scrum tool. When we started to do this task, we found it difficult to implement this function as there are existing and unfinished codes between the story and task function. We need to deep dive into the source first before we modify it.

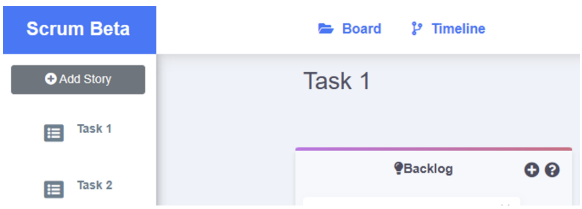


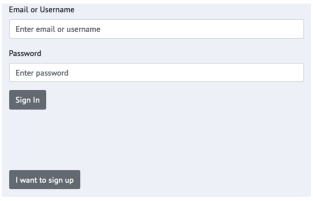
Fig. 22.

B. Sprint 2

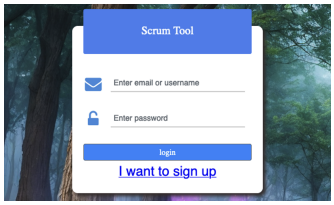
The tasks in the Sprint 2 are mainly focused on cleaning up the tasks done on sprint 1 and developing some new features. The review and clean up are important for the further development, to prevent the unnecessary technical debt produced.

1) User registration: The user registration back-end part has finished on the sprint 1, but not in the front-end part. It is therefore on the Sprint 2, implementing the registration component in the front-end and using the related APIs already developed on the Sprint 1, everyone can register as a new user to the system. The user can create a story and become the owner.

2) adjust the login page: Applied styling and more signin and signup input handling. (Fig. 23)



(a) Sprint 1 Version Auth page



(b) Sprint 2 Version Auth page

Fig. 23. Auth page

3) Restricted story retrievability, only the members of the story can retrieve the story: Apply a filter to the controller of /stories/find route which is used for fetching all stories, the request will return the stories only if the request credential is the owner or member of those stories.

4) Member registration/Member Management: Update the Story structure, added Members to the Story schema. A story owner can add the members to the story. The users are also able to retrieve the stories if the user is the member of the stories. The members of the story can add tasks to the story. The owner can click the button ”Manage Member” to add or remove members of the story.

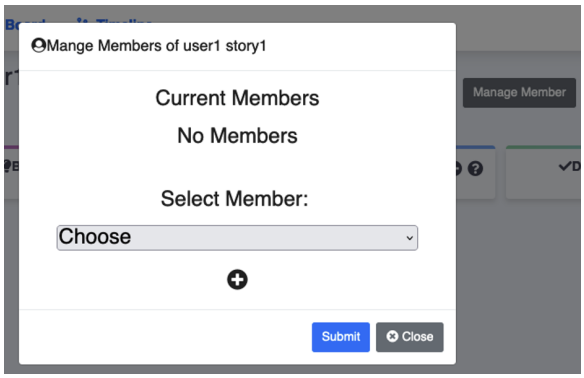


Fig. 24. Member management

5) MCR and Backend api auto unit test: Implement unit test cases for backend parts with the Jest framework. There are three main categories to test, namely Authentication, Story and Task.

For Authentication, there are Signin and Signup test cases. The signup testing is to first randomly generate a dummy credential, then send it with a signup request to process the signup process. The server will return 200 and a JWT token if successful, otherwise 400 or 500.

For Task and Story test cases are similar, all actions required credential, the server will return 500 if the request does not contain credential. Use Story as example, for fetching all stories by credential, The server will return 200 and all the stories related to the credential, while not found any stories

will return 200 and an empty array. There is a test to fetch all stories by story id, server should return 404 if not found by provided story id, otherwise return 200 and the story. There is a test to create a story, send the new story object to the server, the server will return 200 and the new story information which belongs to the credential if the new story passed all conditions, otherwise return the corresponding error code and message, then the test will not pass. There is a test to edit the story, send the request with the story id and the new content, only the owner of the story can edit, if the content passes the conditions, it should return 200 and the updated story, otherwise it should return an error code and message. There is a test for delete story, send the request with story id, if the contained credential is the owner of the story or admin, it should be able to delete the story and server return 200, otherwise it should return error code and message.

```
(base) joe@Joes-MacBook-Pro-2 server % npm run test

> server@1.0.0 test
> jest --outputFile test-results.json --json --no-colors

POST /auth/signin 200 128.271 ms - 320
POST /auth/signin 200 158.085 ms - 320
POST /auth/signin 404 2.715 ms - 29
POST /auth/signup 200 65.812 ms - 45
GET /stories/find 200 76.091 ms - 11264
PASS tests/auth.test.js
POST /auth/signup 400 10.513 ms - 40
POST /stories/create 200 50.696 ms - 219
GET /stories/find/655cc906f81db9168a842f94 200 12.423 ms - 478
PUT /stories/edit/655cc906f81db9168a842f94 200 19.122 ms - 37
(node:5770) DeprecationWarning: Mongoose: `findOneAndUpdate()` and `findOneAndD
.html#findandmodify
(Use `node --trace-deprecation ...` to show where the warning was created)
PASS tests/story.test.js
PASS tests/app.test.js
DELETE /stories/delete/655cc906f81db9168a842f94 200 26.946 ms - 47
GET /stories/find/655cc906f81db9168a842f94 404 6.027 ms - 29
GET / 200 475.348 ms - 315
(node:5768) DeprecationWarning: Mongoose: `findOneAndUpdate()` and `findOneAndD
.html#findandmodify
(Use `node --trace-deprecation ...` to show where the warning was created)
POST /auth/signin 200 69.762 ms - 320
GET /tasks/find 200 12.026 ms - 1231
POST /stories/create 200 17.079 ms - 219
POST /tasks/create 200 15.529 ms - 359
GET /tasks/find/655cc9071a08f81688608d20 200 7.033 ms - 618
PUT /tasks/edit/655cc9071a08f81688608d20 200 9.337 ms - 36
DELETE /tasks/delete/655cc9071a08f81688608d20 200 11.010 ms - 46
GET /tasks/find/655cc9071a08f81688608d20 404 4.403 ms - 28
PASS tests/task.test.js
A worker process has failed to exit gracefully and has been force exited. This
also cause this, ensure that .unref() was called on them.

Test Suites: 4 passed, 4 total
Tests:       16 passed, 16 total
Snapshots:   0 total
Time:        4.257 s
Ran all test suites.
Test results written to: test-results.json
```

Fig. 25. Backend api auto unit test

6) Migrate the timeline function: Migrate an open source Gantt Chart javascript library (dhtmlx-gantt) into the project [8]. The Gantt Chart will show the tasks of the story.

Fig. 26. Gantt chart

7) Roles Permissions: This is a front-end logic to control the permission of different actions (Create, Update, Delete, Read). For example, the code below (Fig. 27) shows getting the permission of the current credential based on the roles from the database and applying the permission to the delete button, which can have better control and organised coding structure in the front-end.

We used Access Control to manage the system permissions. Access Control is a crucial mechanism, aiming to define who can access or manipulate resources in a computing environment. It is paramount, yet flexible to use, and it's compatible with various backend frameworks.

```
const taskDPermission = useAuthorize("task", "D")
```

Fig. 27. Role permission control example

For the permission operations on the server side, we used a Role-Based Access Control (RBAC) system to handle the permissions. RBAC works particularly well with numerous frameworks. It describes the user access as a permission function, and RBAC corresponds to actions to issue permission updates. There are 3 core components in the RBAC, namely User, Role and Permission.

The User is an entity (often a person) which performs actions that may affect the system states.The Role is a symbolic category which stores a set of permissions that a user can have. The permissions in a Role can only be updated by dispatching actions.

The Permission is a definition to determine the result of the access changes in the application in response to actions sent to the Role. It takes the current role and the value from user action to calculate the new access level.

C. Sprint 3

Sprint 3 was highly productive in significantly enhancing overall system stability, substantially elevating user experience, and thoroughly implementing comprehensive quality assurance measures. The dedicated development team diligently worked in a focused manner to successfully complete the high priority user stories that directly address the most critical needs of the product and customers.

For access control, the initial foundational role-based access control (RBAC) model was fully actualized (Fig. 28), which enables the administrator role to completely manage all aspects of users while regular non-administrative users can currently only add new users in a limited capacity at this stage. This lays the essential groundwork for gradually implementing more advanced and granular access control requirements in future sprints. During the extensive design phase, the team had detailed and thoughtful discussions comparing and contrasting RBAC versus attribute-based access control (ABAC) models. Though ABAC provides more dynamic and flexible policies tailored to each individual user, it also brings substantially higher complexity during implementation and maintenance. The team made a well-informed decision to pragmatically implement ABAC in later sprints after extensively testing and validating the simpler RBAC system in a phased approach. Granular permission settings were thoroughly prototyped to eventually support highly specific actions like "User A can view but not edit Module X", thus allowing administrators to finely tune access at a very granular level.

Improving user experience was an important priority focus area for this sprint. The team brainstormed at length during multiple backlog grooming sessions to compile a comprehensive list of major UX pain points that users have reported over time. The two most pressing issues with the highest impact were strategically selected to be addressed in this sprint.

```
// get all stories
router.get( path: '/find', handlers: [jwt.verifyToken, controller.stories]);

// get story by id
router.get( path: '/find/:id', handlers: [jwt.verifyToken, controller.story]);

// create story (developer)
router.post( path: '/create', handlers: [jwt.verifyToken, jwt.isDeveloper, controller.create]);

// edit story by id (developer)
router.put( path: '/edit/:id', handlers: [jwt.verifyToken, jwt.isDeveloper, controller.edit]);

// delete story by id (developer)
router.delete( path: '/delete/:id', handlers: [jwt.verifyToken, jwt.isDeveloper, controller.delete]);
```

Fig. 28. RBAC example

```
npm ERR! Found: react@18.2.0
npm ERR! node_modules/react
npm ERR!   react@"^18.2.0" from the root project
npm ERR!   peer react@"^16.8.0 || ^17 || ^18" from @fluentui/react-component-event-listener@0.63.1
npm ERR!   node_modules/@fluentui/react-component-event-listener
npm ERR!     @fluentui/react-component-event-listener@"~0.63.0" from semantic-ui-react@2.1.4
npm ERR!     node_modules/semantic-ui-react
npm ERR!       semantic-ui-react@"^2.1.4" from the root project
npm ERR!   20 more (@fluentui/react-component-ref, @reduxjs/toolkit, ...)
npm ERR!
npm ERR! Could not resolve dependency:
npm ERR! peer react@"^0.14.0 || ^15.0.0 || ^16.0.0" from react-router@3.2.6
npm ERR! node_modules/react-router
npm ERR!   react-router@"^3.2.1" from the root project
npm ERR!
```

Fig. 29. Version error example

Firstly, ambiguous and unclear authentication error messages were replaced with new polished and descriptive text along with helpful specific instructions guiding users on next steps. For instance, previous generic and confusing error messages like “Login failed” now clearly state “Incorrect username or password entered. Please carefully re-enter credentials or use the Forgot Password option to reset”. Such thoughtful UX tweaks significantly aid confused users during authentication journeys. Secondly, numerous nagging timeline component bugs were finally comprehensively fixed, including pesky refresh data issues as well as erratic scroll positioning problems that users complained about. Smoother timeline interactions provided a much improved user experience for this critical component that users constantly interact with. Beyond these two concrete solutions, fundamental UX principles like self-explanatory error messaging and forgiving interfaces were reinforced across the entire team.

Upgrading the Node.js runtime to the latest robust and performance-focused long-term support (LTS) version 18.12.1 enabled the usage of modern and efficient JavaScript capabilities while also boosting overall system performance. However, inconsistent leftover package-lock artifacts from previous builds caused ‘npm install’ to repeatedly fail on the new runtime version during initial upgrade attempts. The root cause was eventually traced back to mismatches between the old legacy lock files and the new target Node environment after comprehensive debugging and analysis. After multiple unsuccessful attempts to incrementally reconcile the incompatible dependencies, the team decided to ultimately remove the problematic lock files altogether and force complete reinstallation of all dependencies from scratch. This highlighted how vital disciplined version control, proactive dependency management and change planning are when undertaking such major technical upgrades. Important lessons were learned by the team on avoiding dependency drift through clear semantic versioning rules and always rigorously locking down third party package dependencies. As follow-through, the team plans to schedule regular dependency reviews, cleanup and upgrade tasks moving forward.

Sprint 3 mandated rigorous testing and code review practices with the overarching goal of significantly enhancing code quality and stability. Existing frontend unit tests were comprehensively refactored to adopt the improved React Testing Library instead of the outdated Enzyme framework. The new modern library encourages writing robust tests from the user’s perspective and point of view rather than just testing low-level implementation details. Peer code reviews were made mandatory for every pull request to identify defects early and provide thoughtful feedback for improvement. Taken together, these processes collectively improved overall code coverage, long-term maintainability and technical quality. Linting tools and static code analysis solutions were also thoroughly evaluated by the team to supplement automated quality checks in the future.

By the successful conclusion of Sprint 3, the team resolutely resolved over 50 varied issues spanning critical bugs, performance optimizations, UI fixes and technical debt reduction. Burndown charts were analyzed and showed near-perfect alignment between actual progress and initial estimates, indicating stable and predictable velocity. A few lower priority stretch stories were intentionally postponed to the next sprint, as the team pivoted by embracing agile principles and responding to emerging needs. Overall, the team was highly satisfied with the stability and UX improvements achieved through targeted efforts during this sprint. These now provide a rock-solid foundation for implementing the exciting new feature roadmap in upcoming sprints.

In summary, Sprint 3 resulted in tangible gains across access control, UX refinements, major technical upgrades, improved developer practices, and highly effective quality assurance initiatives. Each completed item incrementally advanced the product vision and capabilities in a measurable way. The team is proud of the collaboration culture and knowledge sharing demonstrated, with leaders providing hands-on mentoring to upskill newer members. As specialized expertise and experience deepens in key areas like automated testing, access control patterns, and modern web frameworks, the team is well-positioned to take on more challenging work and level up skills in future sprints.

D. Weekly Evidence

1) We conducted the Atlassian JIRA to manage our project using functions from Kanban template, such as timeline and board (see Fig. 30, 31, 32 and 33).
2) We used GitHub as the code hosting platform for team collaboration, as illustrated in Figure 34. And the commits can be considered as weekly evidence of our development.
3) We used Overleaf as a collaborative editing platform to jointly complete the writing of the report (see Fig. 35).

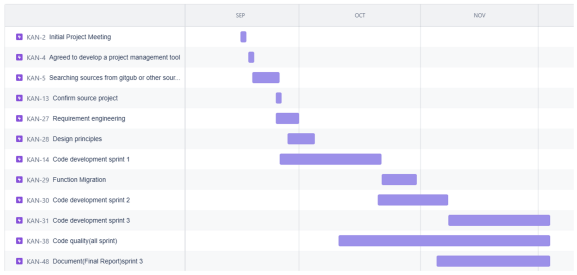


Fig. 30. Using JIRA as project management

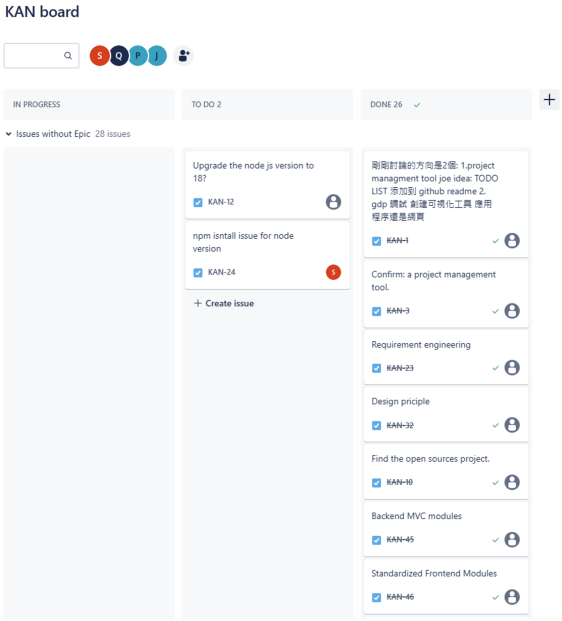


Fig. 31. KAN board (Sprint 1)

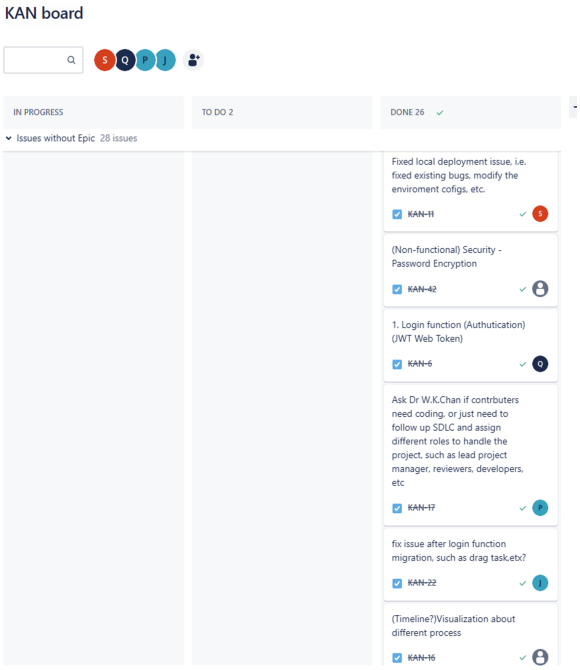


Fig. 32. KAN board (Sprint 2)

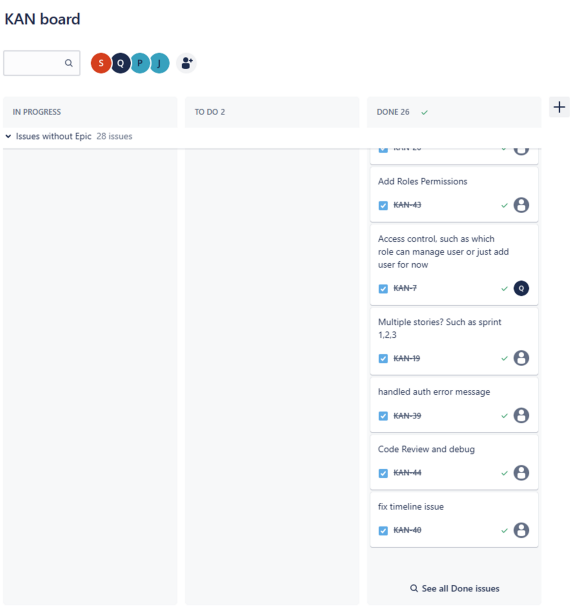


Fig. 33. KAN board (Sprint 3)

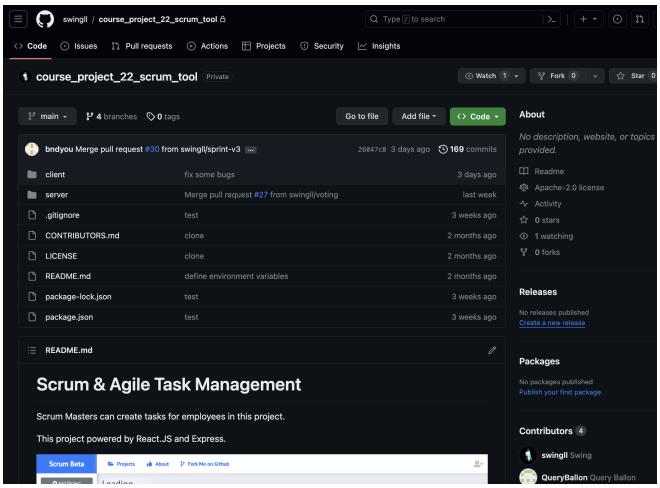


Fig. 34. Repository of our project in GitHub

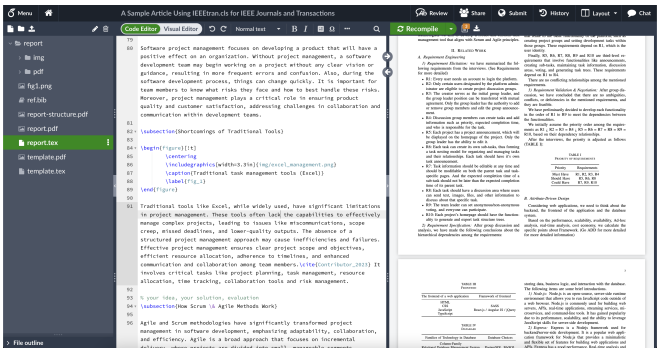


Fig. 35. Using Overleaf for report writng

TABLE XII
REQUIREMENTS VERIFICATION

#	Requirements	Verified
R1	Every user needs an account to login the platform	✓
R2	Only certain users designated by the platform administrator are eligible to create project discussion groups.	✓
R3	The creator serves as the initial group leader, and the group leader position can be transferred with mutual agreement. Only the group leader has the authority to add or remove group members.	✓
R4	Discussion group members can create tasks and add information such as priority, expected completion time, and who is responsible for the task.	✓
R5	Each project has a project announcement, which will be displayed on the homepage of the project. Only the group leader has the ability to edit it.	
R6	Task information should be editable at any time and should be modifiable on both the project task and task-specific pages.	✓
R7	Each task should have a discussion area where users can send text, images, files, and other information to discuss about that specific task.	
R8	The team leader can set anonymous/non-anonymous voting, and everyone can participate.	✓
R9	The progress of all tasks should be displayed on the timeline.	✓
R10	This tool should be easy to maintain and expand with new features.	✓

VI. EVALUATION

Our project's Requirements Verification, as shown in TABLE XII, the majority of requirements have been validated and approved through User Testing. In the evaluation section of our report, it is crucial to highlight the advantages of our tool over conventional tool and existing modern tools, particularly in the scenario of Scrum and Agile task management. Our comparative analysis reveals several key areas where our tool demonstrates superior capabilities.

A. Security

One of the most significant advantages of our project management tool compared to others like Excel or other open-source solutions is its security, which is designed to better protect sensitive project information. With increasing use of distributed development methods in software development process, tools used to manage project progress also need better security.

To address this challenge, our tool is deployed on a private server that requires authentication. Users need to register an account, and only approved users can successfully register. The account approval process and the requirement for an account password ensure that every user of the tool is a verified user, enhancing its security. In contrast, Excel sheets can be easily copied and sent to others without any verification, which poses security risks.

In addition to authentication, our tool provides more advanced functionality to control access at a fine-grained level. Administrators can assign roles with specific permissions to each user. For example, only team leaders have the permission to create tasks, some users may only have access to task information without modification permission, task managers have the permission to modify information, and so on. Role functionality greatly enhances the flexibility and practicality of permission management. In contrast, Excel sheets cannot achieve granular permission control.

Our tool also offers logging functionality, where user access and modifications to task information are recorded. This feature helps manage user behavior and allows for rolling back task information if necessary, ensuring better data security. In contrast, Excel spreadsheets cannot track all modifications made by each user.

B. Visualization

Data visualization plays a crucial role in effective task progress management. In the realm of agile methodologies, understanding the status and relationships between work elements is vital for ensuring smooth workflow and on-time delivery. Compared to alternatives like Excel, our project management tool performs better in visualization, offering clearer and more user-friendly visualization functions.

Core components in our tool, such as task lists and timelines, provide the functionality of displaying information in chronological order, which is lacking in Excel. Users can intuitively view upcoming tasks, ongoing tasks, and completed tasks in a time-ordered manner. The dependencies between tasks and the scheduling of tasks are easier to comprehend compared to the task display in Excel sheets.

Task statuses in our tool are presented through a clear and concise interface. Users can have a unified view of task assignments, task progress, and completion status in a single interface. Additionally, our tool supports real-time collaboration, allowing development teams to simultaneously access and edit task information. This is a significant improvement over the traditional approach of using Excel sheets to record task information and distribute them to individual developers.

C. Data Organization

In Scrum and Agile methodologies, the structure and organization of data are crucial for effective team collaboration and project management. In complex software development and product management processes, there are often many interconnected pieces of information that need to flow seamlessly among team members, such as user stories, tasks, issues, releases, and more. The way data is stored and presented impacts the overall workflow and the ability to make decisions quickly based on data.

Our project management tool was built with this in mind. It utilizes a hierarchical data storage approach, greatly simplifying the management and presentation of relationships

between different task data. For example, user stories can have multiple sub-tasks, each with its own assigned leader, expected completion time, current progress, and other information. This hierarchical organization makes it easier to manage and show task progress.

Compared to alternatives like Excel sheets, the hierarchical data structure is a significant improvement. Excel sheets have limited capabilities in managing relationships and structural information, making it difficult to intuitively display information in a parent-child relationship. While Excel can store basic task information well, its flat structure results in data scattered across different worksheets and cells, making it hard for users to understand the relationships between elements intuitively. It also leads to data duplication, as the same information may need to appear in multiple locations to manage relationship information. Our tool addresses these issues through a dedicated database and a user interface tailored for project management.

Storing task data hierarchically in a central database also provides better data security and integrity. With all task information stored in one database, our tool avoids the inconsistencies that can occur when using Excel sheets, ensuring data integrity. Additionally, users are unable to send task information from the database to others in the same way as copying and sending Excel sheets, ensuring data security. The integration of a dedicated backend database is a key difference of our tool compared to some open-source alternatives, which may only offer UI functionality without persistent data storage capabilities.

The organized data structure also enables the tool to have higher scalability. For example, other developers can implement their own client projects using our API or build additional backend functionality on top of our structure as plugins.

D. Tailored Scrum Development Features

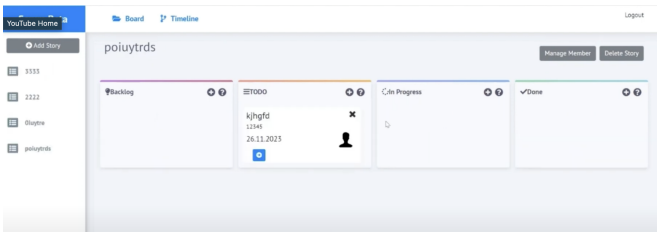


Fig. 36. different states of tasks

Fig. 37. timeline

Our tool is specifically designed to cater to the needs of Scrum development. It is lightweight and offers Story-based task visualization, which is a critical aspect of Agile methodologies. This feature allows for a clear and intuitive view of the different states of tasks (Backlog, TODO, In Progress, and Done) shown in Fig.36, timeline in Fig37, and voting in Fig38, thereby enhancing the user experience and efficiency in managing projects. In contrast, Excel and other open-source free project, despite its wide range of functions, appears as bloatware in this scenario. Its generic design and lack of specialized features for Scrum development make it less suitable for such purposes.

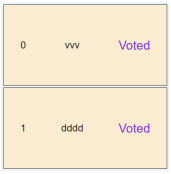
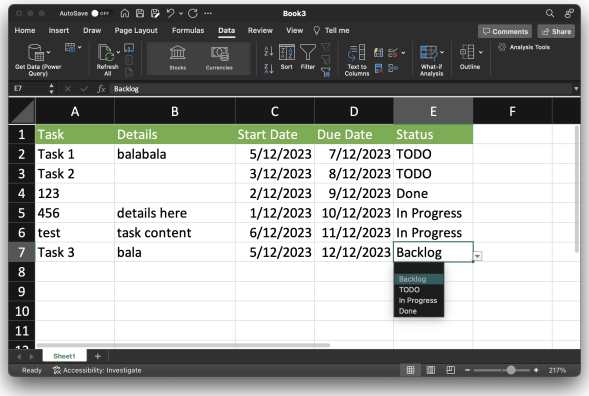


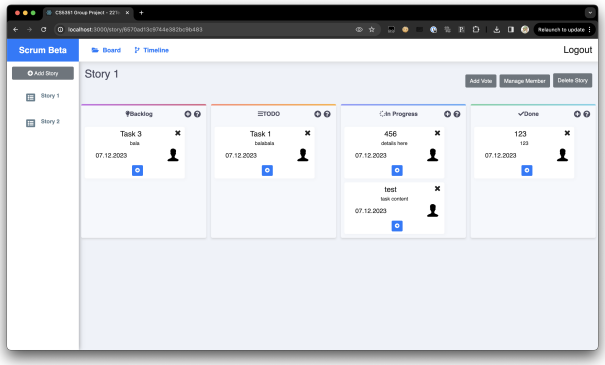
Fig. 38. voting

E. Comparison of Interactions

In the comparative demonstration of functionalities with Excel (as shown in Figure 39), it becomes evident that Excel presents a certain learning curve. Without relying on templates, manual editing is required to create a suitable task management panel. In contrast, our tool features a user-friendly interface that is particularly welcoming to beginners, allowing for intuitive editing of tasks and effortless dragging into different status areas.



(a) Excel

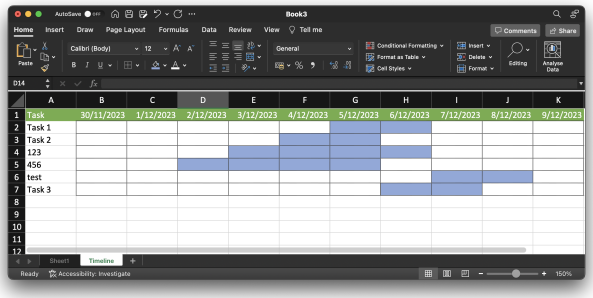


(b) Our tool

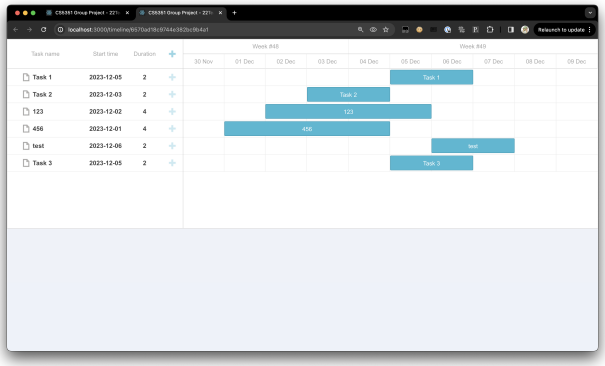
Fig. 39. Comparison of task management

In the timeline tool (as shown in Figure 40), Excel is equally inadequate for this task. Manually dividing time intervals in

Excel does not efficiently depict timelines. In contrast, our tool is better suited for completing this task, offering enhanced capabilities for creating timelines more effectively.



(a) Excel



(b) Our tool

Fig. 40. Comparison of timeline

In summary, we have developed an agile scrum tool for internal use that is more affordable than business tools, and has more advanced features than those open-source projects we found and we evaluated previously, this includes integrated board and timeline functions as well as exciting voting function with stable permission control. Moreover, we conduct software engineering best practices including agile project management, requirements engineering to elicit user needs, attribute-driven design to evaluate frameworks and languages, as well as modern code review and automation test to ensure code quality. This provided greater understanding of the software development lifecycle (SDLC) and further enhanced the scalability and maintainability of our project.

VII. Conclusion

Over the past 3 months, our team has collaboratively developed a Scrum & Agile task management tool that has largely met our expected requirements. Here, we will review the entire development process and propose ideas for further development.

The development process of this project began with requirement engineering. During the requirement engineering phase, we first interviewed stakeholders to understand their expectations for the tool's functionality. After organizing and analyzing the interview content, we identified 10 requirements. Through further discussion and analysis within the team, we made initial speculations about the priority and dependencies of these 10 requirements. In further discussions with stakeholders, they agreed with our analysis of the 10 requirements but provided some suggestions for modifying the priorities. After making further modifications based on their suggestions, both parties reached a consensus on the requirements' priorities.

After clarifying the development requirements, we conducted an evaluation of potential technologies. The technologies we evaluated included the front-end and back-end of the web application, as well as the database. For example, for the backend of a web application and its corresponding framework, we analyzed combinations such as Node.js with Express.js, Python with Django, PHP with Laravel, and Java with Spring. Ultimately, we chose Node.js+Express.js as the implementation for the back-end.

During the development phase, we used JIRA to help manage the development progress and GitHub for team collaboration. We adopted an agile development approach to gradually improve the functionality through iterative development over 3 sprints.

Sprint 1 focused on the initial stages of the software development process, including the requirement engineering and technology evaluation phases described earlier. The project structure was planned using the MVC model. We chose an existing open-source code project as a starting point, but encountered deployment issues and missing key codes, which required us to fix and complete the project. We implemented the login function with authentication and JWT token generation. Swagger API documentation was set up for efficient parallel development. We also added the ability to create multiple stories, although it required delving into the existing code.

In Sprint 2, the focus was on resolving tasks from Sprint 1 and adding new features. Tasks included implementing user registration on the front-end, adjusting the login page, restricting story retrievability to only members, adding member registration and management functionality, and conducting unit testing for authentication, stories, and tasks. Additionally, a Gantt Chart library was integrated for displaying story tasks, and role-based permissions were implemented on the front-end. The sprint aimed to improve user registration, enhance user interface elements, ensure secure access to stories, manage story members, and implement testing and permissions for better control and organization.

Sprint 3 focused on enhancing system stability, improving user experience, implementing access control, upgrading the Node.js runtime, and prioritizing testing and code review. Our team successfully completed medium-priority user stories, implemented role-based access control (RBAC), and addressed major UX pain points. We improved the user experience by enhancing error messages. We upgraded the Node.js runtime, improved code quality through testing and code reviews, and evaluated linting tools. Over 50 issues were resolved, maintaining stable velocity. Lower priority stories were postponed for future development.

Due to time limitation of this project, here are some future development directions for our team:

1) Continue completing lower-priority User Stories that have not been implemented yet. For example, the discussion forum feature suggested in R8. We plan to implement a basic discussion forum in the next iteration, where users can post their ideas about the project in text format. We also plan to add support for images and files in the future.
2) Add email notification functionality. Our initial concept is to send an email notification to users' registered email addresses when they are added to a story by the story owner.
3) Enhance task information by allowing users to set dependencies between tasks and assign responsible users to each task.
4) Introduce template functionality to provide users with a variety of story templates. Users can choose from pre-set templates to expedite the task creation process.

REQUIREMENTS

Requirement#	R1	Requirement Type		Event/BUC/PUC	Platform login
Description:	Every user needs an account to login the platform				
Rationale	1. We need to identify each user; 2. We need to prevent malicious attacks on this platform.				
Fit Criterion	"Each person needs an account to log in to the platform"				
Customer Satisfaction:		5	Customer Dissatisfaction		5
Priority	Must have	Dependences	None	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 41. #R1

Requirement#	R2	Requirement Type		Event/BUC/PUC	Creation of project discussion group
Description:	Only certain users designated by the platform administrator are eligible to create project discussion groups.				
Rationale	Maintaining platform order				
Fit Criterion	"Regarding the creation aspect, only authorized users would be able to create project groups"				
Customer Satisfaction:		5	Customer Dissatisfaction		5
Priority	Must have	Dependences	R1	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 42. #R2

Requirement#	R3	Requirement Type		Event/BUC/PUC	Project group leader
Description:	The creator serves as the initial group leader, and the group leader position can be transferred with mutual agreement. Only the group leader has the authority to add or remove group members.				
Rationale	Maintaining order within the discussion group				
Fit Criterion	"The founder can then invite other users to join the group"				
Customer Satisfaction:		5	Customer Dissatisfaction		5
Priority	Must Have	Dependences	R1,R2	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 43. #R3

Requirement#	R4	Requirement Type		Event/BUC/PUC	Task creation
Description:	Discussion group members can create tasks and add information such as priority, expected completion time, and who is responsible for the task.				
Rationale	Dividing the project into sub-tasks to facilitate project planning				
Fit Criterion	"Additionally, we would like to create tasks within the group"				
Customer Satisfaction:		5	Customer Dissatisfaction		5
Priority	Must Have	Dependences	R1,R2,R3	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 44. #R4

Requirement#	R5	Requirement Type		Event/BUC/PUC	Announcement
Description:	Each project has a project announcement, which will be displayed on the homepage of the project. Only the group leader has the ability to edit it.				
Rationale	Use announcements to remind team members of important matters and share relevant information.				
Fit Criterion	"There should be a group announcement that can be edited by the group leader. "				
Customer Satisfaction:		3	Customer Dissatisfaction		2
Priority	Could Have	Dependences	R1,R2,R3,R4	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 45. #R5

Requirement#	R6	Requirement Type		Event/BUC/PUC	Maintaining task information
Description:	Task information should be editable at any time and should be modifiable on both the project and task-specific pages.				
Rationale	Utilizing task information for better task management.				
Fit Criterion	"I hope it can be done on both levels"				
Customer Satisfaction:		4	Customer Dissatisfaction		3
Priority	Should Have	Dependences	R1,R2,R3,R4	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 46. #R6

Requirement#	R7	Requirement Type		Event/BUC/PUC	Task discussion area
Description:	Each task should have a discussion area where users can send text, images, files, and other information to discuss about that specific task.				
Rationale	Facilitate users to discuss task-related content on the platform.				
Fit Criterion	"Each task should have its own dedicated discussion space"				
Customer Satisfaction:		3	Customer Dissatisfaction		2
Priority	Could Have	Dependences	R1,R2,R3,R4	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 47. #R7

Requirement#	R8	Requirement Type		Event/BUC/PUC	Vote
Description:	The team leader can set anonymous/non-anonymous voting, and everyone can participate.				
Rationale	Collective decision-making for future development direction, with anonymous voting to reflect true opinions.				
Fit Criterion	"Oh, and I also hope to have a voting feature that can be used for group decision-making regarding future development directions"				
Customer Satisfaction:		4	Customer Dissatisfaction		3
Priority	Should Have	Dependences	R1,R2,R3,R4	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 48. #R8

Requirement#	R9	Requirement Type		Event/BUC/PUC	Timeline
Description:	The progress of all tasks and their subtasks should be displayed on the timeline.				
Rationale	Visualizing task progress using a timeline display.				
Fit Criterion	"I would like to have some way to show the deadlines of each task"				
Customer Satisfaction:		4	Customer Dissatisfaction		3
Priority	Should Have	Dependences	R1,R2,R3,R4	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 49. #R9

Requirement#	R10	Requirement Type		Event/BUC/PUC	Maintainability
Description:	This tool should be easy to maintain and expand with new features.				
Rationale	This tool should have a high level of maintainability.				
Fit Criterion	"The tool should be easy to maintain and expand"				
Customer Satisfaction:		3	Customer Dissatisfaction		2
Priority	Could Have	Dependences	None	Conflicts	None
Supporting Materials	Interview with stakeholders				

Fig. 50. #R10

ADD

The backend of a web application

Node.js

Description: Node.js is an open-source, server-side runtime environment that allows you to run JavaScript code outside of a web browser. Node.js is commonly used for building web servers, APIs, real-time applications, streaming services, microservices, and command-line tools. It has gained popularity due to its performance, scalability, and the ability to leverage JavaScript skills for server-side development.

Consequences:

- ★★★ **Performance**: Node.js is known for its excellent performance due to its event-driven, non-blocking I/O model.
- ★★★ **Scalability**: Node.js is highly scalable, making it suitable for building scalable applications. Its event-driven architecture allows for handling a large number of concurrent requests without blocking the event loop.
- ★★★ **Availability**: Node.js is widely available and supported across various platforms, including Windows, macOS, and Linux.
- ★★★ **Ad-hoc analysis**: Support complex real time applications.

Python

Description: Python is a programming language that lets you work quickly and integrate systems more efficiently. It is also a very important language for the back end and for mastering it you can take a look at Python Programming Foundation -Self-Paced course. This is a beginner-friendly course and will help you to build a strong foundation for python.

- ★★ **Performanc**e: Python is an interpreted language, which means it may not offer the same level of performance as compiled languages like C or Java.
- ★★★ **Scalability**: Python is considered highly scalable due to its ability to handle large-scale projects and distributed systems.
- ★★★ **Availability**: Python is widely available and supported on various platforms, including Windows, macOS, and Linux. It has a vast ecosystem of libraries and frameworks that cater to different domains and use cases.

- ★★★ **Ad-hoc Analysis**: Python is a popular choice for ad-hoc analysis and data exploration tasks. It offers powerful libraries like NumPy, Pandas, and Matplotlib.

PHP

Description: PHP is a server-side scripting language designed specifically for web development. Since PHP code is executed on the server side, it is called a server-side scripting language.

- ★★★ **Performance**: PHP is known for its good performance in web applications. It is a compiled language, and modern PHP versions have made significant improvements in performance.
- ★★★ **Scalability**: PHP is highly scalable and can handle large-scale projects and high traffic websites. It offers built-in features like session management, load balancing, and support for distributed computing.
- ★★★ **Availability**: PHP is widely available and supported across various platforms. It is compatible with major operating systems like Windows, macOS, and Linux.
- ★ **Ad-hoc Analysis**: PHP is primarily used for web development, not for Ad-hoc Analysis.

Java

Description: Java is one of the most popular and widely used programming languages and platforms. It is highly scalable. Java components are easily available and for learning this one of the most popular languages you can check the Geeksforgeeks Java Programming Foundation – Self-Paced course. It will help you understand the proper framework, concepts, functions, and more.

- ★★★ **Performance**: Java is renowned for its performance, thanks to its Just-In-Time (JIT) compilation and efficient garbage collection.
- ★★★ **Scalability**: Java is highly scalable and widely used for building enterprise-level applications. It offers features like multithreading, concurrency utilities, and distributed computing frameworks.
- ★★★ **Availability**: Java is platform-independent, allowing applications to run on various operating systems like Windows, macOS, and Linux.
- ★★ **Ad-hoc Analysis**: Java is suitable for ad-hoc analysis and data processing tasks.

PHP / Laravel

Description: Laravel is a web application framework for PHP and is robust. The feature which makes it perfect is reusing the components of different frameworks for creating a web application.

- ★★★ **Performance**: Laravel provides various performance optimizations to enhance application speed. It includes features like route caching, query caching, and optimized autoloading to minimize response times.
- ★★★ **Reliability**: Laravel emphasizes reliability by providing built-in security features and practices.
- ★ **Real-time analysis**: Laravel is primarily designed for building traditional request-response web applications, rather than real-time analysis systems.
- ★★★ **Cost economy**: Laravel is an open-source framework, meaning it is free to use and can contribute to cost savings.

Java / Spring

Description: This server-side framework provides infrastructure support for Java applications. It acts as a support to various frameworks like Hibernate, Struts, EJB, etc. It also has extensions that help in developing Java applications quickly and easily.

- ★★★ **Performance**: Spring provides various performance optimizations to enhance application speed. It offers features like caching, connection pooling, and asynchronous processing to improve performance.
- ★★★ **Reliability**: Spring promotes reliability by offering robust error handling, exception management, and security features.
- ★★ **Real-time analysis**: Spring is primarily designed for building traditional request-response web applications, but it can be integrated with other tools and libraries to enable real-time analysis.
- ★★★ **Cost economy**: Spring is an open-source framework, meaning it is free to use and can contribute to cost savings.

Framework of Backend

Node.js / Express.js

Description: Express is a Nodejs framework used for backend/server-side development. It is a popular web application framework for Node.js that provides a minimalistic and flexible set of features for building web applications and APIs.

- ★★★ **Performance**: Express.js is known for its lightweight and efficient design, which contributes to good performance.
- ★★ **Reliability**: Express.js itself doesn't guarantee reliability, but it provides a solid foundation for building reliable applications.
- ★★★ **Real-time analysis**: Express.js can handle real-time communication using techniques like WebSockets and server-sent events.
- ★★★ **Cost economy**: Express.js itself is open-source and free to use, contributing to cost savings.

Python / Django

Description: Django is a Python web-based framework, following the model-template-views pattern. It is used to build large and complex web applications. Its features include being fast, secure, and scalable.

- ★★★ **Performance**: Django is known for its performance, because it has various optimizations and caching mechanisms.
- ★★★ **Reliability**: Django is designed with a focus on reliability and stability. It provides built-in security features, including protection against common web vulnerabilities.
- ★ **Real-time analysis**: Django is primarily designed for building traditional request-response web applications, rather than real-time analysis systems.
- ★★★ **Cost economy**: Django itself is open-source and free to use, contributing to cost savings.

Family Cards: Performance, Scalability, Availability, Ad-hoc Analysis

Design Decision	Driver points	Bonus Points	Comments
Node JS	12	2	Develop a web application, Node JS allows you to run JavaScript code outside of a web browser
Python	10		
PHP	10		
Java	11		

Technology Cards: Performance, Reliability, Real-time analysis, Cost economy

Design Decision	Driver points	Bonus Points	Comments
Express JS	11	1	It is a popular web application framework for Node.js that provides a minimalistic and flexible set of features for building web applications and APIs.
Django	10	1	Django is also a Python web-based framework, following the model-template-views pattern.
Laravel	10		
Spring	11		

The Frontend of a web application

HTML

Technology/Markup Language/Frontend

Description: HTML (Hypertext Markup Language) is the standard markup language for creating web pages and applications. It provides a structured way to define the content and layout of web pages using tags and elements.

Consequences:

- ★ **Interactivity**: HTML alone has limited interactivity capabilities and is primarily used for defining the structure of web pages.
- ★★★ **Browser Compatibility**: HTML is supported by all modern web browsers, ensuring broad compatibility for web applications.
- ★★ **Ease of Use**: HTML is relatively easy to learn and use, making it a popular choice for

building web applications.

- ★★★ **Integration**: HTML can be seamlessly integrated with other frontend technologies like CSS and JavaScript to create interactive and visually appealing web experiences.
- ★★★ **Community and Compatibility**: HTML has a large and active community, offering extensive documentation, tutorials, and resources for developers. HTML also enjoys high compatibility across browsers and devices.

CSS

Technology/Style Sheet Language/Frontend

Description: CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a document written in HTML. It provides a set of rules for controlling the appearance and layout of web pages.

Consequences:

- ½ **Interactivity**: CSS alone does not have built-in interactivity features like handling user input or complex interaction logic.
- ★★★ **Browser Compatibility**: CSS is widely supported by modern web browsers, ensuring consistent rendering across platforms.
- ★★ **Ease of Use**: CSS is relatively easy to learn and implement, offering a straightforward way to style web pages.
- ★★★ **Integration**: CSS can be seamlessly integrated with other frontend technologies like HTML and JavaScript to create interactive and visually appealing web experiences.
- ★★★ **Community and Compatibility**: CSS has a large community and enjoys high compatibility across various web browsers and devices.

JavaScript

Technology/Programming Language/Frontend

Description: JavaScript is a high-level, interpreted programming language that enables dynamic and interactive behavior on web pages. It is commonly used for frontend development to add functionality, handle events, and manipulate web page elements.

Consequences:

- ★★★ **Interactivity**: JavaScript allows developers to create interactive web experiences by adding event handling, form validation, animations, and other dynamic behaviors.
- ★★★ **Browser Compatibility**: JavaScript is supported by all modern web browsers, ensuring broad compatibility and consistent execution of code.
- ★★ **Ease of Use**: JavaScript has a relatively low learning curve, making it accessible to both beginner and experienced developers.
- ★★★ **Versatility**: JavaScript can be used for frontend and backend development, thanks to technologies like Node.js, expanding its application beyond the browser.
- ★★★ **Community and Compatibility**: JavaScript has a vast ecosystem of libraries, frameworks, and tools, providing developers with extensive resources and community support.

TypeScript

Technology/Programming Language/Frontend

Description: TypeScript is a superset of JavaScript that adds static typing and additional language features. It aims to enhance JavaScript's development experience by providing improved type checking, autocompletion, and error detection during development.

Consequences:

- ★★★★ **Interactivity**: TypeScript enhances the interactivity of web applications by enabling developers to build more robust, scalable, and maintainable codebases.
- ★★★ **Browser Compatibility**: TypeScript is designed to be compatible with all major web browsers. Since TypeScript is a superset of JavaScript, the generated JavaScript code from TypeScript can be executed in any browser that supports JavaScript.
- ★★ **Ease of Use**: TypeScript builds upon JavaScript, making it relatively easy for JavaScript developers to adopt and utilize its additional features.
- ★★★ **Versatility**: Its versatility stems from its ability to be used in a wide range of contexts, including web development, server-side development, desktop applications, mobile applications, and more.
- ★★★ **Community and Compatibility**: TypeScript has gained significant popularity, resulting in a large and active community. It works seamlessly with existing JavaScript code and libraries.

Framework of Frontend

JavaScript / React JS

Description: React is a declarative, efficient, and flexible JavaScript library for building user interfaces. ReactJS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook. Moreover, React Js makes Front-end development very easy.

- ★★★ **Performance**: React.js is designed to optimize UI rendering by efficiently updating only the necessary components when there are changes in data.
- ★★★ **Reliability**: React.js promotes reliability by enforcing a unidirectional data flow and providing a declarative approach to building UIs. This helps in writing predictable and maintainable code
- ★★ **Real-time analysis**: React.js primarily focuses on the UI layer and doesn't directly address real-time analysis.
- ★★★ **Cost economy**: React.js itself is an open-source library, which means it is free to use and can contribute to cost savings.

JavaScript / AngularJs

Description: AngularJs is a JavaScript open-source front-end framework that is mainly used to develop single-page web applications(SPAs). It is a continuously growing and expanding framework which provides better ways for developing web applications. It changes the static HTML to dynamic HTML. It is an open-source project which can be free. It extends HTML attributes with Directives, and data is bound with HTML.

- ★★ **Performance**: AngularJS provides features like two-way data binding, dependency injection, and a digest cycle that optimize performance. However, as the complexity of AngularJS applications increases, it may lead to performance challenges.
- ★★★ **Reliability**: AngularJS promotes reliability through its robust architecture and error handling mechanisms. It follows the MVC (Model-View-Controller) pattern, separating concerns and making code more maintainable.
- ★ **Real-time analysis**: AngularJS is primarily designed for building traditional web applications and does not directly support real-time analysis out of the box.
- ★★★ **Cost economy**: AngularJS is an open-source framework, which means it is free to use and can contribute to cost savings.

JavaScript / jQuery

Description: jQuery is an open-source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript. Elaborating on the terms, jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

- ★★ **Performance**: jQuery is designed to provide a simplified API for common web development tasks. It is generally lightweight and optimized for performance.
- ★★★ **Reliability**: jQuery has been widely adopted and extensively tested, making it a reliable choice for web development.
- ★ **Real-time analysis**: jQuery itself does not directly support real-time analysis.
- ★★★ **Cost economy**: jQuery is an open-source library, making it freely available and contributing to cost savings.

CSS / SASS

Description: It is the most reliable, mature, and robust CSS extension language. It is used to extend the functionality of an existing CSS of a site including everything from variables, inheritance, and nesting with ease.

- ★★★ **Performance**: Sass offers features like variables, mixins, and nesting, which improve code organization and maintainability. These features can also lead to more efficient CSS generation by reducing redundancy and improving code reuse.
- ★★★ **Reliability**: Sass introduces programming concepts like variables, functions, and conditionals, allowing for more robust and maintainable stylesheets.
- ★ **Real-time analysis**: Sass itself does not directly support real-time analysis.
- ★★★ **Cost economy**: Sass can contribute to cost savings by improving developer productivity and reducing maintenance efforts.

Family Cards:

Design Decision	Driver points	Bonus Points	Comments
HTML	12	1	HTML has ability to provide structure, cross-browser compatibility, accessibility, and a content-first approach.
CSS	11.5	2	CSS enables the customization of fonts, colors, layouts, and other visual aspects of web pages, enhancing the user experience.
JavaScript	14		
TypeScript	15	1	TypeScript's static typing helps catch errors during development, increasing code reliability and maintainability.

Technology Cards:

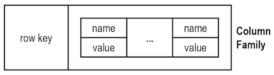
Design Decision	Driver points	Bonus Points	Comments
React JS	11	1	React Js makes Front-end development very fast and easy.
AngularJs	9		
jQuery	9		
SASS	10		

Families of Technology in Database

Column-Family

Family/Data Storage/NoSQL Database

Description: Extends Key-Value databases by storing not strictly defined collections of one or more key-value pairs that match a record. Can be presented as two dimensional arrays whereby each key has one or more key-value pairs attached to it.



Consequences:

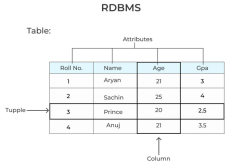
- ★★★ **Performance**: Extremely fast due to absence of schema definition, relational, transactional or referential integrity functionality
- ★★★ **Scalability**: Can be linearly scaled by splitting data across servers using hash value calculated based on a row key
- ★★★ **Availability**: High availability is provided by clustering and distributed file system (e.g. HDFS)
- ★ **Ad-hoc Analysis**: Supports secondary indexing, but no aggregate functions

Sample implementations: HBase

Relational Database Management System (RDBMS)

Technology/Data Storage/SQL Database

Description: RDBMS is a traditional database family that uses a relational model with tables, rows, and columns. It provides a structured way to store and organize data, making it suitable for projects with well-defined schemas and relationships between entities.



Consequences:

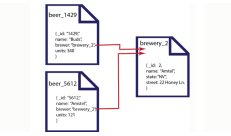
- ★★★★ **Performance**: Extremely fast due to schema definition and relational, transactional, and referential integrity functionality.
- ★★ **Scalability**: Can handle large datasets, but scaling can be more challenging compared to other database families.
- ★★★ **Availability**: Provides high availability through replication mechanisms.
- ★★ **Ad-hoc Analysis**: Supports advanced analysis with aggregate functions and querying capabilities.

Sample implementations: MySQL, PostgreSQL

Document-Oriented

Technology/Data Storage/NoSQL Database

Description: Document-oriented databases store data in a semi-structured format like JSON or XML documents. This flexibility allows for dynamic schemas and makes it easier to store and retrieve hierarchical data structures.



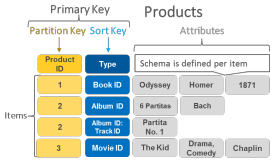
Consequences:

- ★★★½ **Performance**: Efficient for document-oriented queries.
- ★★★ **Scalability**: Can distribute data across multiple servers, allowing for horizontal scaling.
- ★★★ **Availability**: High availability with help of sharding.
- ★½ **Ad-hoc Analysis**: Supports schema-less design, but may require additional effort for data consistency.

Sample implementations: MongoDB

Key-Value Store

Description: Key-value stores provide a simple and efficient way to store and retrieve data using a key-value pair model. They are suitable for scenarios where the structure of the data is relatively simple and the emphasis is on fast retrieval and scalability.



Consequences:

- ★★ **Performance**: Fast read and write operations due to the simplicity of the key-value model. However, compared to other database models like relational databases, deleting or updating individual values in a Key-Value store can sometimes be more challenging.
- ★★★★ **Scalability**: Highly scalable with linear scaling using hash-based data distribution.
- ★★★ **Availability**: Provides high availability through replication mechanisms.
- ½ **Ad-hoc Analysis**: Primarily supports simple key-based retrieval; complex queries may be challenging.

Sample implementations: Redis

Database Choices

MongoDB

Technology/Data Storage/NoSQL Database/Document-Oriented

Description: MongoDB (from "humongous") is an open-source document database and the leading NoSQL database. It features JSON-style documents with dynamic schemas, providing simplicity and power. MongoDB supports indexes on any attribute, offers mirroring across LANs and WANs for scalability, and scales horizontally without compromising functionality. It also provides flexible aggregation and data processing capabilities.

Consequences:

- ★★★★ **Performance**: MongoDB is not as fast as simplest key-value storages, but features like auto-sharding, full index support, and map-reduce make it fast enough. It is written in C++.
- ★★★★ **Reliability**: MongoDB has known durability issues (though being fixed), and there can be challenges with repairing databases. Implementing replication setup is necessary to ensure reliability.
- ★★★ **Real-time analysis**: MongoDB is commonly used for real-time analysis. It supports schema design, indexing, and sharding, making it suitable for real-time analytics workloads.
- ★★★½ **Cost economy**: MongoDB is released as open source under the terms of the GNU AGPL license, and commercial licenses are also available.

PostgreSQL

Technology/Data Storage/Relational Database Management System (RDBMS)

Description: PostgreSQL is a powerful open-source RDBMS known for its robustness, extensibility, and adherence to SQL standards. It offers features such as ACID-compliant transactions, advanced indexing, and support for complex data types. PostgreSQL provides a wide range of functionality and is highly customizable.

Consequences:

- ★★★★ **Performance**: PostgreSQL offers good performance, especially for complex queries and data manipulations. It provides various optimization techniques and allows fine-tuning for specific workloads.

- ★★★ **Reliability**: PostgreSQL is known for its reliability, durability, and data integrity. It provides features like write-ahead logging and crash recovery mechanisms to ensure data consistency.
- ★★ **Real-time analysis**: PostgreSQL supports real-time analysis through its advanced querying capabilities, support for indexing, and features like partitions and parallel execution.
- ★★★★ **Cost economy**: PostgreSQL is an open-source database released under the PostgreSQL License, which allows free usage and distribution. It offers a cost-effective option for projects.

MySQL

Technology/Data Storage/Relational Database Management System (RDBMS)

Description: MySQL is a widely used open-source RDBMS known for its performance, reliability, and ease of use. It offers features like ACID compliance, transaction support, and a range of storage engines to optimize for different workloads. MySQL is suitable for various applications and scales well.

Consequences:

- ★★★ **Performance**: MySQL provides good performance for general-purpose use cases. It has optimizations for read-heavy workloads and supports caching mechanisms to improve performance.
- ★★★ **Reliability**: MySQL is known for its reliability and stability. It offers features like replication, backup and recovery options, and transaction support to ensure data consistency and durability.
- ★★ **Real-time analysis**: MySQL supports real-time analysis through its querying capabilities, indexes, and integration with other tools for data processing and analytics.
- ★★★★ **Cost economy**: MySQL is an open-source database released under the GNU General Public License. It offers a cost-effective solution for projects with its free usage and commercial support options available.

Redis

Technology/Data Storage/In-Memory Data Structure Store

Description: Redis is an open-source in-memory data structure store. It is often used as a cache, message broker, or for real-time data processing. Redis supports various data types and provides high performance and low-latency access to data.

Consequences:

- ★★★★½ **Performance**: Redis is known for its exceptional performance due to its in-memory nature. It provides fast read and write operations, making it suitable for use cases that require high-speed data access.
- ★★ **Reliability**: Redis offers different persistence options to ensure data durability. It also supports replication and clustering for high availability and fault tolerance.
- ★★★ **Real-time analysis**: While Redis is primarily used for caching and real-time data processing, it can also support real-time analysis through its data structures and built-in features like sorted sets and stream processing.
- ★★½ **Cost economy**: Redis is an open-source database released under the BSD 3-Clause License. It provides a cost-effective solution for projects with its free usage and commercial support options available.

Remember to consider your specific project requirements, data modeling needs, and performance expectations when selecting a database for your project.

Family Cards: Performance, Scalability, Availability, Ad-hoc Analysis

Design Decision	Driver points	Bonus Points	Comments
Column-Family	10		
RDBMS	10		
Document-Oriented	11		
Key-Value Store	9.5		

Technology Cards: Performance, Reliability, Real-time analysis, Cost economy

Design Decision	Driver points	Bonus Points	Comments
MongoDB	14.5	1	It features JSON-style documents with dynamic schemas, providing simplicity and power
PostgreSQL	13	1	
MySQL	12		
Redis	12		

REFERENCES

[1] A. B. G. Contributor, "Why is project management important? the significance of effective project management," Jul 2023. [Online]. Available: https://www.capterra.com/resources/importance-of-project-management/

[2] yappkahowe, "scrumboard," Sep 2016. [Online]. Available: https://github.com/yappkahowe/scrumboard/tree/master

[3] V. Mehta, "scrum-board," Nov 2016. [Online]. Available: https://github.com/i-break-codes/scrum-board

[4] aliasaria, "scrumblr," Jun 2023. [Online]. Available: https://github.com/aliasaria/scrumblr

[5] S. Al-Saqqa, S. Sawalha, and H. AbdelNabi, "Agile software development: Methodologies and trends." *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, 2020.

[6] Dhtmlx, "How to create react gantt chart component with dhtmlxgantt," May 2023. [Online]. Available: https://dhtmlx.com/blog/create-react-gantt-chart-component-dhtmlxgantt/

[7] Mreorhan, "Scrum-Task-Management-with-ReactJS-Express-Server," 7 2022. [Online]. Available: https://github.com/mreorhan/Scrum-Task-Management-with-ReactJS-Express-Server

[8] Dhtmlx, "How to create react gantt chart component with dhtmlxgantt," May 2023. [Online]. Available: https://dhtmlx.com/blog/create-react-gantt-chart-component-dhtmlxgantt/

A Student Bio