SpaceBar ReadMe

Proposed Level of Achievement: Apollo 11

Contributors: Tan Jun Han & Wong Zhi, Chester

Web Application Link:
http://spacebar-1a6ff.web.app

GitHub Repo:
https://github.com/Chesterwongz/SpaceBar

Motivation

Due to the COVID-19 pandemic, workers who never thought of working from home have now found themselves in the new reality of remote work. This paradigm shift can be especially jarring for teams accustomed to the traditional methods of project management. Feelings of isolation and loneliness can also lead teams to become less cohesive and affect team performance.

A key aspect of a positive workplace culture is creating opportunities for employees to mingle around. However, without a physical space for face-to-face interactions, teams may grow distant and leaders might lose track of team dynamics. Thus, the looming question on the minds of project managers and leaders as they navigate through this new normal has become, "How can I still meet project goals with my team now working from home?" Although working as a team remotely can be challenging, it certainly is not impossible. Managing a team online need not be difficult at all as most (if not all) project management tasks can be done in the comfort of one's home, and connectivity tools can be used to substitute for physical interactions!

Aim

We aim to build a web application that facilitates efficient and effective collaboration despite the lack of a physical environment like an office. The web application will have easy and intuitive features that aids in the project management with Agile Methodology, encourages conversations by promoting a remote workplace culture, and provides insights by exploiting useful data analytics. This is our own take on a project management application, inspired by the success of applications like Jira, Trello and ZenHub. No more excuses for overlooking deadlines and issues!

User Stories

Our features can be broken down into 5 Epics that aim to increase efficiency and effectiveness of working remotely as a team:

[Epic A]: As a project manager, I must be able to use Agile Methodology to manage my team.

[Epic B]: As a project manager, I need tools for Business Intelligence and Analytics (BI & BA).

[Epic C]: As a project manager, I want to build a remote Workplace Culture.

[Epic S]: As a project manager, I must be sure that my project is secure.

[Epic U]: As a project manager, I want a simple, and intuitive UI/UX.

A. Agile Project Management

Agile is a project management methodology characterized by building products using short cycles of work that allow for rapid production and constant revision. Two effective and popular styles of Agile Methodology that our web application will facilitate are:

1. Kanban: A visual approach to project management where teams create physical representations of their tasks, often using sticky notes on whiteboards. Tasks are moved through predetermined stages to track progress and identify common roadblocks.
2. Scrum: A PM methodology in which a small team is led by a Scrum master, whose main job is to clear away all obstacles to completing work. Work is done in short cycles called sprints, but the team meets daily to discuss current tasks and roadblocks.

Related features:

☑	[A-01: Kanban Board |Priority: High] As a user that uses Kanban style project management, I must be able to categorise tasks into their respective stages of completion (e.g. Todo, Doing, Done) so that I have a clear idea of the progress of the project.
☑	[A-02: Task Card |Priority: High] As a user that wishes to manage a project online. I must be able to create, edit and delete tasks, so that the team will be able to keep track of the project's tasks.
☑	[A-03: Task Info Page |Priority: High] As a user that wishes to access and update all relevant information regarding a task, I must be able to view and append any such information to a task (e.g. pdf, mp4, etc), so that members will have necessary up to date information to efficiently carry out their tasks.
☐	[A-04: Task Relationships |Priority: Low] As a user that wants to know the relationship between tasks, subtasks and blocked tasks, I must be able to model their relationships, so that the team knows what needs to be done first.
☑	[A-05: Scrum Cycles |Priority: Medium] As a user that uses Scrum style project management, I must be able to add tasks into the backlog and assign them into sprints (scrum cycles), so that I have well designated sprints.
☑	[A-06: Project Customizability |Priority: Medium] As a user that is managing an Agile team, I must be able to choose my preferred Agile method i.e. either Kanban or Scrum, to fit the unique needs of my project.
☑	[A-07: Workflow Customizability |Priority: Low] As a user working on a project, I must be able to customise the stages of completion of a task by adding, arranging, and removing categories in the kanban board, to meet the unique organisational requirements of my project.
☑	[A-08: Task Card Customizability |Priority: Medium] As a user working on a project, I must be able to customise the configuration of a Task Card, to fit the unique properties of a task in my project.
☑	[A-09: Task Delegation |Priority: High] As a team leader, I must be able to assign members to a task, so that members know exactly what they need to do.

☑	[A-10: Priority Tag |Priority: Medium] As a user, I must be able to prioritise the tasks that need to be done, so that more important tasks are done first.
☐	[A-11: Task Time Estimates and Logging |Priority: Medium] As a user that wants to incorporate business analytic tools into my project, I must be able to set estimates and log time, so that I can utilise this time data for analysis.
☑	[A-12: Teams |Priority: High] As a user, I must be able to add and remove users in a project so that only the specified team members can access the project and work together.
☑	[A-13: Realtime Updates |Priority: High] As a user that is collaborating with others, my updates must show on my teammates' end immediately, and vice versa, so that the entire team stays up to date.
☐	[A-14: Notifications |Priority: Low] As a user that wants to stay up to date on relevant information, I must be notified of the changes that I am interested in, so that I will not miss important updates.
☑	[A-14: Task Archive| Priority: Low] As a user that wants to revisit or reopen previously completed tasks, I must be able to archive tasks so that I do not lose important information.

Implementation:

We use Cloud Firestore to store project information (See Cloud Firestore Database Design below). Firestore's powerful queries and snapshot listeners allow us to get real time updates for multiple users that are collaborating in the same project.

B. Business Intelligence & Analytics (BI & BA)

Integrated BI & BA tools can not only reduce the chances of staff being overutilized, and thus stretched beyond their means and possibly burn out, they also can help ensure project scheduling timelines are realistic and identify risk factors that might become obstacles to achieving those timelines.
We want to help project members to utilize data to develop resource management KPIs such as indicators of resource conflicts or on-time task completions.

Related features:

☑	[B-01: Pie Charts | Priority: High] Shows a pie chart of issues for a project/filter grouped by a specified field. This helps you see the breakdown of a set of issues, at a glance.
☑	[B-02: Cumulative Flow Diagram | Priority: High] Shows the statuses of issues over time. This helps you identify potential bottlenecks that need to be investigated.
☑	[B-03: Average Age Report | Priority: High] Shows the average age of unresolved issues for a project or filter. This helps you see whether your backlog is being kept up to date.

☑	[B-04: Time Tracking Report | Priority: Medium] Shows time tracking information on issues.
☐	[B-05: User Workload Report | Priority: Medium] Shows the time estimates for all unresolved issues assigned to a user across projects. This helps you understand the user's workload better.
☐	[B-06: Scrum Burnup Report | Priority: Low] tracks how your team did when comparing the work completed in a sprint with the work you planned to complete. It requires the Sprints feature to be enabled.
☐	[B-07: Scrum Velocity Report | Priority: Low] breaks down how much work your team was able to complete over a period of time to help you plan your next sprint. It requires the Sprints feature to be enabled.
☐	[B-08: Scrum Burndown Chart | Priority: Low] shows how much work has been completed in a sprint against what's remaining. It can help you predict and avoid scope creep. It requires the Sprints feature to be enabled.

Implementation:

We used a library called "recharts", which provided analytic components such as charts and diagrams. We chose to use this library as the components matched our theme aesthetically and documentation on how to use the library was clear. While this library was easy to use, a trade-off was customizability as compared to a library such as "d3", which had a much steeper learning curve.

B-01: We queried the tasks collection in the database which contained all the tasks in a project, with each task having a field describing its priority level. The data queried from Firestore had to then be formatted so that it matched the format required for this library.

B-02: The cumulative flow diagram shows users the history of a sprint, showing the number of "to-do", "doing" and "done" tasks for each day. This allows users to visually identify the statuses of tasks over time. To achieve this, we had to cache the statuses of the tasks everyday in a collection. The cached data is queried and formatted. We used a line chart to represent the change in statuses of tasks over time. Whenever our analytics component renders, it will check if the current date's status is stored in the cache and create a new document if it does not exist. It then updates the current document throughout the day as the statuses of tasks change throughout. When the date changes, a new document will be created and the previous document will be stored as history.

One challenge faced for this feature was to figure out how to design the database such that we can make cached data based on date. We were not able to use backend functions which would allow us to make a snapshot of the data and cache it at a specific time everyday. Therefore we had to do this logic on the frontend.

B-03: This bar chart shows the age of all of a project's tasks to a user, allowing a user to identify how long the current project is taking. Every task object has a field containing the data which it was created. The bar chart displays the difference between the current date and the date a task was created, converted to hours.

C. Workplace Culture

The Culture Code by Daniel Coyle teaches us that effective collaboration is built upon three key principles:

1. A safe environment
2. Open communication and trust
3. A sense of purpose through a shared goal and a simple way towards it.

We hope to promote such a culture despite the absence of a physical office space, by creating dedicated spaces for ideation, communication, and building rapport.

Related features:

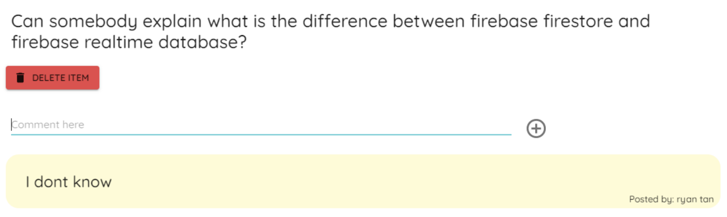
☑	[C-01: Discussion Page | Priority: High] As a team leader who wants to bring workplace culture online, I must have a space where my members can communicate openly with each other, so that members are engaged and socially connected to build rapport and promote camaraderie, as well as to prevent feelings of loneliness and abandonment
☑	[C-02: Project Info Page | Priority: Medium] As a team leader who wants to bring workplace culture online, I must be able to share the mission and vision of the team, so that members will develop a sense of purpose through the shared goal.
☑	[C-03: Roadmap Page | Priority: Low] As a team leader who wants to bring workplace culture online, I must be able to share a roadmap that defines a goal or desired outcome and includes the major steps or milestones needed to reach it, so that members have a clear understanding of the goal and how to get there.
☑	[C-04: Posts | Priority: High] As a user who wants to share my thoughts or initiate a discussion, I must be able to post on the discussion page and receive comments on my post, so that meaningful conversations can be had.

Implementation:

C-01: We used a library called “react-masonry” to make discussion posts look like post-its on a whiteboard.

C-03: Creating a roadmap required manipulation of firebase date objects and javascript date objects. Users could select dates using a Date picker package from material UI. This date had to be converted to firebase date objects to be saved in firebase. To display the duration of each epic, we calculated the number of pixels of each bar using (duration of epic in days / 2 x 365) * (Number of pixels for roadmap). The roadmap only showed months 2 years after the earliest start date. Therefore there is an assumption that a project would not last more than 2 years.

C-04: When a user clicks on a task in the drawing board, a task window will appear. A user will be able to edit the task description, delete the task, and comment on a task. As the drawing board feature is a platform for discussion and idea generation, all comments


Can somebody explain what is the difference between firebase firestore and firebase realtime database?

I dont know

Posted by: ryan tan

will be shown as a thread ordered by the most recent comment to facilitate meaningful discussion. Comments made will be updated immediately using the onSnapshot listener provided by Firebase.

Security

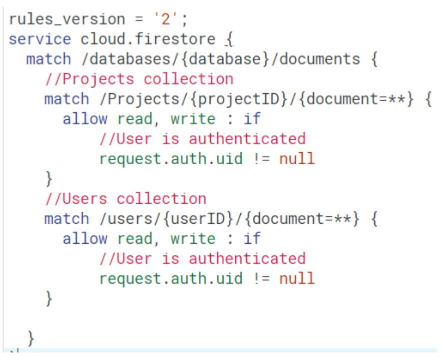
Being a project management web application, we must ensure that confidential project information does not get accessed by people not meant to access them.

Related features:

☑	[S-01: Confidentiality | Priority: High] As a user who wants my project to be confidential, no users besides the ones specified should be allowed any access to my project.
☑	[S-02: Credential storage | Priority: Medium] As a user who wants to log into SpaceBar without having to re-enter my credentials over and over again, I must be able to automatically log in using my preferred device, so that I can use SpaceBar with more ease and less unnecessary effort in the long term.
☑	[S-03: Firestore Rules | Priority: Medium] As a user who wants my project details and user information safe, reads and writes to the database should only be restricted to verified users.

Implementation:

S-01 & S-02: We designed the database such that each user document has an array containing a list of unique project document Ids that the user has access to. If a project's ID does not exist in this array, a user will not be able to view it. This ensures that users can only view projects which they have created or have been added to.



```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    //Projects collection
    match /Projects/{projectID}/{document=**} {
      allow read, write : if
          //User is authenticated
          request.auth.uid != null
    }
    //Users collection
    match /users/{userID}/{document=**} {
      allow read, write : if
          //User is authenticated
          request.auth.uid != null
    }

  }
```

S-03: Google Firestore security rules were implemented to ensure that reads and writes to the database were only done by authorised users of Spacebar, therefore adding a layer of security to protect the app from malicious users.

The Firebase rules that we wrote ensure that only authenticated users are allowed to make reads and writes to the database. While this protects the app from users who are not registered, current security rules are still not ideal as any user who is logged in has access to read and write our entire database. Tighter rules will be implemented in the future so that rules are specific to not only the base collections but also sub-collections.

UI/UX

Cluttered and unintuitive UI/UX can make project management a turnoff. We strive to avoid this being the case

☑	[U-01: Simplicity | Priority: High] As a user busy with work, the web application must be easy to understand, and display all the necessary information without it being too cluttered, so that I do not have to waste time navigating through an unintuitive UI.

☑	[U-02: Drag and Drop Tasks | Priority: HIgh] As a user working on the board, updating tasks must be easy, hassle free, and intuitive so that I can be more efficient with my time.
☐	[U-03: Mobile Compatibility | Priority: Low] As a user that wants to view my tasks on my mobile device, I must be able to use the web app on the mobile screen, so that I can access and update project information at my convenience.
☐	[U-04: Filtered View | Priority: Medium] As a user that wants to view only necessary information, I must be able to specify what information I want to be filtered out, so that I can have a less cluttered workspace.

Implementation:

U-02: We used “react-beautiful-dnd” to achieve drag-and-drop functionality. The library required the use of arrays splicing to order the lists of draggable objects, which was problematic due to Cloud Firestore’s array and ordering limitations. See Cloud Firestore Database Design section below on how we designed our backend to overcome these challenges.

Difference From Existing solutions

Every Other:

- Our discussion page provides a place for discussion and ideation that stands out from other existing solutions (see wireframe). This space for conversation and ideation encourages users to share suggestions and promote creative thinking, and is something that existing solutions currently lack.

Jira:

- We found that while Jira is a powerful tool for project management, its UI is too cluttered, making it difficult for new users to get used to the platform. We want to make a cleaner UI/UX for our users while maintaining only the necessary features required for smaller teams.

Trello:

- Trello provides an extremely customizable board. While this approach to project management has its merits, we want to provide more structure by building our website with a project management framework called Agile Methodology in mind. This makes setting up projects easy and intuitive.

Zenhub:

- Zenhub integrates with Github to target software developers. We target any person or team in need of project organisation.

Project Specification

Development Plan

The project has been divided into 4 phases that align with orbital milestones 1, 2, 3 and splashdown.

Phases 1 and 2 are the main development phases for this project as they entail the implementation of the core functionalities of the webapp as described briefly below:

Phase 1 (10th to 31st May):

- Kanban board for the project's todo, doing, and done lists. (A-01, A-02, U-02)
- Backend firebase implementation (A-12).
- Login and Authentication (S-01).
- Basic discussion page (C-01).
- Team member invitation and project exclusivity (A-11)

Phase 2 (1st to 28th June):

- Task card detailed info (A-03, A-08, A-09, A-10).
- Scrum Cycles and choice between scrum vs non-scrum PM methodology (A-04, A-05).
- Important Project Analytics (B-01, B-02, B-03).
- Working discussion page and posts (C-01, C-04).

Phase 3 is an extension to the main development cycle of this project. It entails the development of additional features that aim to enrich user experience, as well as important user testing. The plan is described briefly below:

Phase 3 (29th June to 26th July):

- More extensive user testing and squashing of current bugs/issues.
- Kanban board customizability (A-06).
- Account settings and notifications (A-13).
- Task Archive (A-14)
- Discussion page posts interactable comments (upvotes, reply, edit/delete) (C-01, C-04).
- Project Info page + Roadmap (C-02, C-03).

Phase 4 is for future extensions (i.e. features that will only be implemented beyond Orbital). It entails the implementation of lowest priority features that serve to touch up and finish off the webapp:

Phase 4 (27th July onwards):

- Task relationships (A-04).
- UI/UX refinement and additional features such as filtered view (U-01, U-04).
- Additional BI & BA tools (B-04, B-05).
- Extension BI & BA tools (B-06, B-07, B-08).
- Mobile compatibility (U-03).

Technology Stack

Our webapp will be developed with the following technologies:

1. React
   We chose to use React as our framework for a number of reasons. React has a fast learning curve, making it easy for beginners to pick up. It also allows for greater speed in web applications as only selective components are updated and re-rendered on the DOM when their state changes, rather than needing to re-render unnecessary components. React also has a wide community support and integrates well with libraries such as Google Firebase and Material UI.
2. Firebase

   We chose to use Firebase because of its security and clear documentation provided by Google, making it easy to learn. Firebase is a No-SQL database, which provides much flexibility in database structure as compared to an SQL database such as mySQL. With that being said, extra precaution is needed to prevent NoSQL injections as well as ensure deletion and insertion anomalies are not committed due to the inability to normalize data in No-SQL databases.

   Firebase also allows for fast deployment and hosting. Firebase Auth also allows us to implement different modes of logins (Google, Github, Facebook etc) while taking care of security, making it simple to complete the login and signup features of the app.

3. Material UI

   We decided to use Material UI for our UI/UX as it is a simple yet powerful React library. It provides customizable and aesthetic components which follow the principles of material design, a design system used by Google, which we believe ties well with our aim to make our interface minimalistic and clean.

4. GitHub

   We use GitHub for version control to track files and coordinate work within the team as we collaboratively develop source code during software development. GitHub’s speed, data integrity, and support for distributed, non-linear workflows make it the perfect tool for this.

Cloud Firestore Database Design

Cloud Firestore is a NoSQL, document-oriented database. Unlike a SQL database, there are no tables or rows. Instead, you store data in documents, which are organized into collections. Each document contains a set of key-value pairs. Cloud Firestore is optimized for storing large collections of small documents. All documents must be stored in collections. Documents can contain subcollections and nested objects, both of which can include primitive fields like strings or complex objects like lists.

This gives us a few options in modelling our Cloud Firestore database to suit the needs of our web application.

1. Nested data in documents

   We can nest complex objects like arrays or maps within documents.

   - Advantages: If you have simple, fixed lists of data that you want to keep within your documents, this is easy to set up and streamlines your data structure.
   - Limitations: This isn't as scalable as other options, especially if your data expands over time. With larger or growing lists, the document also grows, which can lead to slower document retrieval times.
2. Root-level collections

   Create collections at the root level of your database to organize disparate data sets.

   - Advantages: Root-level collections are good for many-to-many relationships and provide powerful querying within each collection.
   - Limitations: Getting data that is naturally hierarchical might become increasingly complex as your database grows.
3. Subcollections within documents

   You can create collections within documents when you have data that might expand over time.

   - Advantages: As your lists grow, the size of the parent document doesn't change. You also get full query capabilities on subcollections, and you can issue collection group queries across subcollections.
   - Limitations: You can't easily delete subcollections.

With the above considerations in mind, we decided to model our hierarchical database as such:

- users-collection
    - user-document-1
      displayName: “Chester Wong”
      createdAt: June 17, 2021 at 3:00:08 PM UTC+8
      projectRef: [project1, project2]
    - user-document-2
      displayName: "Ryan Tan"
      createdAt: June 17, 2021 at 4:20:42 PM UTC+8
      projectRef: [project2, project3]
- projects-collection
    - project-document-1
        - tasks-subcollection
            - task-document-1
              title: "Complete milestone 2 README"
              description: ...
            - task-document-2
              title: “Record milestone 2 video"
              description: ...
        - kanban-board-subcollection (for kanban projects)
            - list-document-1
              title: "Todo"
              items: [task2, task1]
            - list-document-2
              title: "Doing"
              items: [ ]
            - list-document-3
              title: "Done"
              items: [ ]
        - scrum-board-subcollection (for scrum projects)
            - backlog-document
              items: [task2, task1]
                - board-subcollection
            - sprint-document-1
              items: [task3, task4]
                - board-subcollection
            - sprint-document-2
              items: [task3, task4]
                - board-subcollection
        - discussion-posts-subcollection
            - post-document-1
              title: "Should zombies have human rights?"
              createdBy: “user1”
                - comments-subcollection
        - analytics-subcollection
            - 210617-document
              cummulativeFlow: { list1: 2, list2: 0, list3: 0 }

Firstly, we decided to create two root-level collections that have a many-to-many relationship. One for users, and another for projects (one user can have many projects / one project can have many users). Each project document contains its project information and multiple subcollections.

Secondly, since Firestore arranges its collections, documents, and fields in alphanumeric order by default, the only way to preserve ordering required for tasks in our todo lists is by keeping track of them in an array nested in each todo list document. However, arrays are a problematic data structure for multi-user environments. Bad things happen when multiple clients are trying to update or delete array elements at specific indexes. For example, consider the case where three clients attempt to modify an array simultaneously:

1. Initial data: [{name: 'foo', counter: 1}, {name: 'bar', counter: 1}, {name: 'baz', counter: 1}]
2. Client A moves the "bar" record to position 0
3. Client B attempts to delete the "bar" record at position 1
4. Client C tries to update bar's counter with something like data[1].counter++

The end result is that record 'foo' is deleted, and the counter for record 'baz' is incremented, while the now incorrect 'bar' sits at position 0. None of the operations worked quite as intended.

Hence to achieve our drag-and-drop functionality we had to first retrieve the arrays, splice and reorder them at the client-side, then replace the entire array in the database. The nested arrays contain reference keys which are used to retrieve information of the tasks stored in their own respective documents in the “tasks” subcollection. Storing tasks in a subcollection allows us to do simpler and more powerful queries as compared to nesting task objects in the todo lists.

However, using subcollections to maintain a project’s data makes it difficult to delete projects. For now, deleting a project simply removes all users’ access to it, enforced by our database security rules. We plan to overcome this limitation by utilising Google Cloud Functions to safely delete subcollections.

Separation of Concern Design Principle (SoC)

In computer science, separation of concerns (SoC) is a design principle for separating a computer program into distinct sections such that each section addresses a separate concern. Modular programs that follow the SoC principle have many benefits such as:

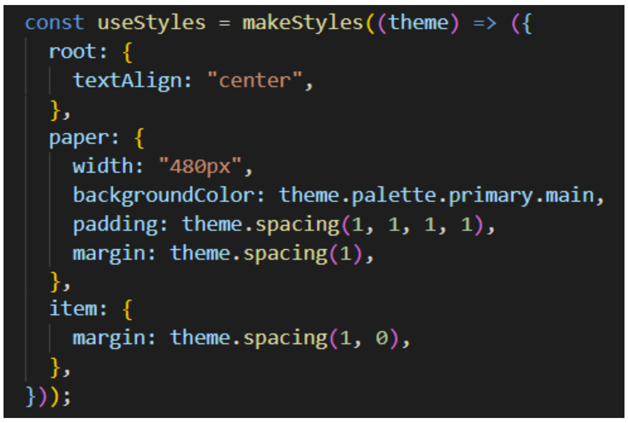
1. Clear responsibilities and concerns for each of the main modules
2. Tests are easier to write and maintain
3. Easier refactoring and locating bugs due to decoupling
4. Predictable and unidirectional data flow
5. Ready for advance requirements such as code sharing between web and native projects.

Layered designs are an embodiment of separation of concerns (e.g. presentation/view layer, data access layer, persistence layer, controller layer). For example, development of web pages and websites typically use HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS). Layers are clear and defined as HTML is mainly used for semantic content and organization of webpage content (Model Layer), CSS is used for definition of content presentation style (View Layer), and JS defines the behaviour of the content and how the content interacts with the user (Controller Layer).

However, because we use functional React and Javascript XML (JSX) language, the layers become less distinct. JSX mixes Semantic Content (Model Layer) with Behavior (Controller Layer), and our usage of React components and libraries further adds Presentation (View Layer) into the mix. Presentational components responsible for rendering JSX, and container components responsible for all the functionality, are no longer distinctly separated.

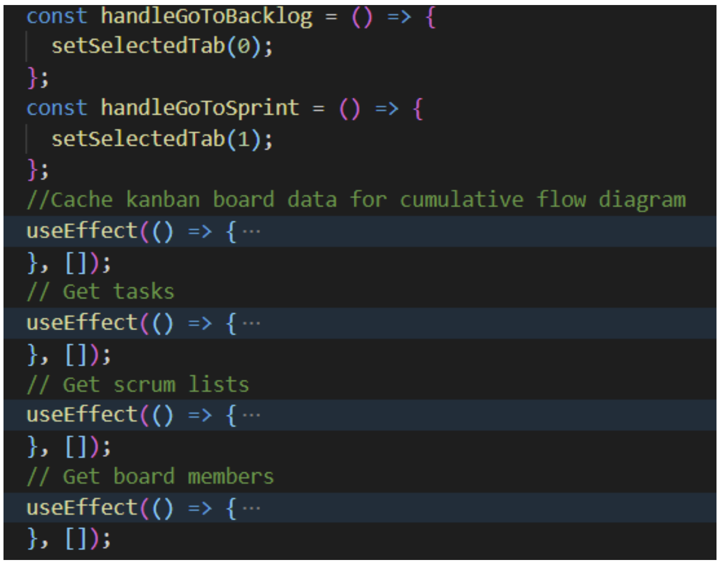
Therefore, we self-enforce SoC into our architecture with React hooks, as well as separating each page and feature in our web application into their own individual components.

- Firstly, we use the useStyles() custom hook with MUI's makeStyles() to internally apply CSS to our components. The hook enforces a better separation between the View and Model layers as compared to using inline-CSS.



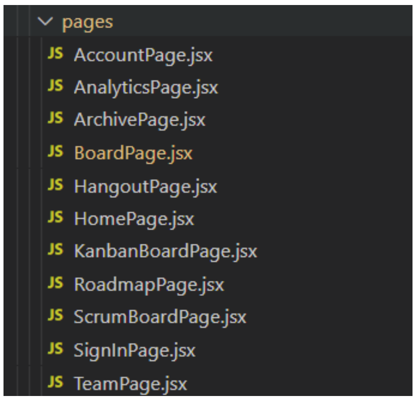
```
const useStyles = makeStyles((theme) => ({
  root: {
    textAlign: "center",
  },
  paper: {
    width: "480px",
    backgroundColor: theme.palette.primary.main,
    padding: theme.spacing(1, 1, 1, 1),
    margin: theme.spacing(1),
  },
  item: {
    margin: theme.spacing(1, 0),
  },
}));
```

- Next, useEffect hooks and JS functions separate the Behaviour layer. Accessing and updating the Firestore database is also done here.

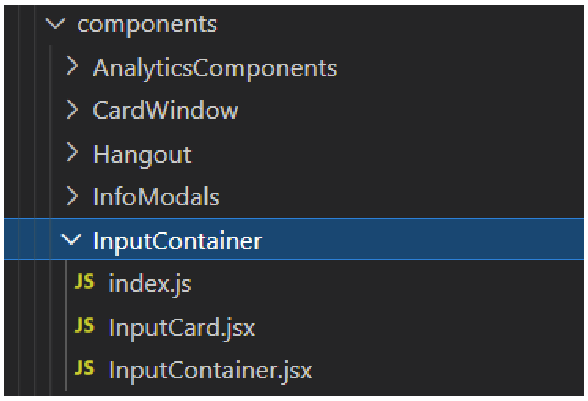


```
const handleGoToBacklog = () => {
  setSelectedTab(0);
};
const handleGoToSprint = () => {
  setSelectedTab(1);
};
//Cache kanban board data for cumulative flow diagram
useEffect(() => {…
}, []);
// Get tasks
useEffect(() => {…
}, []);
// Get scrum lists
useEffect(() => {…
}, []);
// Get board members
useEffect(() => {…
}, []);
```

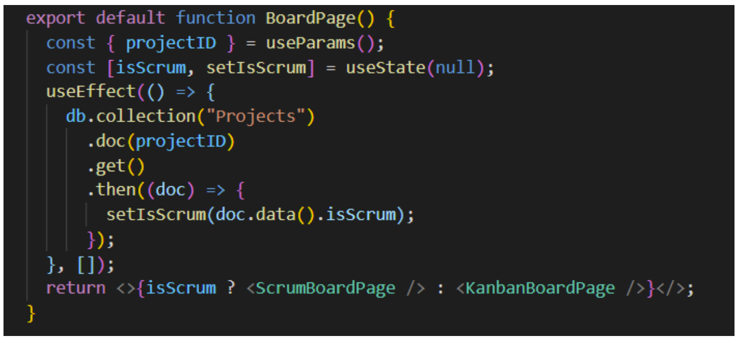
- Objects that compose each feature are plainly separated into sets of simple and reusable components. Each page component is only concerned about rendering itself and the reusable components that compose it.



```
pages
  AccountPage.jsx
  AnalyticsPage.jsx
  ArchivePage.jsx
  BoardPage.jsx
  HangoutPage.jsx
  HomePage.jsx
  KanbanBoardPage.jsx
  RoadmapPage.jsx
  ScrumBoardPage.jsx
  SignInPage.jsx
  TeamPage.jsx
```



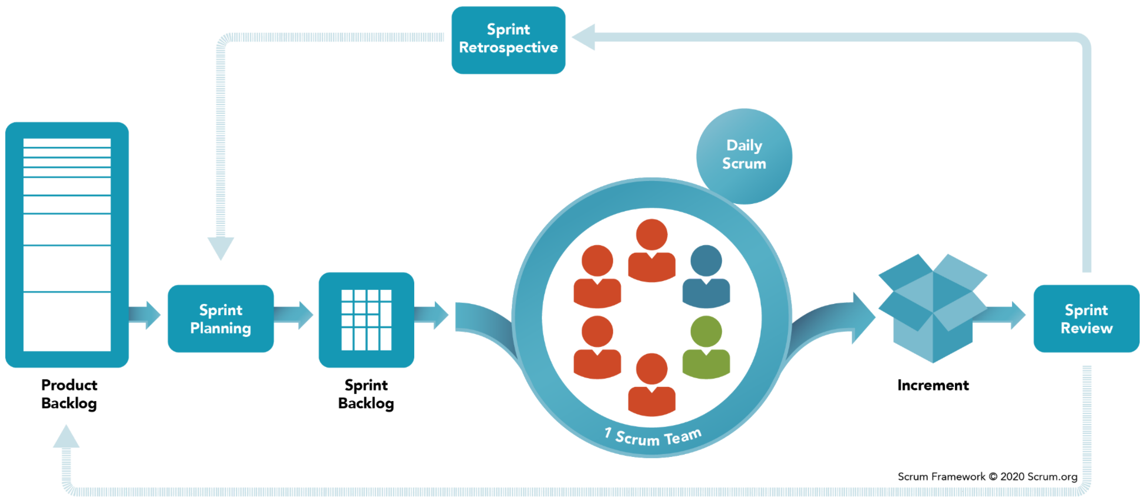
```
components
  > AnalyticsComponents
  > CardWindow
  > Hangout
  > InfoModals
  InputContainer
    index.js
    InputCard.jsx
    InputContainer.jsx
```



```
export default function BoardPage() {
  const { projectID } = useParams();
  const [isScrum, setIsScrum] = useState(null);
  useEffect(() => {
    db.collection("Projects")
      .doc(projectID)
      .get()
      .then((doc) => {
        setIsScrum(doc.data().isScrum);
      });
  }, []);
  return <>{isScrum ? <ScrumBoardPage /> : <KanbanBoardPage />}</>;
}
```

Project Management

We incorporate "SCRUM" techniques into our development process with the help of Jira, an Agile project management app that offers powerful tools for our project management needs. From agile boards, backlogs, roadmaps, reports, to integrations and add-ons we can plan, track, and manage all your agile software development projects from a single tool.



The features of our web application are categorised into 5 Epics of 34 user stories total. The user stories/features are further broken down into smaller sub-goals which are queued into the backlog. Every Saturday, we meet to share what we have learnt and how we should proceed in the following weekly sprint. Backlog issues are added to the sprint based on their priority and are addressed accordingly during the sprint.

The overarching timeline and implementation plan is outlined above in the development plan.

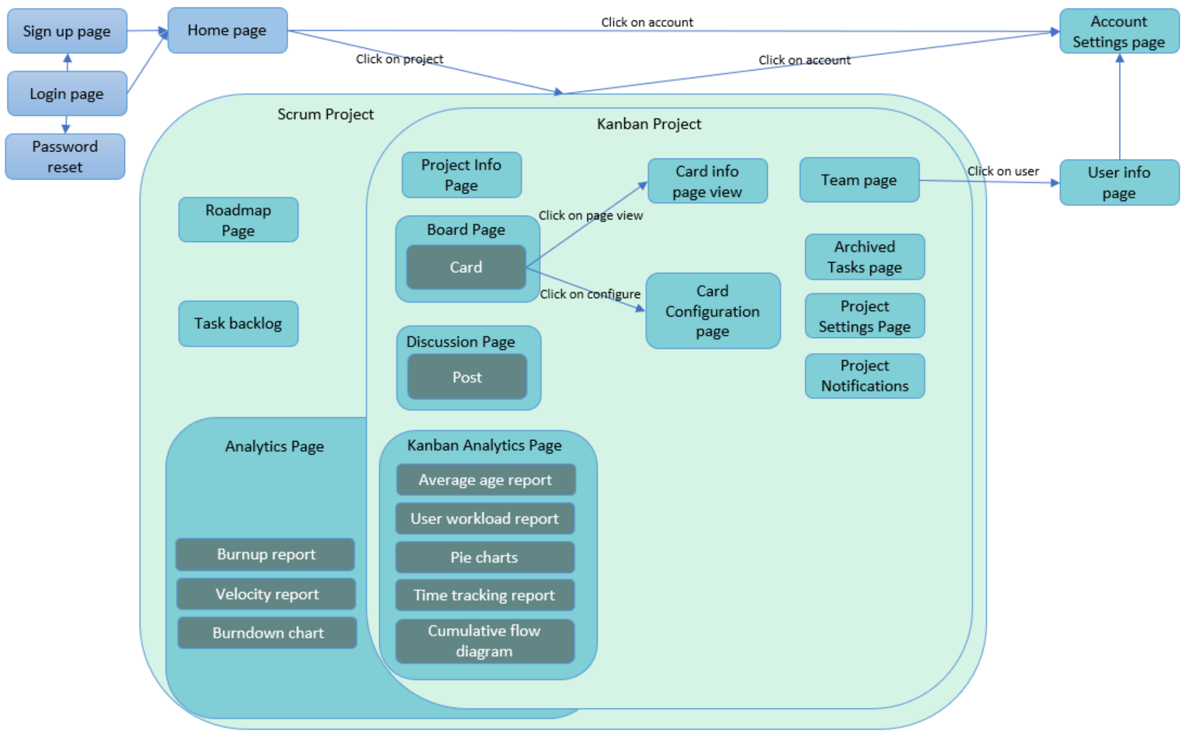
Tasks are assigned and done individually on separate GitHub branches and pull requests are made once a feature has been completed. For example, Ryan works on BI/BA features on the "analytics" branch and submits a pull request when he has finished implementing a feature. If we need help, we will create sub branches to assist each other. Having separate branches for features and clearly demarcated work assigned during sprint planning, any merge conflicts are quickly and easily resolved.

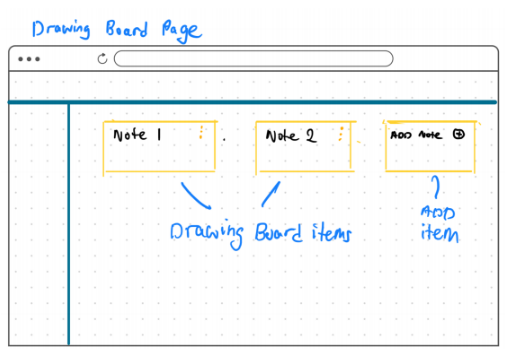
Problems, bugs or issues that inevitably cropped up during development are examined and triaged by priority and added to our Jira backlog to be added to the current or subsequent sprints.

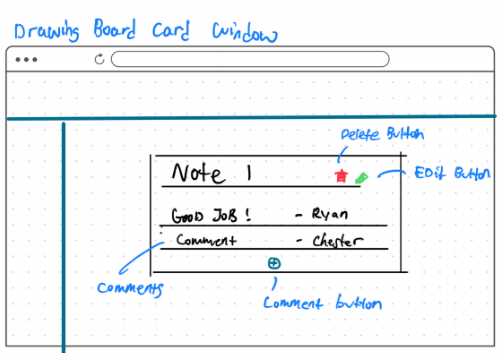
Overall, our use of the Agile Scrum methodology with Jira allowed for an easy, seamless and efficient development experience.

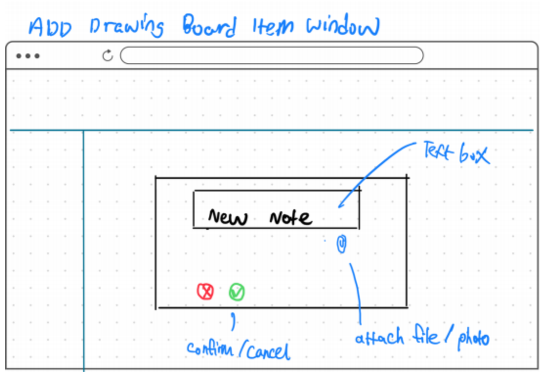
Wireframe

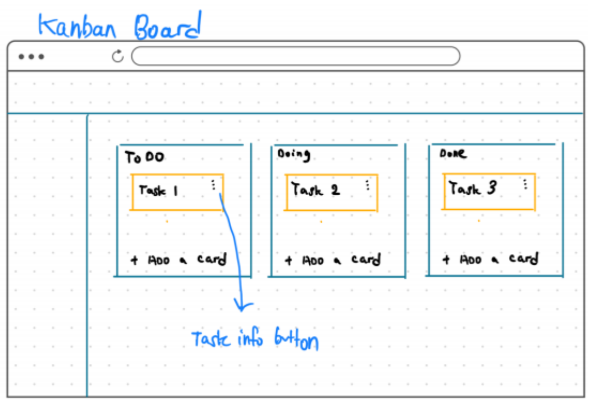
Flow Diagram:

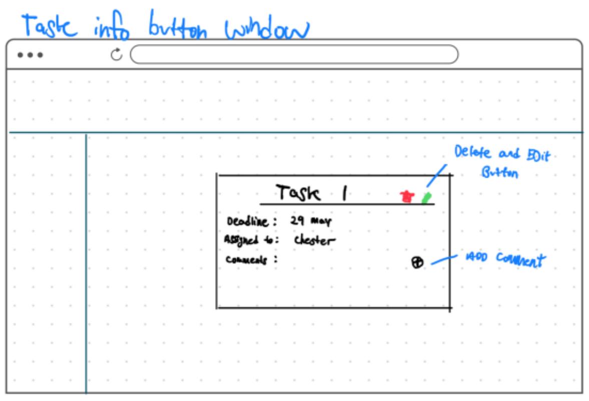


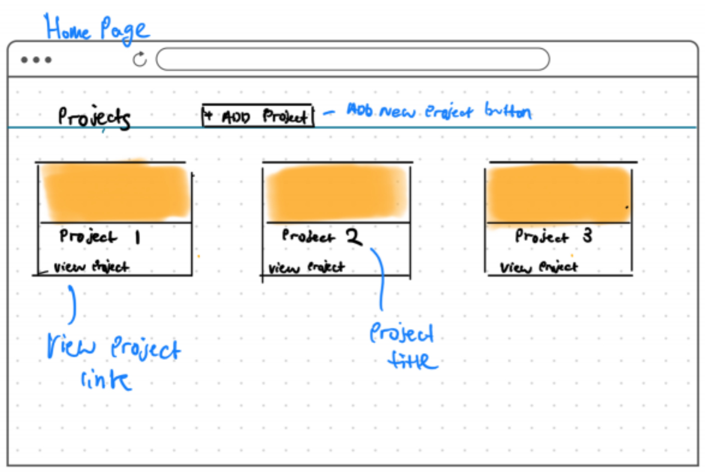

Drawing Board Page
Note 1
Note 2
Add Note ⊕
Drawing Board items
Add item


Drawing Board Card Window
Delete Button
Note 1
Edit Button
Good Job ! - Ryan
Comment - Chester
Comments
Comment button


Add Drawing Board Item Window
Text box
New Note
confirm / cancel
attach file / photo

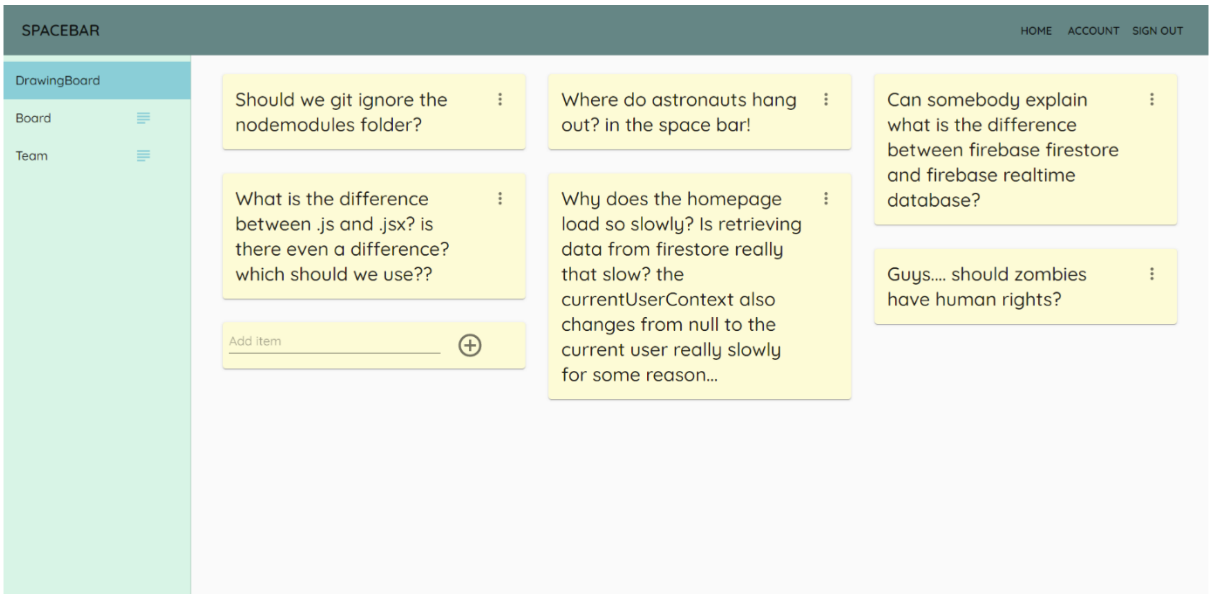

Kanban Board
To Do
Task 1
+ Add a card
Doing
Task 2
+ Add a card
Done
Task 3
+ Add a card
Task info button


Task info button window
Delete and Edit Button
Task 1
Deadline : 29 May
Assigned to : Chester
Comments :
Add Comment


Home Page
Projects
+ Add Project
Add new Project button
Project 1
view project
Project 2
view project
Project 3
view project
View project link
Project title

Prototype

Discussion page:


SPACEBAR

HOME ACCOUNT SIGN OUT

DrawingBoard

Board

Team

Should we git ignore the nodemodules folder?

Where do astronauts hang out? in the space bar!

Can somebody explain what is the difference between firebase firestore and firebase realtime database?

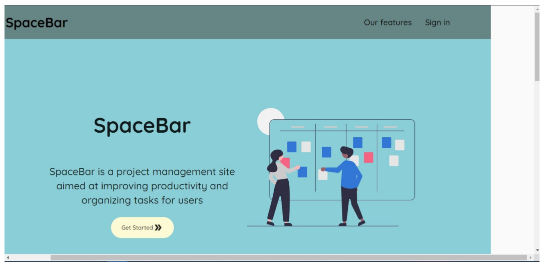
What is the difference between .js and .jsx? is there even a difference? which should we use??

Why does the homepage load so slowly? Is retrieving data from firestore really that slow? the currentUserContext also changes from null to the current user really slowly for some reason...

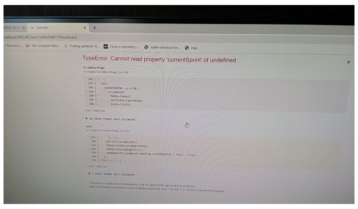
Guys.... should zombies have human rights?

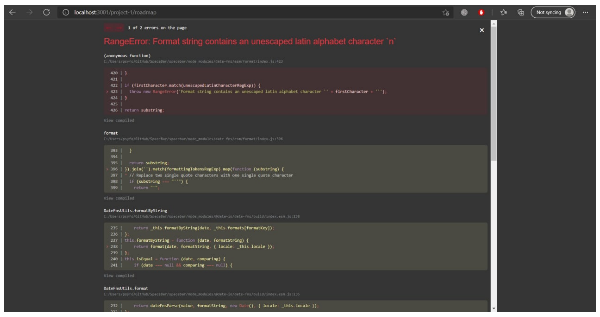
Add item

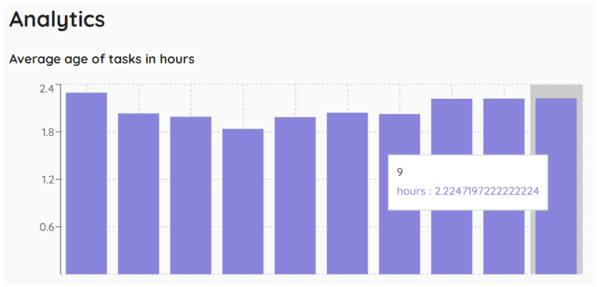
Bug Report

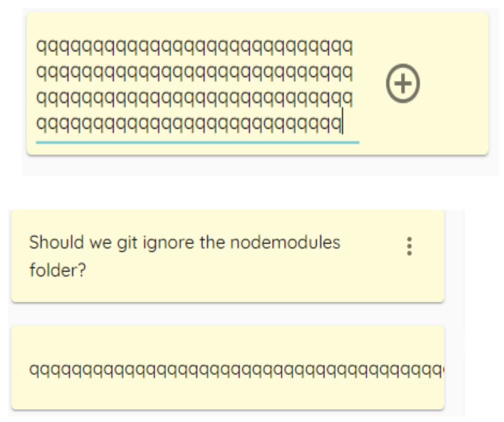
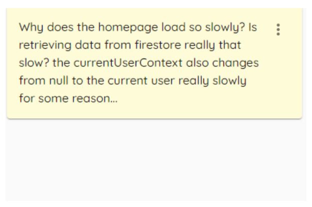
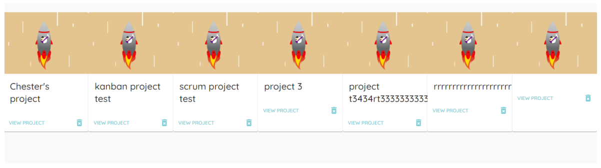
Severity	Description	Steps to reproduce	Expected behaviour
High (Fixed)	When screen width is below 1500 pixels, a horizontal scroll bar appears and an empty space forms on the left of the page.  SpaceBar Our features Sign in SpaceBar SpaceBar is a project management site aimed at improving productivity and organizing tasks for users Get Started	1. Lower screen width below 1500 pixels.	No horizontal scroll bar should appear and divs should be full width of screen always.
High (Fixed)	Sprint names are not consecutive when adding sprints.	1. Create a new Scrum project and go to Board Page 2. Click on "Go to backlog" 3. Add 2 new sprints.(This creates Sprint 1 and Sprint 2 automatically) 4. Click on	On step 5, after a user adds a new sprint, the new sprint added should be called sprint 3. Instead, a user has 2 sprints called Sprint 2.

		"Complete sprint" for sprint 1 5. Click on Add Sprint. Instead of Sprint 3 being created, Sprint 2 is created again.	

High (Fixed)	Viewing a new project causes the webapp to crash.
	
	localhost:3000/4EZxum1bJMcPWB71fXns/board
	TypeError: Cannot read property 'currentSprint' of undefined
	ScrumboardPage
	src/pages/ScrumboardPage.jsx:199
	`) : (`
	`<div>`
	`{selectedTab === 0 && (`
	`<Scrumboard`
	`tasks={tasks}`
	`sprintIDs={sprintIds}`
	`lists={lists}`
	View compiled
	▶ 18 stack frames were collapsed.
	next
	src/pages/ScrumboardPage.jsx:162
	`}, {});`
	`setLists(scrumLists);`
	`setSprintIds(scrumSprintIds);`
	`setSprintsLoading(false);`
	`setNoSprint(scrumLists.backlog.currentSprint ? false : true);`
	`});`
	`return () => {`
	View compiled
	▶ 3 stack frames were collapsed.
	This screen is visible only in development. It will not appear if the app crashes in production.
	Open your browser's developer console to further inspect this error. Click the 'X' or hit ESC to dismiss this message.
		1. Click on Sign in with google and sign in with a new google account
		2. Create a new project
		3. Click on "View project"
			Board page should render without error.

High (Fixed)	Clicking on add sprint causes the webapp to crash
	
	localhost:3001/project-1/roadmap
	1 of 2 errors on the page
	RangeError: Format string contains an unescaped latin alphabet character `n`
	(anonymous function)
	C:/Users/psyfo/GitHub/SpaceBar/spacebar/node_modules/date-fns/esm/format/index.js:423
	`}`
	`if (firstCharacter.match(unescapedLatinCharacterRegExp)) {`
	`throw new RangeError('Format string contains an unescaped latin alphabet character `' + firstCharacter + '`');`
	`}`
	`return substring;`
	View compiled
	format
	C:/Users/psyfo/GitHub/SpaceBar/spacebar/node_modules/date-fns/esm/format/index.js:396
	`}`
	`return substring;`
	`}).join('').match(formattingTokensRegExp).map(function (substring) {`
	`// Replace two single quote characters with one single quote character`
	`if (substring === "''") {`
	`return "'";`
	View compiled
	DateFnsUtils.formatByString
	C:/Users/psyfo/GitHub/SpaceBar/spacebar/node_modules/@date-io/date-fns/build/index.esm.js:238
	`return _this.formatByString(date, _this.formats[formatKey]);`
	`};`
	`this.formatByString = function (date, formatString) {`
	`return format(date, formatString, { locale: _this.locale });`
	`};`
	`this.isEqual = function (date, comparing) {`
	`if (date === null && comparing === null) {`
	View compiled
	DateFnsUtils.format
	C:/Users/psyfo/GitHub/SpaceBar/spacebar/node_modules/@date-io/date-fns/build/index.esm.js:235
	`return dateFnsParse(value, formatString, new Date(), { locale: _this.locale });`
		1. Go to a scrum project.
		2. Add a sprint.
			Adding a sprint should work without error.

Medium (Fixed)	Task age bar chart does not show task title  	1. Go to barchart 2. Hover over bar	Bars should show task title
Low (Fixed)	When typing a long title for a post, not the whole title could be seen and the menu icon is cut off. This only happens when a user types a long string of characters without spaces.Iin the unlikely event that this happens, a user would not be able to delete the card.	1. Go to Hangouts page 2. Type a long string without any spaces and click on the Add icon.	Long text should overflow to the next line as shown in diagram below.

	 qqqqqqqqqqqqqqqqqqqqqqqqqqqq qqqqqqqqqqqqqqqqqqqqqqqqqqqq qqqqqqqqqqqqqqqqqqqqqqqqqqqq qqqqqqqqqqqqqqqqqqqqqqqqqqq Should we git ignore the nodemodules folder? qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq		 Why does the homepage load so slowly? Is retrieving data from firestore really that slow? the currentUserContext also changes from null to the current user really slowly for some reason...
Low (Fixed)	Having more than 5 projects with long project titles, the css breaks.  Chester's project VIEW PROJECT kanban project test VIEW PROJECT scrum project test VIEW PROJECT project 3 VIEW PROJECT project t3434rt3333333333 VIEW PROJECT rrrrrrrrrrrrrrrrrrrr VIEW PROJECT VIEW PROJECT	1. Go to "Your Projects" 2. Add 7 new projects with long names	There should be only 3 cards in one line. If there are more than 3 cards, it should go to the next line

Testing

We conduct two types of tests to ensure the quality, usability, and functionality of the app. These tests are important in determining whether each version of the app is production standard.

1. Functionality testing
   Functionality testing checks if the initial build works as per its design. It often covers link testing, form validation, cookie testing, HTML and CSS validation, and database connection checkup.
2. Usability testing
   Usability testing is a method of testing the functionality of a website, app, or other digital product by observing real users as they attempt to complete tasks on it.The goal of usability testing is to reveal areas of confusion and uncover opportunities to improve the overall user experience.

Firstly, for functionality testing, we designed scripted tests internally where detailed test scripts were written for each feature. Scripted testing ensures that tests are consistent and reproducible. This allows us to easily identify any bugs when new updates are pushed to the website as tests can be simply repeated to identify if the changes caused unexpected results.

Next, for usability testing, we selected users in need of project management to test our web application, as they would most likely be familiar or require some sort of project management software. These users include other Apollo and above orbital teams, and project leaders in NUS Fintech Society. A google form was given to these participants, as well as the link to the live website. The form contained tasks to be completed, such as creating an account, as well as a feedback form to highlight any bugs or room for improvement. Exploratory user testing also allows us to capture bugs that did not surface during scripted testing.

Scripted Tests

Test Condition	Procedure	Expected result
User is able to successfully sign in with google account	1. Click on "Sign in with google" button 2. Follow instructions on popup	1. Google pop-up appears on step 1 2. User gets redirected to home page on step 2
User is able to create a new account using email after entering appropriate details	1. Enter valid name, email, password and confirm password on signup form 2. Click on Sign up button	1. Pop-up appears on step 2 informing users that a verification code has been sent to their email 2. Verification email shows up in inbox
User is able to sign in to account via email after verifying account	1. Enter email and password on sign in page 2. Click on sign in button	1. User is redirected to home page on step 2
User gets clear error messages when trying to create a new account with wrong credentials	1. Enter the following data and click on sign up button: Name: User123 Email: invalidemail Password: Spacebar123 ConfirmPassword: Spacebar123 2. Enter the following data and click on sign up button: Name: User123 Email: user@yahoo.com Password: Spacebar123 ConfirmPassword: Spacebar 3. Enter the following data and click on sign up button: Name: User123 Email: user@yahoo.com Password: 123 ConfirmPassword: 123 4. Click on the sign up button when one of the fields is left empty	1. Error message saying that email is badly formatted appears on step 1 2. Error message saying that passwords do not match appears on step 4 3. Error message saying that passwords should be at least 6 characters 4. Error message informing user to fill in all required fields
User gets error message when signing in with wrong credentials	1. Enter a valid email and an invalid password and click on sign in button 2. Enter an invalid email and valid password and click on sign in button 3. Enter on valid email which	1. Error popup showing the following: “The password is invalid or the user does not have a password.” 2. Error popup showing the following: “ There is no user record corresponding to this

	has not yet been verified and valid password before clicking on sign in button	identifier. The user may have been deleted.” 3. Error popup showing the following: “ Verify your email before signing in”
User can reset account password	1. Click on Forgot your password link 2. Enter the email address where the reset link will be sent to 3. Click on reset password button 4. Follow instructions in email	1. Password reset email appears in inbox on step 3 2. Enter new password in step 4 3. Sign in successfully with new password
User can create a new Kanban project	1. Click on “Add a project” button 2. Enter project title and click on “Add Kanban Project” button	1. New project card appears on home page on step 2
User can create a new Scrum project	3. Click on “Add a project” button 4. Enter project title and click on “Add Scrum Project” button	2. New project card appears on home page on step 2
User can create an epic on the roadmap page	1. Click on “Add Epic” 2. Enter a title for the epic and the start and end dates 3. Click on “Add epic” button	1. Calendar popup appears on step 2 when calendar icon is clicked 2. Calendar and progress bar appears on step 3
User can edit epic details	1. Click on epic to be edited 2. Click on the title and edit the words 3. Enter a text in the “Add a description” text field 4. Click on the green edit icons for the start and end dates and change the dates by clicking on the calendar icon 5. Click on the green button to confirm the changes made to the start or end date 6. Click on the delete epic icon	1. On step 1, title should become an editable text field, and a green edit icon will appear to confirm edit. Changes should be updated on the page automatically. 2. On step 3, a green icon to confirm the edit should appear after clicking on the text field. Description should be saved to the epic. 3. DateFields should appear on step 4, where dates can be edited on clicking on the calendar icon.
Epics should be ordered with the earliest start date at the top	1. Create 2 epics with different start dates	1. The epic with the later start date should appear below the epic with the earlier start date

User can create a post on Hangout page	1. Enter a text in the text field and click on icon	1. New post appears on page on step 1
User can edit a post on Hangout page and make comments	1. Click on menu icon on the post to be edited 2. Click on title and change the text 3. Enter a comment and click on add icon 4. Click on “Delete item” button	1. Popup window appears on step 1 2. On step 2, title becomes an editable text field, and a green edit icon will appear to confirm edit. Changes should be updated on the page automatically. 3. On step 3, comment appears with name of user on comment bar 4. On step 4, post is deleted
User can invite team member to project on team page	1. Enter email of team member in the text bar 2. Click on “Invite Member” button	1. On step 2, a notification appears with the text: “Please wait for the user to accept your invitation”
User gets the relevant error messages when entering wrong credentials when adding team members to the project	1. Enter 123 and click on “Invite Member” button 2. Click on “Invite Member” button with an empty text field 3.	1. On step 1, notification appears stating that 123 is not a valid user 2. On step 2, a notification appears showing “Please enter a value”
User can accept an invitation to a project	1. Log in to account which has been invited to a project 2. Click on join button	1. On step 2, project card appears on home page for the user who has been added to the project 2. For the user inviting the new member, a new member card appears on the team page
User can reject an invitation to a project	1. Log in to account which has been invited to a project 2. Click on Ignore button	1. On step 2, invitation card disappears from home page
User can view profile page and edit username	1. Click on ‘Account’ link on the nav bar 2. Click on edit profile button 3. Change the user name and click on “Save changes”	1. User is redirected to profile page with name, email and edit profile button rendered 2. On step 2, window appears to edit profile 3. On step 3, new username is saved
User can create a new sprint and	1. Click on add sprint button 2. Click on “Add a card button”	1. New sprint appears on Board page after step 1

add tasks to sprint	under the newly created sprint 3. Enter a title and click on “Add card”	2. Textfield appears on step 2 3. New card added to sprint on step 3
User can add new tasks to backlog and move tasks to sprint	1. Click on “Add a card” button under the backlog list 2. Enter a title and click on “Add Card” 3. Click on “Add sprint” 4. Drag newly created card to the new sprint	1. On step 1, textfield appears 2. New card appears in backlog on step 2 3. New sprint appears on step 3 4. Card is dragged from backlog to sprint in step 4
User can start a new sprint and complete a sprint	1. Click on Board on the menu bar on the left 2. Click on “Go to backlog” button at the bottom right of the page 3. Click on add sprint 4. Add a card to the sprint 5. Click on start sprint 6. Click on “Go to sprint” button at the bottom right of the page 7. Go back to backlog page and click on “Complete sprint” button	1. On step 1, a page is rendered informing a user to go and start a sprint 2. On step 2, user is directed to backlog page 3. On step 3, new sprint is added 4. On step 4, new card is added to sprint and “Start Sprint” button can now be clicked 5. On step 5, “Start sprint” button becomes a “Complete Sprint” button. 6. On step 6, boards are rendered, with the task inside the ToDO list. 7. On step 7, completed sprint disappears from the page
User can assign a task to a team member	1. Create a new task in the backlog 2. Click on the task card 3. Click on the text which says “Unassigned” 4. Click on the name of the team member to assign the task to	1. On step 1, task appears in backlog list 2. On step 2, a window pops up 3. On step 3, a list will appear, showing the current members in a project 4. On step 4, the team member is assigned to the task and is reflected on the task card
User can change the priority of a task	1. Create a new task in the backlog 2. Click on the task card 3. Click on the text which says “ Medium” 4. Click on the priority level to be assigned	1. On step 1, task appears in backlog list 2. On step 2, a window pops up 3. On step 3, a list appear showing different levels of priority: Highest, High, Lowest, Low 4. New priority level is reflected in the window

User can edit task title, add task description and comment on task	1. Create a new task in the backlog 2. Click on the task card 3. Click on title and change the text 4. Enter a comment and click on add icon 5. Click on “Add a description” and add a text	1. On step 1, task appears in backlog list 2. On step 2, a window pops up 3. On step 3, title becomes an editable text field, and a green edit icon will appear to confirm edit. Changes made after confirmation are saved. 4. On step 4, comment appears with name of user on comment bar 5. On step 5, the description box becomes an editable text field, and a green edit icon will appear to confirm the edit. Changes made after confirmation are saved.
User can archive tasks in board page	1. Go to board page 2. Click on add sprint 3. Create a task in the backlog list 4. Click on the archive icon on the top right of the card 5. Create a task in the sprint 6. Click on the archive icon on the top right of the card	1. On steps 3 and 5, new tasks appear in their respective lists 2. On steps 4 and 6, tasks are archived from their respective lists and appear in the task archive, with their status changed to “Archived”
User can permanently delete archived tasks	1. Go to archive page 2. Click on the delete forever icon on the top right of the card	1. On step 2, tasks will be permanently deleted
User can unarchive tasks in the archive	1. Go to archive page 2. Click on the unarchive icon on the top right of the card	1. For scrum projects, unarchived tasks will appear in the backlog with status set as “Todo”. 2. For Kanban projects, unarchived tasks will appear in the Todo list of the board page.

User Testing

Users were given general tasks to complete and encouraged to explore the website to capture new bugs as well as receive feedback to improve UI/UX.

Link to form:
https://docs.google.com/forms/d/e/1FAIpQLSesfx9kK-6BiHIoMQX35N-4Gp-wtDOup_4uxu3H0itkCfYzxg/viewform?usp=sf_link

Task	Issues/Feedback	Response
Create an account and sign in using either email or the "Sign in with google option"	Have invalid dialog pop out even though can login.	Fixed
Create a new Scrum project	Having more than 5 projects with long project titles, the css breaks	Fixed
View your account page	No bug but its very slow, adding a loading logo would at least let the user know bcs initially i thought there was a bug, but ended up its just slow. When i edit the name to be very long the name goes out of the card.	Fixed
Explore the features on the Roadmap page	Not mobile friendly. Table is not responsive when window width is changed (goes out of the window) Not really sure if this function is useful? I am assuming there's nothing else we can do than to see the epic duration	Postponed
Explore the features on the Roadmap page	My mouse cant scroll horizontally (only vertical) so abit inconvient to scroll. Maybe can display only the necessary months	Postponed
Explore the features on the Hangouts Page	When title is too long, cannot delete or comment on ticket. css destroyed when width of window is small (mobile), when there is at least one ticket with long title	Fixed
Explore the features on the Hangouts Page	Will be better if i can change the color of the note	Will not fix
Explore the features on the Board Page	Sprint should have a confirmation dialog before deleting, ux principle The board columns are not resizable depending on window width, gets out of the way and user is limited to creating columns (or view) depending on window's width. When there are things on my backlog but no sprint, the homepage shows "No sprint", although not a bug, but the ux is quite weird	Postponed
Explore the features on the Board Page	I have no idea how to get the board to work. Created a sprint(?) but its not working. Side note, the backlog page reminds me of monday.board	Will not fix
Explore the features on the Board Page	When i click on project, it goes to board and then shows a blank screen after	More info needed

Explore the features on the Team Page	I have a long name which i edited from account, the card shown on team page is broken	Fixed
Archive a task from the board page and explore the features on the Archive page.	if i unarchive from something that's done, it goes back to "todo" as default, not sure if this is intended but ux wise can be improved if it goes to original place.	Will not fix