Project Proposal: Taskmaster

Team Placeholder
James Dang, John Dao, Jake Edwards, Edward Gauld, Jaydon Tse
z5209597, z5258962, z5114769, z5246767, z5214494

University of New South Wales

COMP3900: Computer Science / Information Technology Project

13 June 2021

Background

The problem

As projects and collaboration amongst team members scale in a digital world, the demand for task-management software grows. Naturally, the role of software has grown and now is being relied upon to manage individual workloads, track the progression of work and distribute tasks amongst collaborating team members.

The Task Master project is a task management solution that allows teams within the workplace to better manage teams of greater scale. It combines the personalisation of individual task managers, with the collaboration benefits of project management tools to create a better task solution for group workflows. Through its task management and assignment system, individual workload determiners and team collaboration tools, it will become a disruptor to current workplace task management techniques and bring forward the potential for a more efficient workflow.

Current Market Solutions and their Drawbacks

There are currently a few task-based management applications currently available on the market today, with the most notable being: Todoist ("Todoist: The to do list to organize work & life", 2021), Google's taskboard ("TasksBoard | Desktop app for Google Tasks", 2021) and Microsoft's To Do ("Microsoft To Do", 2021).

However, common major drawbacks exist within these established task management apps that the Task Master project is capitalising upon.

1. All of these solutions are not focused upon group and project task management being more geared towards individual workloads, preventing these tools from being more commonly utilised in the work industry. Although possibilities do exist to share projects within these apps, they are not the focal point of these products and thus, do not perform the role of being a group task manager to a high level of depth.

2. These solutions do not provide any features for quantifying workloads. This is a severely limiting factor to these applications as it discourages the even and fair distribution of work, and therefore, decreases the efficiency of said team.
3. Google's taskboard and Microsoft's To Do both attempt to capitalise and rely on their developed ecosystems to draw in users from their existing userbase. Though this may be a positive thing for individual users, when working within enterprises, such ecosystems must be implemented at scale and may not be realistic and/or require great resources to implement.

As an extension onto these current market solutions, there are also many collaboration tools geared more towards enterprise teams, with two of the most popular being: Jira ("Jira | Issue & Project Tracking Software | Atlassian", 2021) and Active Collab ("Project Management Software", 2021).

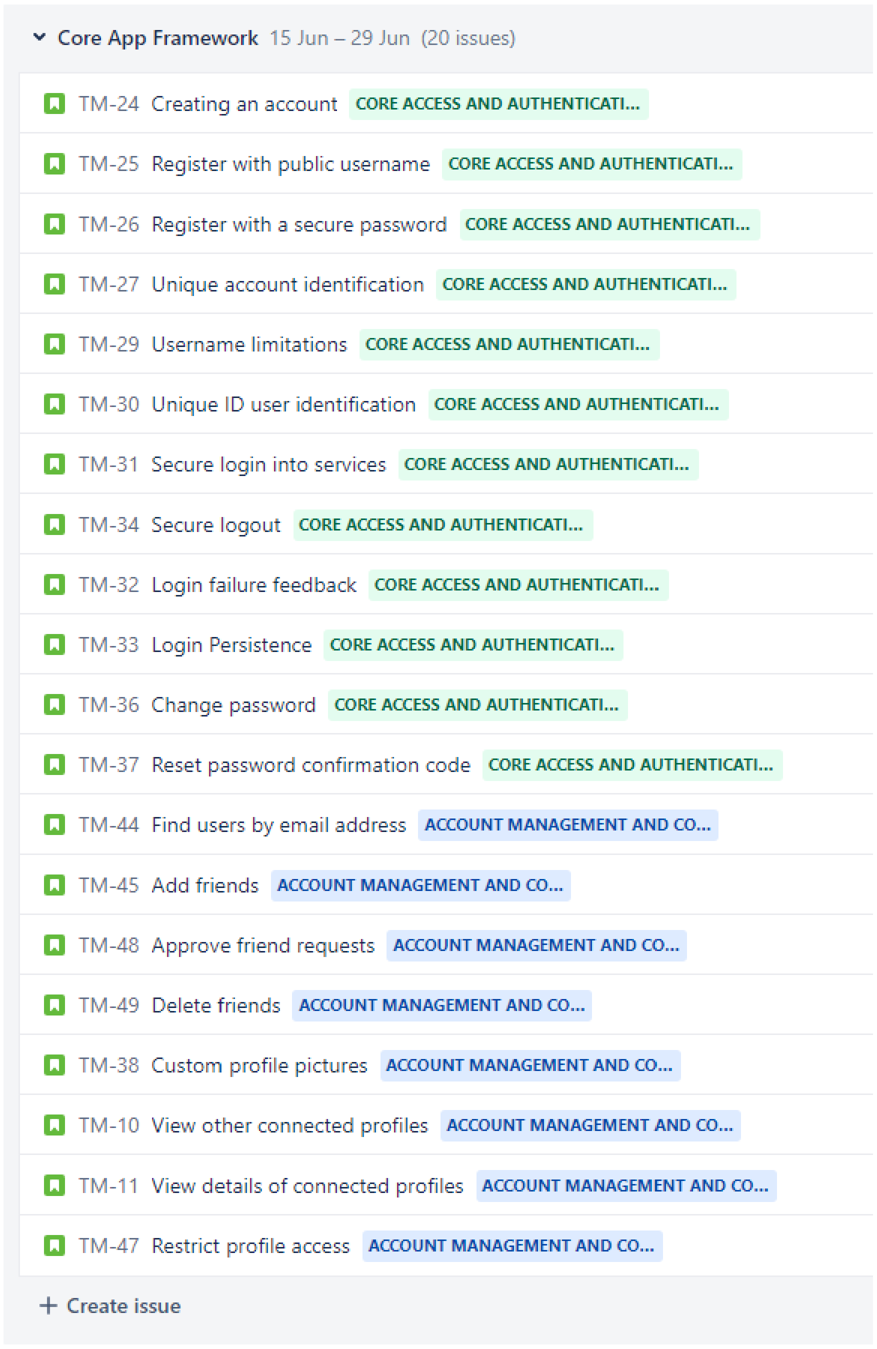
Although these tools are geared towards enterprise team project management, there are still significant drawbacks that the Task Master project capitalises upon.

1. Both of these solutions are focused on project management and not individual task distribution. Although issues can be assigned to specific team members, there are no specified tools to allow users to manage their own allocated workloads, resulting in these tools serving roles as being task hubs rather than task managers. This means that they do not operate with the same individual effectiveness as would be provided by direct task allocation, requiring other methods to provide the same effect, such as team meetings or relying on individual team members to manage tasks.
2. The above issue also extends into individual task management due to their lack of personal task boards and day-to-day task management tools. These solutions are unable to operate with the same individual focus as prior mentioned solutions, preventing them from being used more commonly by the personal customer for individual workloads.
3. There are no features that can properly quantify the total workload of individuals. This results in the need to hold meetings to ensure proper workloads are distributed evenly amongst the team. Although both solutions have quantifiers for the difficulty of certain task issues (story points), this only correlates with the project itself and does not

help promote workload distribution transparency and efficiency as much as direct indicators would.

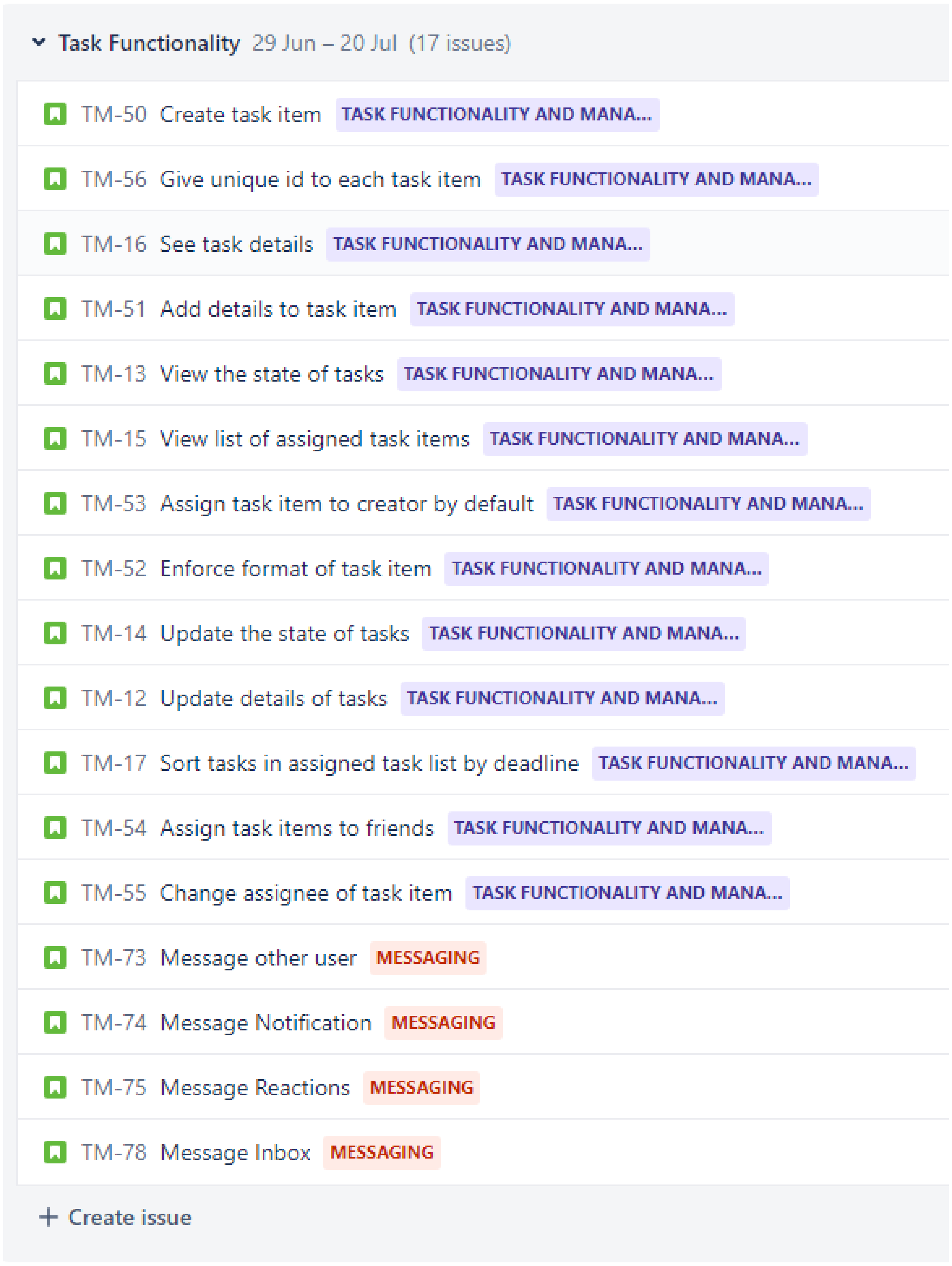
User stories & Sprints

User Stories



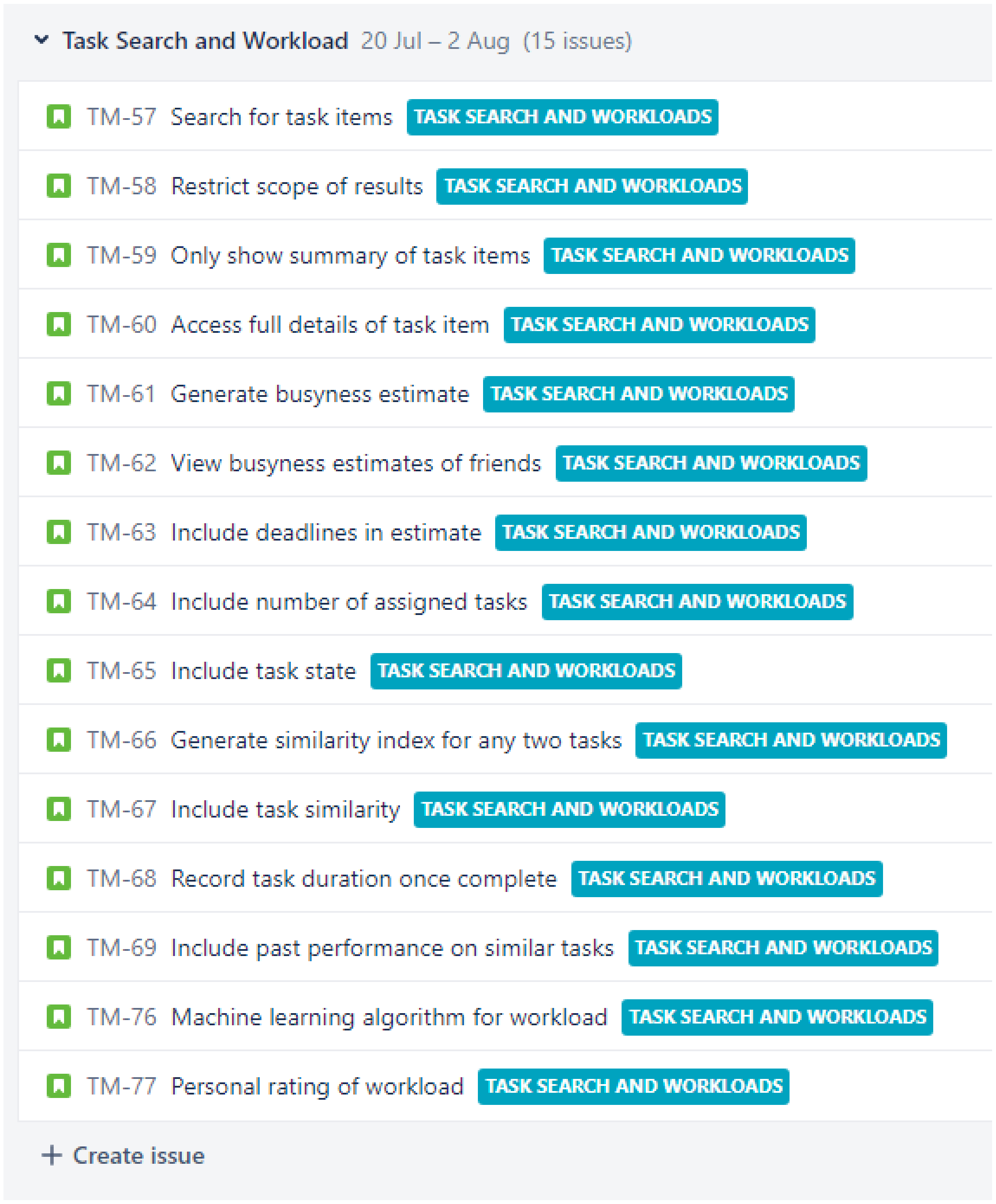
Core App Framework 15 Jun – 29 Jun (20 issues)

TM-24 Creating an account CORE ACCESS AND AUTHENTICATI...
TM-25 Register with public username CORE ACCESS AND AUTHENTICATI...
TM-26 Register with a secure password CORE ACCESS AND AUTHENTICATI...
TM-27 Unique account identification CORE ACCESS AND AUTHENTICATI...
TM-29 Username limitations CORE ACCESS AND AUTHENTICATI...
TM-30 Unique ID user identification CORE ACCESS AND AUTHENTICATI...
TM-31 Secure login into services CORE ACCESS AND AUTHENTICATI...
TM-34 Secure logout CORE ACCESS AND AUTHENTICATI...
TM-32 Login failure feedback CORE ACCESS AND AUTHENTICATI...
TM-33 Login Persistence CORE ACCESS AND AUTHENTICATI...
TM-36 Change password CORE ACCESS AND AUTHENTICATI...
TM-37 Reset password confirmation code CORE ACCESS AND AUTHENTICATI...
TM-44 Find users by email address ACCOUNT MANAGEMENT AND CO...
TM-45 Add friends ACCOUNT MANAGEMENT AND CO...
TM-48 Approve friend requests ACCOUNT MANAGEMENT AND CO...
TM-49 Delete friends ACCOUNT MANAGEMENT AND CO...
TM-38 Custom profile pictures ACCOUNT MANAGEMENT AND CO...
TM-10 View other connected profiles ACCOUNT MANAGEMENT AND CO...
TM-11 View details of connected profiles ACCOUNT MANAGEMENT AND CO...
TM-47 Restrict profile access ACCOUNT MANAGEMENT AND CO...



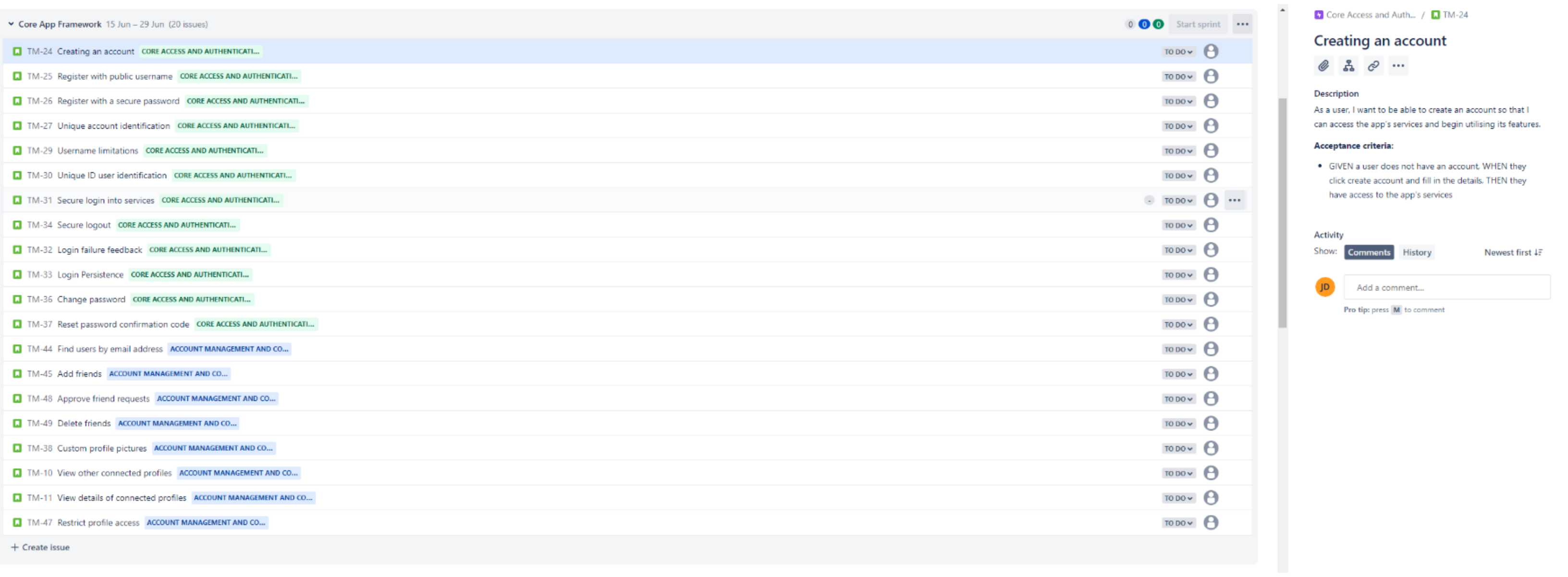
Task Functionality 29 Jun – 20 Jul (17 issues)

TM-50	Create task item	TASK FUNCTIONALITY AND MANA...
TM-56	Give unique id to each task item	TASK FUNCTIONALITY AND MANA...
TM-16	See task details	TASK FUNCTIONALITY AND MANA...
TM-51	Add details to task item	TASK FUNCTIONALITY AND MANA...
TM-13	View the state of tasks	TASK FUNCTIONALITY AND MANA...
TM-15	View list of assigned task items	TASK FUNCTIONALITY AND MANA...
TM-53	Assign task item to creator by default	TASK FUNCTIONALITY AND MANA...
TM-52	Enforce format of task item	TASK FUNCTIONALITY AND MANA...
TM-14	Update the state of tasks	TASK FUNCTIONALITY AND MANA...
TM-12	Update details of tasks	TASK FUNCTIONALITY AND MANA...
TM-17	Sort tasks in assigned task list by deadline	TASK FUNCTIONALITY AND MANA...
TM-54	Assign task items to friends	TASK FUNCTIONALITY AND MANA...
TM-55	Change assignee of task item	TASK FUNCTIONALITY AND MANA...
TM-73	Message other user	MESSAGING
TM-74	Message Notification	MESSAGING
TM-75	Message Reactions	MESSAGING
TM-78	Message Inbox	MESSAGING



Task Search and Workload 20 Jul – 2 Aug (15 issues)

TM-57	Search for task items	TASK SEARCH AND WORKLOADS
TM-58	Restrict scope of results	TASK SEARCH AND WORKLOADS
TM-59	Only show summary of task items	TASK SEARCH AND WORKLOADS
TM-60	Access full details of task item	TASK SEARCH AND WORKLOADS
TM-61	Generate busyness estimate	TASK SEARCH AND WORKLOADS
TM-62	View busyness estimates of friends	TASK SEARCH AND WORKLOADS
TM-63	Include deadlines in estimate	TASK SEARCH AND WORKLOADS
TM-64	Include number of assigned tasks	TASK SEARCH AND WORKLOADS
TM-65	Include task state	TASK SEARCH AND WORKLOADS
TM-66	Generate similarity index for any two tasks	TASK SEARCH AND WORKLOADS
TM-67	Include task similarity	TASK SEARCH AND WORKLOADS
TM-68	Record task duration once complete	TASK SEARCH AND WORKLOADS
TM-69	Include past performance on similar tasks	TASK SEARCH AND WORKLOADS
TM-76	Machine learning algorithm for workload	TASK SEARCH AND WORKLOADS
TM-77	Personal rating of workload	TASK SEARCH AND WORKLOADS



Core App Framework 15 Jun – 29 Jun (20 issues)

TM-24	Creating an account	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-25	Register with public username	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-26	Register with a secure password	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-27	Unique account identification	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-29	Username limitations	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-30	Unique ID user identification	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-31	Secure login into services	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-34	Secure logout	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-32	Login failure feedback	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-33	Login Persistence	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-36	Change password	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-37	Reset password confirmation code	CORE ACCESS AND AUTHENTICATI...	TO DO
TM-44	Find users by email address	ACCOUNT MANAGEMENT AND CO...	TO DO
TM-45	Add friends	ACCOUNT MANAGEMENT AND CO...	TO DO
TM-48	Approve friend requests	ACCOUNT MANAGEMENT AND CO...	TO DO
TM-49	Delete friends	ACCOUNT MANAGEMENT AND CO...	TO DO
TM-38	Custom profile pictures	ACCOUNT MANAGEMENT AND CO...	TO DO
TM-10	View other connected profiles	ACCOUNT MANAGEMENT AND CO...	TO DO
TM-11	View details of connected profiles	ACCOUNT MANAGEMENT AND CO...	TO DO
TM-47	Restrict profile access	ACCOUNT MANAGEMENT AND CO...	TO DO

+ Create issue

Core Access and Auth... / TM-24

Creating an account

Description

As a user, I want to be able to create an account so that I can access the app's services and begin utilising its features.

Acceptance criteria:

- GIVEN a user does not have an account. WHEN they click create account and fill in the details. THEN they have access to the app's services

Activity

Show: Comments History Newest first

Add a comment...

Pro tip: press M to comment

Sprints

Our project plan will be broken down in to four sprints. The first sprint is focused on completing the project proposal and planning. Despite our proposal being due week 4, this is a short sprint as we want to start the development process as soon as we can. The following three sprints are structed as to finish on the project demonstrations in week 5, 8 and 10. The aim is to complete a major piece of functionality to show at each demonstration. Each sprint builds on what is developed from the previous sprint, and so the ordering of schedule is crucial. We are open however, to allocating more or less time for a certain sprint, depending on how difficult we find the implementation.

Week 1-3 Proposal, Planning, and Setup

Start Date: 01/06/2021, End Date: 15/06/2021

The first sprint focuses on completing the proposal that is due in week 4. We want to finish this as quickly as possible so that we can begin development. Additionally, during this sprint we are writing the APIs for our program so that the front-end and back-end developers will have an understanding of the format their code needs to be created in. This sprint also involves the planning and distribution of tasks at a high level, including who will be responsible for front end or back end development, and the general timeline for our project moving forward. Lastly, we will begin the construction of the basic code framework to prepare us to get started on building app specific functionality. The programming languages and data bases to be used are discussed in the 'System Architecture' section below.

Week 3-5 Core App Framework

Start Date: 15/06/2021, End Date: 29/06/2021

The second sprint's focus is the implementation of the core framework for the app, including the log-in process, profile viewing, and connection with other users. It is essential that we complete these features before moving on to features that depend on these such as task functionality and messaging. During this split, the proposal will be due on Monday Week 4 . If needed, changes will be made to the proposal from the information we gather in this sprint. This sprint will end with Demo A to be presented in week 5. It is possible that we will need more than 2 weeks to complete all requirements, however we will focus on producing a presentable piece of work at the bare minimum for this demonstration.

Week 5-8 Task Functionality & Messaging

Start Date: 29/06/2021, End Date: 20/07/2021

The focus of this split is implementing the task board and all its related functionalities, including assigning tasks to connected task masters. Additionally, we will be implementing a messaging service, which is a novel idea for a task management system. This sprint will include submitting Retrospective A in week 7, and demo B to be completed in week 8. Completing this split will allow us to have almost all the required functionalities to be completed for the demonstration. This split has a longer time allocated of three weeks, due to the flexibility week during week 6, effectively giving us two weeks to complete our objectives.

Week 8-10 Task Search and Workload

Start Date: 20/07/2021, End Date: 02/08/2021

The final split will involve the implementation of the extra features related to tasks, including searching and a users workload. The workload algorithm will be generated using a technically novel machine learning approach. Given extra time, we will look back on our completed code and produce more novel features that we have brainstormed, including adding task privacy, importance, and a calendar to visually place them. This sprint will additionally involve a review of our code to ensure that we are submitting a high level of work to be assessed. During this split we will have Retrospective B due in week 9, and our completed code paired with out final report due on the Monday of week 10.

Technical depth, scale, report formatting

System Architecture

User types and their interaction with the system

The system will only have two main user types: managers and team members. The typical interactions these users will have with the system include:

Managers

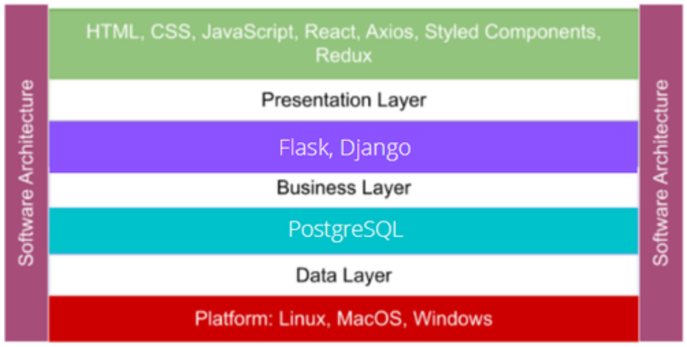
- Overlooking the progress of tasks
- Allocating tasks to team members
- Assigning deadlines to tasks
- Viewing the workload of team members

Team Members

- Updating the progress of a task
- Assigning subtasks or main tasks to other team members
- Connecting with other team members
- Checking the progress of tasks assigned to other team members

The Stack

The full-stack solution for the web application will be React for the frontend, Flask for the backend and MongoDB/Firebase for the database.



Frontend

According to "SitePoint – Learn HTML, CSS, JavaScript, PHP, Ruby & Responsive Design" (2021), React.js is the most popular frontend framework, which renders HTML, CSS and JavaScript. Its widespread favouritism and support has proven its strong development process since its existence in 2011.

With React, we will be integrating the following libraries:

- Redux - A popular state management library that allows for global states to be updated and read from any component. While this can be done with React itself, it is more maintainable with Redux and has shown to be the most popular method. You can check out its documentation here ("Redux - A predictable state container for JavaScript apps. | Redux", 2021).
- Styled-components - Component styling library that doesn't require an external file, like a CSS file, to change the style of a HTML element. Styled components also allow arguments to be passed through, changing styling based on a particular argument. For example, passing a *hovered=true* argument can change the colour of a button. Documentation can be found at styled-components, (2021).
- Axios - Will send API requests to the backend. Compared to React/JavaScript's native fetch, Axios will automatically convert JSON data from a HTML response, which would be essential since the backend will mostly send data in JSON.

In production, we will host the React application onto AWS Amplify, which provides continuous integration and deployment.

Backend

We will be using a Django application, which uses Python, to perform CRUD (Create, Read, Update, Delete) operations on the database. This will be a monolithic server since TaskMaster does not offer a lot of microservices and is managed by a small team of developers.

To perform these CRUD operations, we will use the REST API design, where the frontend (React) will send HTTP requests to the backend and invoke CRUD.

There are several reasons why Django is a good choice for our app and team:

- It is a well established, highly popular framework for developing backend APIs, with a lot of documentation and support
- It uses python, which all our team has experience in.
- Most of the CRUD operations will require user authentication, and Django has a built in user authentication system, which has proven to be secure and effective. Features we will use include user profiles, account verification services, secure password storage, password resetting, token authentication, and the ability to restrict access to certain endpoints in just a couple lines of code. We can also use it to store user login persistence in the frontend, so when a user exits an application, they can reopen it again and remain logged in.
- Excellent support for testing
- It integrates well with our choice of database and SQL databases in general, abstracting away most of the querying logic required
- We can use Django Rest Framework, which provides support for REST apis, including:
    - Generic classes, which recognise that many CRUD operations across different components share a similar structure. It takes in the format of these components and generates CRUD methods based on this. This reduces the amount of code and simplifies development.

- Routing
- Pagination (limiting the amount of results returned on each call)
- A simple interface for structuring HTTP responses
- Support for mapping data to JSON objects, as well as data validation

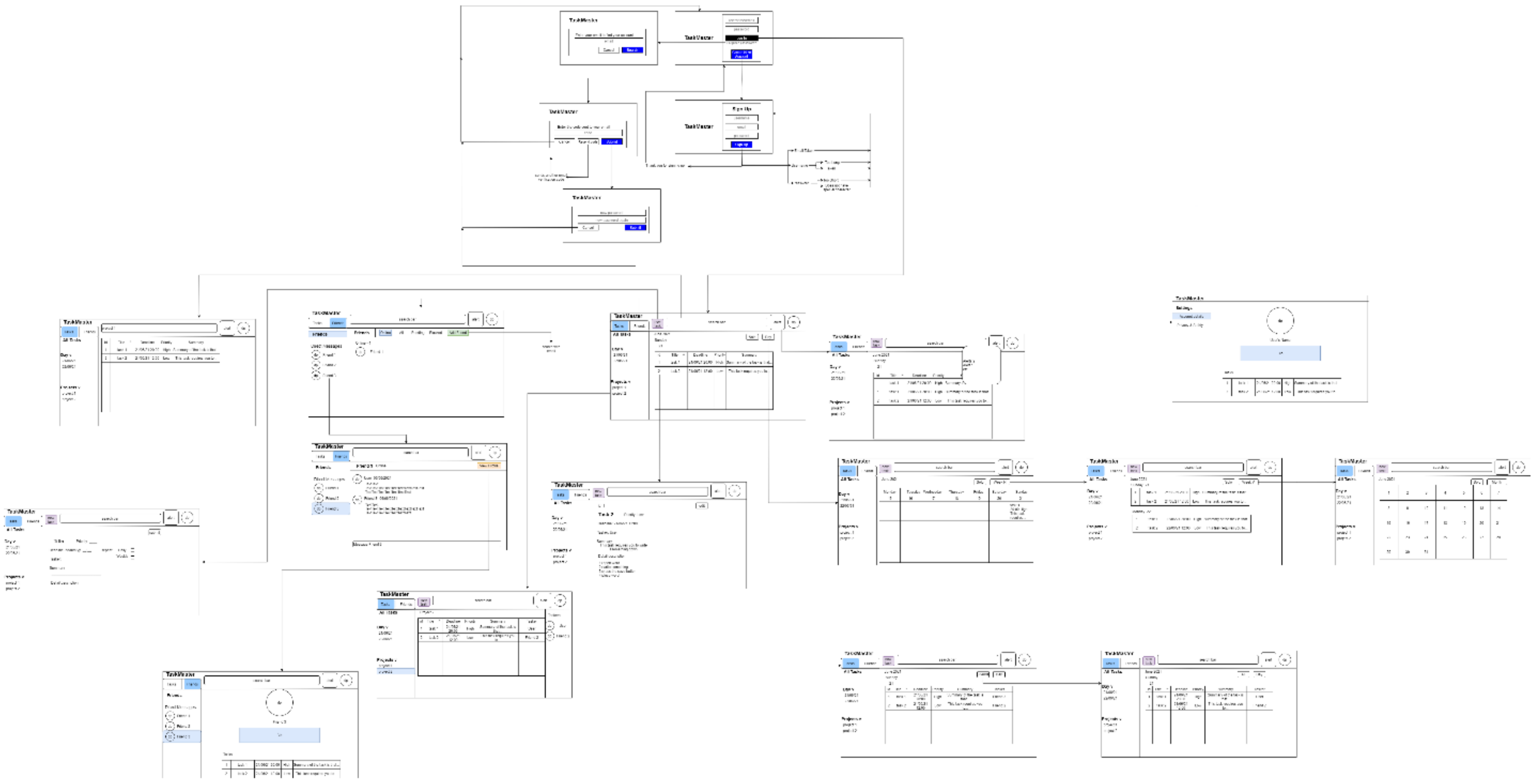
In production, we will host the Django application on AWS Elastic Beanstalk.

Database

PostgreSQL will be the only database instance to be used by TaskMaster. Being a SQL database means the developer team can provide a pre-defined structure so that valid data is being stored. This would be useful for error handling and preventing data redundancy. It is also more effective in querying, compared to noSQL databases, so that retrieval and updates of data would be more performant.

In production, we will host it on AWS, which allows the database to be in the cloud.

Storyboard



References

Jira | Issue & Project Tracking Software | Atlassian. Atlassian. (2021). Retrieved 13 June 2021, from https://www.atlassian.com/software/jira.

Microsoft To Do. To Do. (2021). Retrieved 13 June 2021, from https://todo.microsoft.com/tasks/.

Project Management Software. ActiveCollab. (2021). Retrieved 13 June 2021, from https://activecollab.com/.

Redux - A predictable state container for JavaScript apps. | Redux. Redux.js.org. (2021). Retrieved 13 June 2021, from https://redux.js.org/.

SitePoint – Learn HTML, CSS, JavaScript, PHP, Ruby & Responsive Design. Sitepoint.com. (2021). Retrieved 13 June 2021, from https://www.sitepoint.com/.

styled-components. styled-components. (2021). Retrieved 13 June 2021, from https://styled-components.com/.

TasksBoard | Desktop app for Google Tasks. TasksBoard. (2021). Retrieved 13 June 2021, from https://tasksboard.app/.

Todoist: The to do list to organize work & life. Todoist. (2021). Retrieved 13 June 2021, from https://todoist.com/.

Yangkatisal, M. (2021). *The Battle of the NoSQL Databases - Comparing MongoDB & Firebase*. Severalnines. Retrieved 13 June 2021, from https://severalnines.com/database-blog/battle-nosql-databases-comparing-mongodb-firebase.