FIT3162 Team 23

Project Final Report

Word Count: 9725
(Not including cover, headers, images, code, etc)

Project 8: KanbanX - An improved tool for Project Management and task allocation

Team Members	
Marcus Facchino	30604176
Matthew Zupan	25122533
Stephanie Tjin	31580025
Jiangye Song	31186408

Table of Contents

1.0 Introduction

This Final Project Report written by Team 23 consisting of Jiangye Song, Stephanie Tjinn, Marcus Facchino and Matthew Zupan involves the discussion of our projects successes and shortcomings within the following five main topic areas being:

The Project Background in Section 2.0. Here we introduce our project (project 8), which has been to produce a Kanban Web Application named KanbanX with the goal to demonstrate an improvement upon existing Kanban applications. We explain the rationale for choosing this project and a background in order to provide context to the improvements demonstrated. This section involves a literature review covering academic materials in relation to the Kanban ideology, Agile project management methodology and its application.

Outcomes discussed in Section 3.0 outline the features which have been implemented in the current state of the project with a discussion as to how the requirements of this project have been met and extended upon in order to consider the result of the project a success. We justify our design choices and rationale from the perspective of both the development team and the end user, and provide consideration to the limitations of the project which allow us to come to a greater understanding of the future actions and implementations that would lead to a more complete and desirable outcome.

Methodology is identified and discussed in Section 4.0, where we divide this discussion into both the methodological principles adhered to throughout, as well as the physical process of design itself (the implementation). The latter refers to the ‘what’, being the tools used and a description of their purpose. We compare in this section the differences between the initial project proposal and the current outcome in correspondence to our previously (last semester) identified user acceptance criteria.

Software Deliverables are covered in Section 5.0. Here we include not only the software itself and summarily state how it is used, but a detail of the supporting documents such as the User Testing Guides (User and Technical), as well as the Testing report which are necessary in order to provide a more complete package for the end user. Deliverables themselves possess certain qualities, of which software qualities are most important which are: robustness, security, usability, scalability, documentation and maintenance. We discuss how our software possesses these qualities and identify what may be necessary to satisfy these to a greater level.

The last main section is the Critical Discussion in section 6.0. Here we derive as a brief extension to contents of previous sections, a discussion of what changes are necessary (especially with respect to tooling, implementation methodology) that would facilitate a better outcome based on the functional and nonfunctional requirements brought forth as a revision from our initial proposal. Here we seek to provide those extra considerations not covered in previous sections.

Finally, we conclude our report with a summary of the main topics and important discussions identified. We provide additional materials in the Appendix (such as a required code example and the explanation of how this satisfies project outcomes), along with a tailing set of references to any external materials cited for the purpose of discussion and reflection upon our work.

2.0 Project Background

2.1 Background

Kanban boards, derived from the Japanese term "kanban" meaning signboard, are a potent tool for project management. Originally employed in manufacturing, this concept has found its way into software development and various other industries. A fundamental board consists of columns populated with task tickets, with each column representing different states or processes. These tickets act as regulators, controlling the flow between various states on the board. Kanban boards serve as visual aids that facilitate the visualisation of workflows, impose limits on work in progress (WIP) at each state, and gauge cycle time to boost productivity. The principles of Kanban, rooted in lean and agile methodologies, have attracted attention as a potential approach to improving project management and productivity.

In the literature review conducted last semester, we delve into the guiding principles of Kanban as outlined by Anderson (2010), which encompass:

1. Visualising Workflow: The central tool for visualising and coordinating teamwork is the Kanban board. This board is composed of columns representing various stages of activities, such as "To-Do," "Doing," and "Done," with cards representing the features or tasks currently under development.
2. Limiting Work in Progress: The effective management and limitation of work in progress (WIP) is vital. Establishing mechanisms to control and signal when a new task can be introduced into the workflow is crucial.
3. Measuring and Managing Flow: Monitoring the Kanban process entails utilising various statistics and diagrams, including measuring cycle/lead time, tracking queue size, and utilising cumulative flow diagrams to gain insights into work progress.
4. Explicit Process Policies: Clear process policies are essential for ensuring a smooth flow, as they define the conditions necessary for the pull system to function effectively. These policies address aspects such as task assignment, activity allocation to developers, and the transition of work items from one state to another.
5. Utilising Models for Improvement: Kanban advocates the use of models to identify areas for improvement. Three suggested models include the Theory of Constraints,

a subset of Lean Thinking that identifies wasteful activities as economic costs, and other variants that focus on understanding and reducing variability in the workflow.

To facilitate practitioners in comprehending and analysing the advantages and challenges of implementing Kanban in software projects, several research studies have probed its impact. This literature review offers a synthesis of two crucial research articles focusing on the application of Kanban in software engineering and its associated benefits and challenges.

2.2 Contemporary Digital Project Management Tools

Various project management tools with a digital approach exist to aid teams in organising and collaborating efficiently. Notable online tools, such as Trello, ProofHub, Asana, and Jira, provide features for task management, collaboration, task assignment, progress tracking, attaching files to tasks, setting due dates, and facilitating team communication (Pandey, A. 2020). These tools exhibit differences, with Trello emphasising simplicity and workflow visualisation, while ProofHub offers additional task views, such as the Gantt chart (Pandey, A. 2020). Jira, on the other hand, is tailored specifically for software teams, featuring elements designed for software development and automatic reporting (Pandey, A. 2020).

2.3 Study 1 - Assessing the Benefits and Challenges of Kanban in Software Engineering: A Structured Synthesis Study

Conducted by Paulo Sérgio Medeiros dos Santos, Alessandro Caetano Beltrão, Bruno Pedraça de Souza, and Guilherme Horta Travassos (2018), this study employed a structured synthesis approach to consolidate evidence from published primary studies concerning the benefits and challenges associated with implementing Kanban in software engineering. The research aimed to pinpoint the actual advantages and hurdles tied to Kanban. By analysing 20 chosen primary studies, the authors identified significant benefits, including:

1. Enhanced Work Visibility: The Kanban board offers superior clarity on the development process and indicates which developer is working on each task.
2. Control over Project Activities and Tasks: The WIP limit directs the software team's efforts, preventing them from taking on too many parallel tasks and ensuring that they focus on value-delivering tasks.
3. Enhanced Flow of Work: Kanban provides deeper insights into team activities, thereby improving feedback loops and highlighting resource constraints.
4. Reduced Time-to-Market: While the impact of the Kanban board on software product release time is somewhat weak, it does contribute to more efficient product production.

These advantages extend to niche areas like 'portfolio management' and educational applications. However, an organisation's 'culture', 'management expertise', and 'supportive practices' are crucial for realising these improvements.

Additionally, the authors posit that while Kanban has the potential to enhance 'team cohesion' based on the observations, its success in doing so hinges on the team's willingness to embrace the mindset shifts associated with Lean principles and Kanban methods. This implies that for Kanban to positively influence social dynamics, there must be a foundational openness to change and collaborative work within the team.

2.4 Study 2 - Examining the Impact of Kanban in Software Project Work: An Empirical Case Study Investigation

Authored by Marko Ikonen, Elena Pirinen, Fabian Fagerholm, Petri Kettunen, and Pekka Abrahamsson (2011), this empirical case study investigates the influence of Kanban on software project work in a Software Factory research and education setting.

The authors state that some common project risks in software development utilising Kanban mitigates, such as those associated with scheduling and scope. Kanban's method of joint goal-setting by customers and developers, along with its flow-focused, incremental task structure, enables precise, data-driven scheduling and budgeting. Lean's "defer commitment" principle guides decision-making when most informed, reducing uncertainty. This flexible system adapts to changing requirements, streamlines decision-making, and prevents unnecessary features, thus improving requirement management control.

The study relied on empirical data gathered during a case study of a team employing a Kanban board for project management. The research methods encompassed video and direct observations, one-hour thematic interviews, diary records of observations, and taped interviews. The findings revealed that Kanban facilitated problem-solving by illuminating workflow bottlenecks, promoted informal and open communication, and encouraged shorter feedback cycles through task decomposition. The study also highlighted the intuitive nature of Kanban and its ability to empower developers in task selection.

They also emphasise that the more Kanban is scrutinised, the clearer its adaptability and potential benefits become, pointing to a rich field of research yet to be fully explored and utilised.

2.5 Concluding Remarks

By amalgamating insights from these two studies, it becomes evident that Kanban holds significant potential for enhancing software engineering, with benefits including improved work visibility, better control over project activities, enhanced workflow, and reduced time-to-market. Nonetheless, the influence of organisational culture during Kanban implementation cannot be underestimated. These studies provide valuable guidance to practitioners considering the adoption of Kanban in their software projects, enabling them to make informed decisions and leverage the method's advantages while addressing challenges effectively.

3.0 Outcomes

Within the section of project outcomes we will discuss the features that have been implemented along with the results achieved in the product. We shall discuss how project requirements have been met and justify our design choices due to the decisions made throughout. We will further discuss the limitations of our work and extend upon this by considering the possible improvements that could be applied in future if this project were to be given further attention beyond the scope of the semester. We save further discussion of meeting acceptance criteria in Section 6.0: Critical Discussion.

3.1 What has been implemented

Over the course of the semester, we have implemented the following features to our project. A more thorough and complete guide of the KanbanX application contents exists in our previous report “User Guides” (note that usage will be covered in Section 5.0: Software Deliverables), the core contents of which are provided in Section 8.0: Appendix - User Guide Contents. So we will only summarise what has been implemented at a high level here.

Sign Up, Sign in and Sign out:

- Sign up, if you do not already have an existing account and would like to become a user of the application. This involves creating a username and password which registers the account to the application.
- Sign in, if you already have an existing account and would like to login. This involves entering in registered account details to access the application contents.
- Log out of the application by clicking on the user profile icon in the top right corner of the navigation panel and selecting ‘log out’.

Board

The board page provides multiple features to the end user, these are:

- The ability to view a project Kanban Board and create and move tickets on this board. Board columns are titled ‘TODO’, ‘PROGRESS’, ‘COMPLETED’ and ‘ACCEPTED’. The ticket ID, title and description are visible.
- Click on ticket ID to go to an expanded view of the ticket.
- The ability to create a new project through the projects dropdown menu, as well as select the project board to view.

Plan

- The plan section allows for viewing of information related to the past, present and future sprints in a project. Extending upon this, a new sprint can be created and the current sprint can be completed. Sprints are short bursts of work on specific tasks that can be repeated throughout the project lifecycle.
- Ticket creation for the current sprint can be performed in the plan section for the currently selected project, which is then added to the project board.

History

- The ability to view a project boards' history is available in the history section. A date-time selector can be used to view the state of the board at the chosen date-time. The projects' Kanban board will then be displayed statically (ie, non-interactive).

Ticket

- Tickets refer to Kanban cards in this application. These tickets while present on a given project board, can be clicked on to provide an expanded view which presents more detailed information on the ticket itself, such as the user assigned to work on the ticket, the project and sprint it belongs to, the ability to comment on tickets and read ticket comments, view and edit its description, as well as to change its state on the board. Changes can be saved within this expanded view.

3.2 Results achieved/product delivered

The results of the currently delivered product follow from the implementations outlined previously. Users are able to sign up and log in to the application, create and interact with projects, sprints and Kanban cards that can be moved across the Kanban board based on the status of any defined task. With the ability to view board history and leave comments on tickets, progress and decisions made can be monitored. We will outline the implications of each major feature next.

Sign Up, Sign in and Sign out

Sign up and login allow for users to interact with the application uniquely, that is, to have a distinct profile that cannot be accessed by any third party. This provides a level of security, such that unwanted users cannot view or modify the contents of a project nor access sensitive data/information.

Logout prevents those who have left their computer etc running from having others enter the application and modify/view any information that they should not have permission to do.

Board

The board page provides a high level overview of the current state of a project, such that the tickets within can all be viewed to understand which tasks are in progress, need yet to be started, or have come to completion or require review. The columnar representation of the Kanban board provides a simple and easily readable temporal display of the project state. The ease of interaction with the board by dragging cards across columns removes unnecessary complexity of manipulating this high level overview. Furthermore, with project and sprint information clearly displayed above the board, there is little confusion as to what the board information is referring to/represents. Relevant information and features to a project should be easily accessible from the board page in order to avoid the need to traverse across multiple pages in order to perform a project specific action, hence we include the ability to create and select projects, as well as create tickets within the board page to keep this functionality cohesive.

Plan

The plan page introduces a bit more complexity as this delves into project specifics. It is not regarded as the high level overview, but allows for manipulation of sprint activities. Given a project may consist of one or more sprints, we can plan for future sprints, creating and assigning tickets as necessary into a backlog such that when the current sprint has ended, the new tasks are present in the next sprint. By selecting a past sprint, a tabular summary of the tickets on the project board are presented where the state of the board at the end of that sprint is captured. The consequence is that it is then clear what has remained unfinished and what is required to be ported over to the next sprint. This captures the essence of planning, as we can see what follows from the previous and create what is necessary in response.

History

Board history provides a higher degree of temporal context beyond the tabular summary of ‘end of sprint’ information, because it allows a more fine grained view of a selected project board based on exact date-time specifications. A specific ticket or collection of tickets can be seen to be stagnant when present in the same board location over some duration. The direct implication being that either the task is

complex and requires more time to complete, or an assignee is stagnating in their work. Notably, viewing history allows one to see at what time a ticket has moved state or to be able to reference the time that progress for the ticket commenced or completed for reporting reasons. The date-time selection feature, allowing for calendar date selection and its accompanying clock selection sub-feature allow for a more flexible granularity in the selection process when varying levels of time-based specificity are required upon review. However, it should be noted that the board state is not represented as a ‘snapshot’ at every minute interval, given that the amount of data required to store this grows largely. Changes are stored only when they occur, therefore the history presents the most recent version of the board at or prior to the selected time.

Ticket

The expanded ticket page provides the user extra support and access to ticket-specific information. Having a single user assigned to the ticket prevents duplication of work for that ticket and the ability to view all ticket information in a single location and modify that information allows for task specific consistency. An editable description allows errors or misinterpretations regarding the nature of the task to be fixed or elaborated on, and the ordered listing of comments provide not only questions and updates related to the ticket, but the time-based context of this information. Given that we can also change and save information, we do not need to go back to the board to move the ticket, thus giving us greater control without having to jump back and forth between board and ticket views.

3.3 How are requirements met

The project brief outlined four main requirements for the Kanban project, these being:

1. The ability to view history of the Kanboard, ie, allow users to view how the board was at various times earlier (eg, 1 or 2 weeks earlier).
2. The ability to pre-commit tasks, ie tasks that would be undertaken in the future.
3. The ability to view/monitor each team member’s percentage contribution to the whole project, and better allocate tasks to team members.
4. Access to improved collaboration and communication between team members.

These requirements represent an ‘improvement’ that a Kanban application can have. However, this set of requirements is not comprehensive, thus other improvements

may be considered. For this reason, we have replaced requirement 3 with a different feature, being:

3. The ability to break down projects into smaller units (sprints).

We have met the requirements 1,2,3 and 4 in the following way:

1. The board history page allows for date-time selection of a selected project, providing a view of the state of that kanban board at the selected time.
2. The plan page allows for ticket creation, where the backlog is available for a future sprint.
3. Sprints can be created and completed in the plan page, the board page displays the kanban board for the current sprint on the currently selected project. This board is interactive.
4. Tickets in the expanded view have the comments feature, displaying all comments on the ticket which can be read and responded to.

These requirements occur in conjunction with the implicit requirement that we provide for a regular functioning Kanban application with the previously outlined being integrated within. Not just separate ‘improvement’ components, but instead provided within a contextual setting. Furthermore, it is implicit that the product has been tested and working at least on a basic level, of which we have somewhat achieved (discussed further in the Limitations subsection).

3.4 Justification of decisions made

Implementation decisions and design alterations occurred for three main reasons:

1. Time and skill limitations - Team members were required to maintain a work/life and study balance across multiple units while satisfying project outcomes. Additionally, team member experience in relation to this project varied.
2. Design Complexity - Considerations to the need for restructuring the code and interactions across modules.
3. User Convenience - How the end user would receive, interact with and learn to utilise the applications functionality.

Time and Skill

The time available for members to work on the project was limited and due to the lack of experience, we were unable to implement more advanced functionality such

as tagging users in comments and the display of user statistics in a graph on a separate page. So we had to keep design simple and allow for basic functionality while showing some improvements as described before. Having too many features would have significantly reduced the quality of each deliverable, with insufficient time spent to achieve a working standard beyond a prototype.

We had omitted the ability to rename Kanban board columns as this would require a more advanced script to then run through the database and change all instances throughout to reflect the renaming, with possible data inconsistencies. Figuring out how to make alterations to the database to extend upon existing features (such as adding an estimate of task difficulty as user work scoring method) already posed a challenge, so we omitted this functionality as well.

Design Complexity

Maintaining a modular design, such as modules for the expanded ticket, board, board history, the plan page and such kept the development of such items within their own scope. This helped reduce the complexity of design by allowing items to be developed in their own location and reduce the amount of coupling needed between modules. Having a design that induces too many dependencies and inter-related items increases the debugging time due to one change somewhere affecting many other items elsewhere. We also wanted the project to be scalable, thus packaging together related modules was necessary. In the backend for example, we keep controllers together, fetch requests etc. We also separated our frontend and backend, with vscode using javascript and tailwind css as the development environment for the front end such that we can style and manipulate data on this side once transferred from the backend logic and controllers, which themselves provide the servicing and structure of the application. The backend utilised IntelliJ and java for this purpose as it integrates well with the data related functionality.

User Convenience

User convenience was a significant driving factor with regard to the design decisions made. We chose a web based application for accessibility, requiring sign up and login to commence use and operation, this along with the ability to log out was intended as a layer of security. User technical experience must also be considered, so for that reason breaking up core elements of functionality such as project creation, the Kanban board overview, expanded ticket information, history and planning/sprint manipulation allowed for relevant information to be located in mostly one place. We included a navigation panel at the top with the same layout across all

of the above mentioned pages to ensure consistency in design and reduce the need for learning.

Using an interactive Kanban board provides immediate visual feedback of the board's state, the ability to edit ticket descriptions within the expanded ticket reduces the propagation of errors and the separation of the board history page from the current board reduces confusion as to what is being viewed. We wanted our board history selection process to be symbolically conveyed, and so the calendar date and clock selection widget made it clear what the selection represents.

While the plan page involved more project specific information, we wanted to ensure that components appeared together but were clearly distinct from one another. Which is why we used a tabular format of sprint information at the bottom of the page, with sprint manipulation and ticket creation in a separate panel above.

3.5 Limitations of project outcomes

The current limitations of the project depend on the disparity between expected and projected functionality. However, from the basic perspective, we have the following limitations:

- The inability to assign more users to the project, including the inability to assign users to tickets and thus to re-assign users.
- The inability to reset or recover account information if it is lost/forgotten.
- Assigning tickets to different sprints is not functional.
- Inability to assign a work estimate (weighting to the ticket), such that this weighted contribution by assigned users can be compared and presented graphically according to an average of all other users working on the project.
- Inability to rename Kanban board columns to better suit the project nuances.
- Inability to be tagged in comments and pinged to be notified of this.

A more projective set of limitations include:

- The inability to schedule tasks of a recurring nature, that is to automate assignment to future sprints. This includes the limitation that sprint starts and completions are actioned manually.
- Advanced ticket creation features are not implemented, such as assigning a type to the ticket (for example, ‘feature’, ‘prototype’, ‘test’ etc).
- Breakdown of projects into ‘features’ (sub-projects), which themselves contain sprints.
- Inability to delete tickets or reverse/revert changes, such as reversing the action to complete a current sprint if done by accident.

- Inability to personalise account details/information to a level beyond the sign up functionality.

3.6 Possible improvements and future works

Necessitated by the current limitations of the application we should consider addressing each of the previous as an improvement upon our current work. We also consider a few other ideas that would be interesting:

- The ability to block tickets from being started until the dependence is marked as completed. The removal of this block is automated.
- Restrict access to certain actions such as moving tickets that do not belong to a user, unless their access level is higher than that of a developer. This can involve the ability to grant and revoke access.
- Include a leaderboard which displays developers who have completed the most work.
- Include tooltips when hovering over buttons to provide hints on their functionality.
- Include a help page in the application which addresses main usage questions and links to a complete user guide.
- Colour theme customization for users with visual impairments, so that features are easily distinguishable.

4.0 Methodology

In this section on methodology, we focus on two main aspects of the project. These are:

1. The aspect of design, which includes a brief of the initial proposal and moves on to a discussion of the deviation and impact of this between this initial and current design.
2. The Methodology (principles) adhered to during implementation of the project in relation to task management, and the physical methodology in regards to design itself, tooling/software used and how this process took place.

Our intention here is to more so convey how we arrived at the current state of the project, as we will cover what the project looks like in Section 5.0: Software Deliverables.

4.1 Design

Initial Proposal

The initial design from semester 1 was a listing of user acceptance criteria rather than a physical design itself, and for this reason we list these criteria here:

- An acceptable product will run on any pc, laptop or tablet device with an internet connection able to connect to the application webpage. Given that a website is being developed, access is independent of the operating system of the end users' device.
- A login screen that allows the user to create an account with a username and password if their account does not already exist, and to be able to enter in the user login details to access the rest of the application functionality if an account does already exist. Users who close the webpage are logged out by default.
- From this point onwards, the application must establish a permissions hierarchy such that additional project security is implemented, allowing for user access levels such as:
- Administrator, with the greatest level of access, to be able to create and delete projects, features, epics and tickets as well as add users to such projects and assign their access levels. The administrator is thus considered as an owner of generated project and user data.

- Project leader, with unrestricted access the same as an administrator has, but only within their assigned projects, project leaders can accept users into projects but only assign developer level access and no level higher.
- Developer, with the least access. Developers cannot add or remove tickets, assign users to tickets or add users to projects, nor grant permissions. They are unable to view any project outside their own assignments. However they are able to comment on all tickets and work on their assigned tickets as well as view board and user metrics.
- A menu bar which contains the menu options to change pages/view from a selection of projects in the project option, to access the historical board selection page from the board history option and to access the user metrics page from the user metrics option. Project selection also involves a nested subset of selection options to select the relevant epic and feature within a project.
- The ability to view the state of a project board given by a time specification, that is, at a specified time, the tickets present on the board at that time are displayed in a separate historical view in the columns they resided on at that time.
- The ability to assign to each ticket a value which weighs how many units of work the ticket is worth.
- The ability to display how much work a chosen user has completed in comparison to all the tasks during a given time interval based on the weight of the tickets in this interval. This includes the ability to display rankings of users against other users based on how much work was completed, say in the form of a pie chart or bar graph.
- The ability to move cards across the columns of the Kanboard for users who are working on that given card/ticket, or in the case of users with project leader or administrative permissions to do so as well. Tickets that have been placed in the done column will show up in user metrics searches as the user having completed the work.
- The ability to comment on tickets for all users assigned to a given project for which that ticket is located within. This includes notifying an user who is tagged in a comment with the ‘@<username>’ tag.
- The ability for users to read comments in a separate popup near the ticket when the comments option on the ticket is clicked on, the user is defaulted to their least recently unread comment.
- A database for which to hold all related data and relations between data for storage and processing, of board view and metrics as well as permissions, assignments, projects, epics, features, tickets, comments and login data.

- The ability for a ticket to display the derived attribute ‘days since creation’ in days to easily determine how long the ticket has been active for.
- The ability for project leaders and administrators to pre schedule tickets in project boards to begin at a given date, with the option of allowing recurring pre scheduling intervals.
- Periodic purging of tickets from the done column in a project board to avoid excess build up of tickets in this column, with the option to purge based on a selected time period (multiples of one week).
- The ability for only one user to work on a ticket.
- Product testing by the team to ensure reliability, accuracy, security, quality and that acceptance requirements have been met.

Final Design And Deviation From The Proposal

The final design implementing the features identified from Section 3: Outcomes deviates from the above in the following ways:

- The product is a web application but requires Windows or Mac running our software stack as it has not been floated as a standalone website. This reduces accessibility.
- The permissions hierarchy does not exist, which means that projects are viewable and interactive to any user. A reduction in information sensitive security.
- The navigation/menu bar does not contain user metrics, so user workload is not tracked, reducing user workload analysis ability.
- Comments on tickets do not involve username tags, so there is no explicit notification to alert relevant users of potentially important information, such information can be missed.
- Ticket creation time is not displayed, although it does have an entry in the database. While board history can determine this metric, it could be made more accessible, say viewable directly on the board or in the expanded ticket page for simpler analysis.
- Purging of tickets can be done manually by ending a current sprint, rather than the automated periodic purge. An automation feature is convenient, the manual option does allow for greater flexible pacing control however.

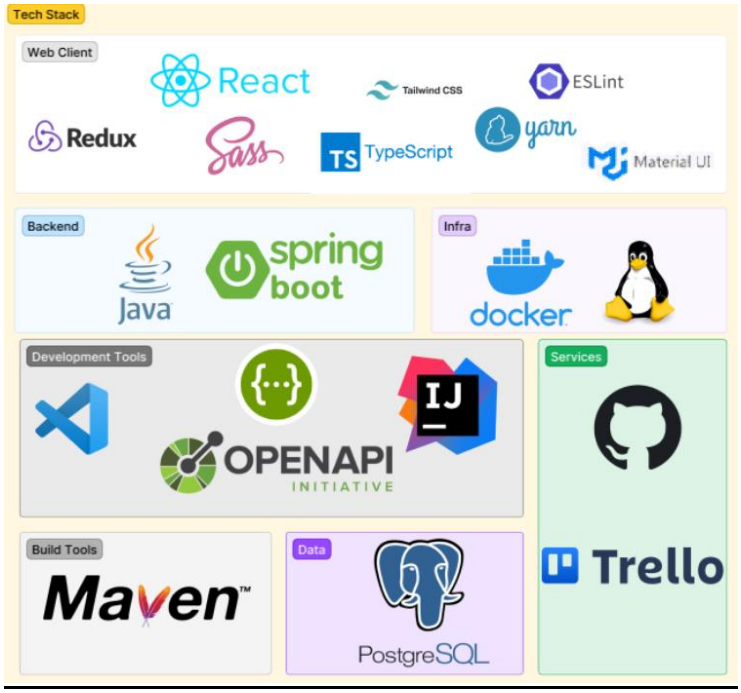
4.2 Implementation

Task Management

Task management utilised Trello, an online Kanban board tool to manage tasks with team members assigning themselves to tasks and moving them across the Kanban board based on completion. The Kanban methodology considered by us to be encompassed within the Agile methodology allowed for tasks to be re-assigned if a member had excessive difficulty, or to move a task back from the completed column to ‘todo’ based on any future extensions that were deemed appropriate, such as extending upon the information present in ‘expand ticket to view’.

While the user acceptance criteria is considered highly predictable, such that a waterfall methodology can be used (prototype, develop, test), we still required the flexibility of Kanban for greater task tracking, and an organised presentation of all tasks at hand. Movement to the progress column determined when a task had been started, and movement to ‘review’ indicated that it was time for another member to pull the separate branch in which this task took place, test that branch and then provide comments and feedback based on the result. A successful review would then see to the merging of the task and any changes with the master branch on Git. This process/methodology remained the same throughout the project lifecycle.

Software Stack



We made use of React Typescript as our framework and our language to build our client. React is a popular framework which uses Javascript and Typescript. We chose React as it is a popular modern framework used in the industry so future engineers who work on this will not have trouble finding resources to troubleshoot errors**.** We used typescript to ensure we have type safety in our language and errors that we don't propagate stupid type errors to the user and keep our solution robust. To accompany our React we make use of popular libraries such as Redux which is a local data store that allows us to store data and access it across our application similar to a local database. WE use tailwind CSS, Material UI and SASS to build and style our UI components to a consistent standard. ES lint is used frequently to keep our typescript files to a specific code standard and set custom rules such as ensuring we handle null variables and not use the '!' operator which tells our compiler this is not a null value which is bad practice. Lastly for our client we use Yarn package manager to install, add and remove packages.

Our Backend uses Java, a popular programming language that can be run on most machines. We chose to use Springboot, a popular and mature API server framework. Springboot was chosen due to its heavy presence in the industry and its wide adoption also means any future work done on expanding the solution will mean lots of documentation online.

For our infrastructure we use linux as our development OS using any flavour. We use docker to run our database locally alongside docker will be chosen to package our solution and deploy it.

For our development tools we made heavy use of VS Code for our Non Java development and for our Java Development we made heavy use of Intellij. To build our Java code we use MAVEN similarly to yarn; it provides us with an interface to add and remove packages to help development.

We used postgresql for our database services. Postgresql provides a strongly typed system that ensures our tables and relations contain no mismatches on deployment.

We used trello to keep track of tickets and activities so we had oversight into what other members of the group were doing. Github was used for our version control to help us work separately and merge and review code.

5.0 Software Deliverables

The software deliverables do not come standalone as a set of source code/packaged software, as it is required that software come with a manual for usage, in our case a User Guide, which additionally includes a Technical Guide for installing, running and

even being able to begin developing with the code. The software deliverable itself also inherently consists of its own internal software qualities, being: robustness, security, usability, scalability and documentation/maintainability. Here we introduce the previous and provide a sample of our source code in the Appendix under the title ‘Code Sample’ to demonstrate how software requirements are being fulfilled.

5.1 Deliverables

Software

We will provide only source code and not a packaged version of the program to deploy. We will provide a website alongside documentation inside of our README in our source code on how to start up the solution. We will deliver an API server which will handle requests sent by our Kanban Board website. Our website is a Java Server that takes requests. Our Client Interface is a react website that can be started and hosted which interacts with our backend. We also provide a database docker image with an accompanying script to start and initialise the database which our Java backend talks to.

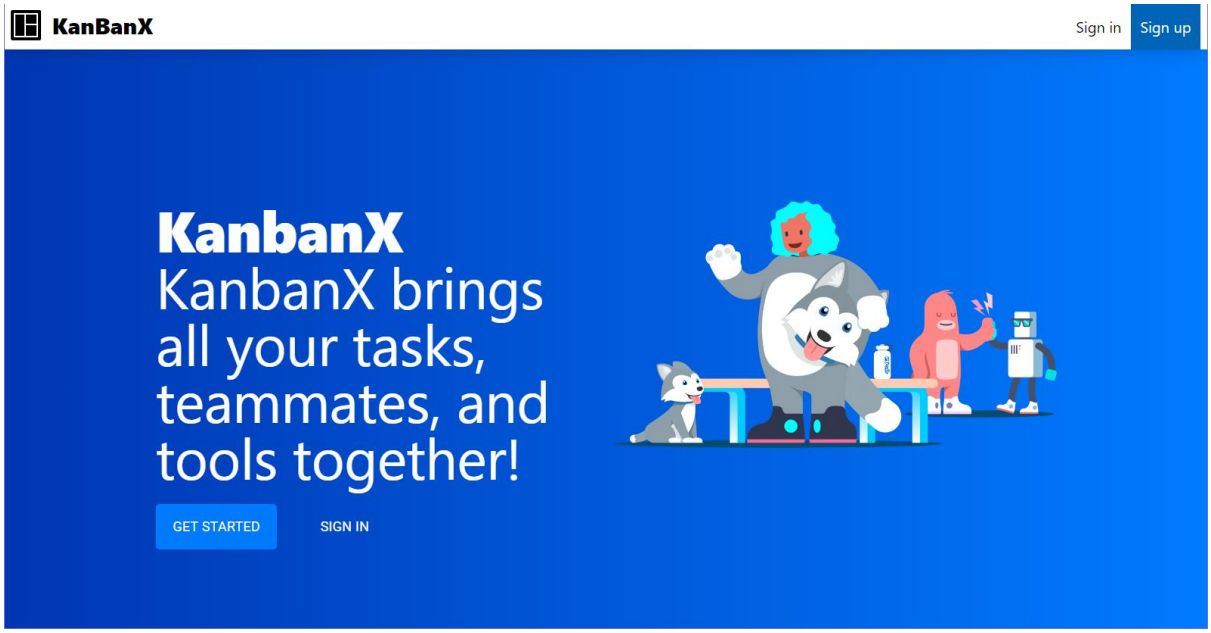
User Guides

The user guides have been presented as a report written by the team and uploaded to Moodle as part of the semester 2 assessment. Furthermore it is available on google docs but with restricted access to those who are given explicit permission to view and/or edit this folder. The Moodle upload is considered the most important here as it is readily accessible to the teaching team who is considered our ‘project sponsor’ given that we have no other external party that this project affects.

The user guide is a thorough documentation of all of the software functionality and comes complete with contents and screenshots of all processes and features. These are documented in detail, with explanations of where items are located, their exact functionality and how to read and use them. An extension to this guide is the Technical Guide, which covers the main aspects of software installation. This contains the contents of the teams’ Github page explaining what software is used along with what commands are used to proceed with the installation process and how to access the collaborative source code from Github. Here we assume a Windows machine is to be used. The main three topics covered are Git Branching, setting up the development environment and building and running (the server and website). We provide the contents of this in the Appendix under the title “Technical Guide Contents”.

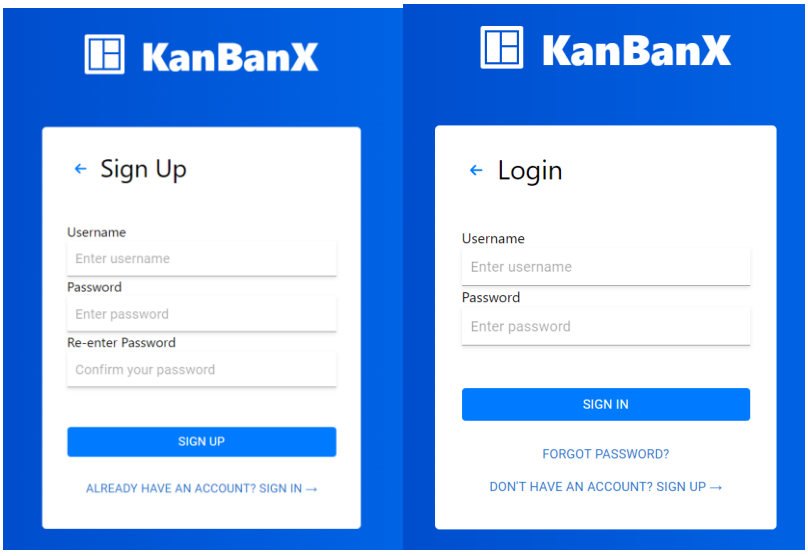
We will cover usage here to a lesser extent than present in the User Guides:

Landing Page



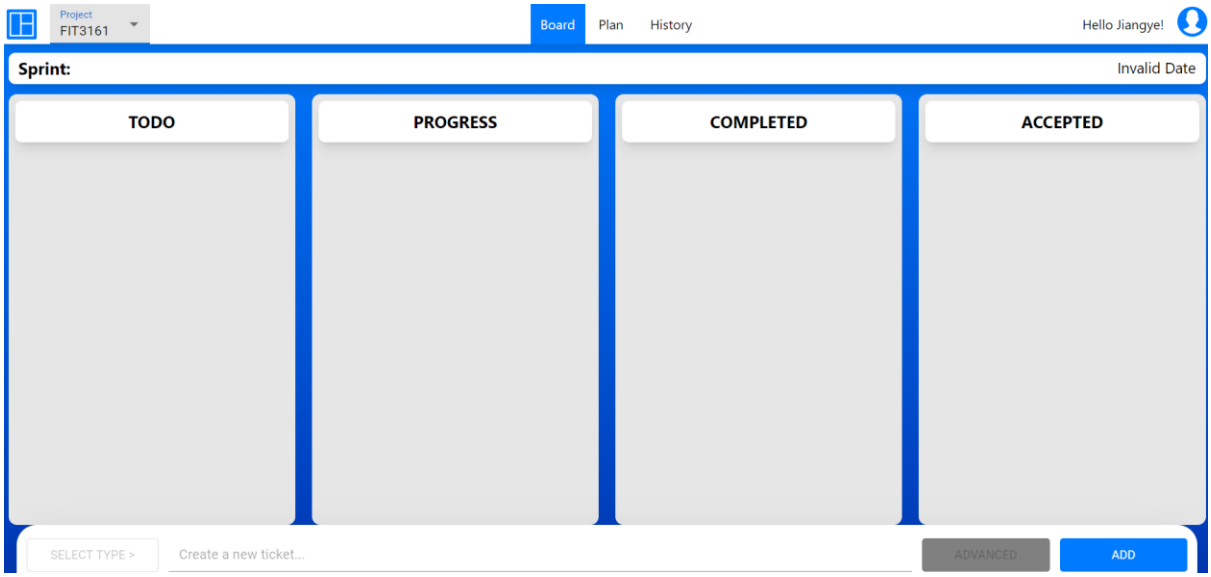
Here we arrive at the main landing page when entering the website, where the options to sign up or sign in are present.

Sign Up and Sign In

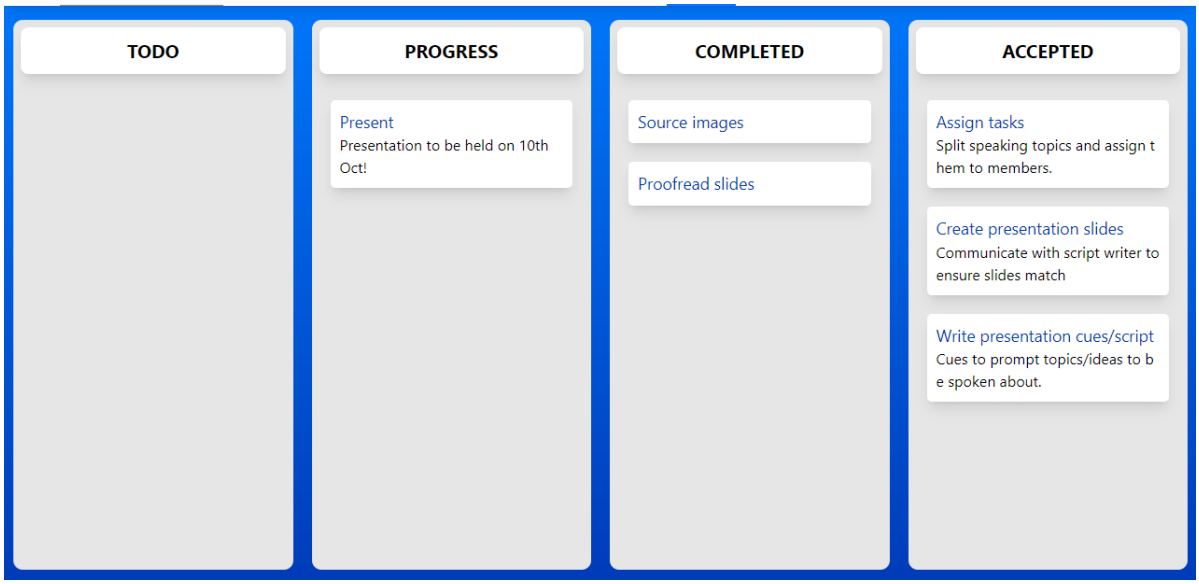


To sign up, enter a username and password, then re-enter the password to ensure correctness and click sign up, then the login option becomes valid, alternatively if we have login details already, we can directly login by entering our username and password and click login.

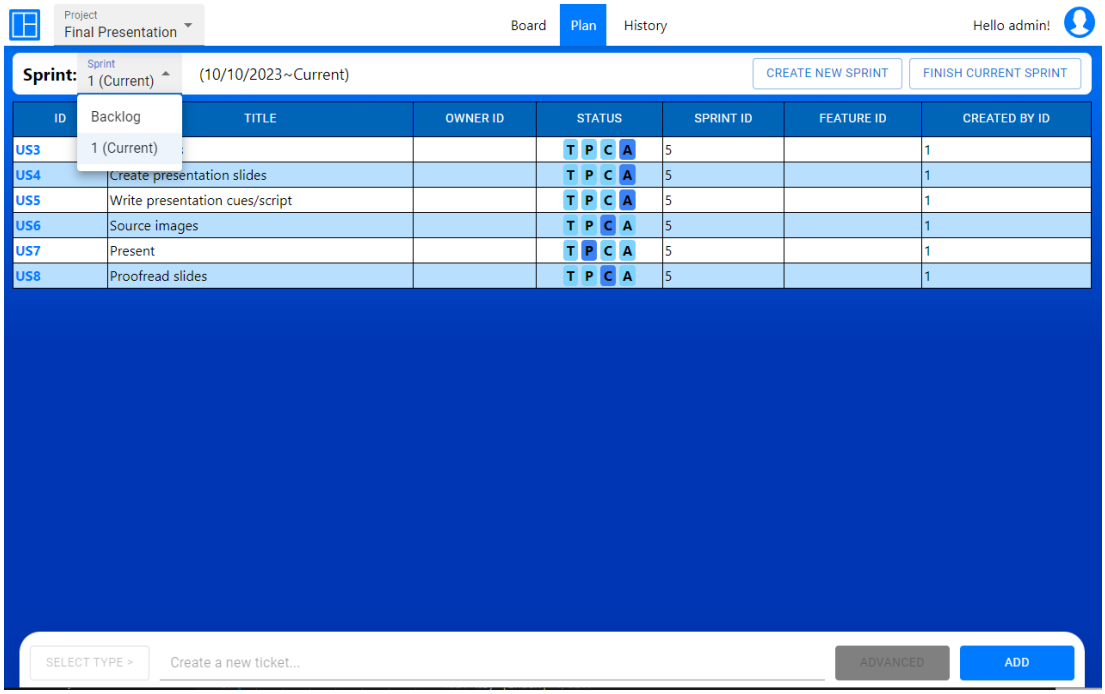
Board Page



The board page offers the ability to select or create projects by clicking the top left project dropdown, the create new ticket function here is erroneous thus we should only create tickets for a project in the Plan page (present in navigational panel). Tickets appearing on the Kanban board in this board page are draggable and clickable to enter an expanded view.

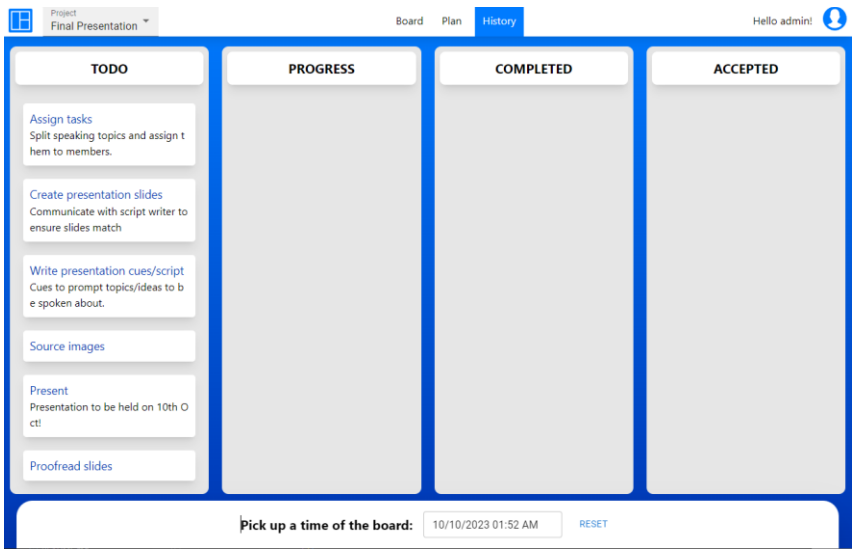


Plan Page

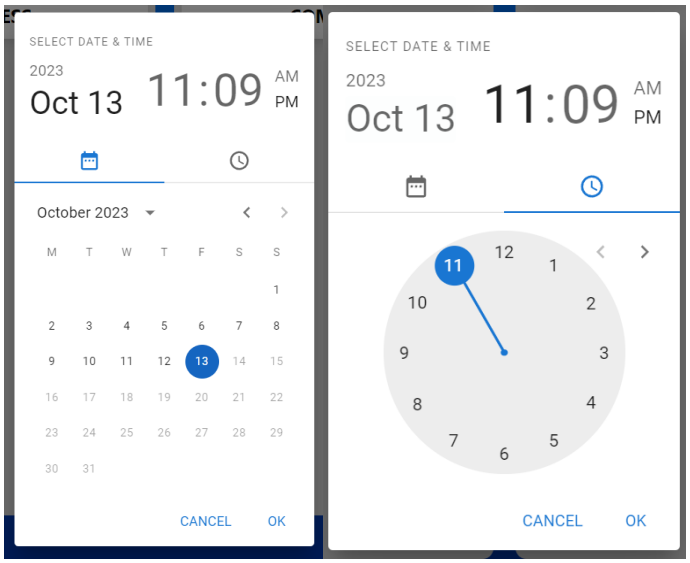


Tickets can be successfully created here, sprints can be selected for view in the table, presenting a summary of tickets in the selected spring. Sprints can also be created and completed to enter the new sprint.

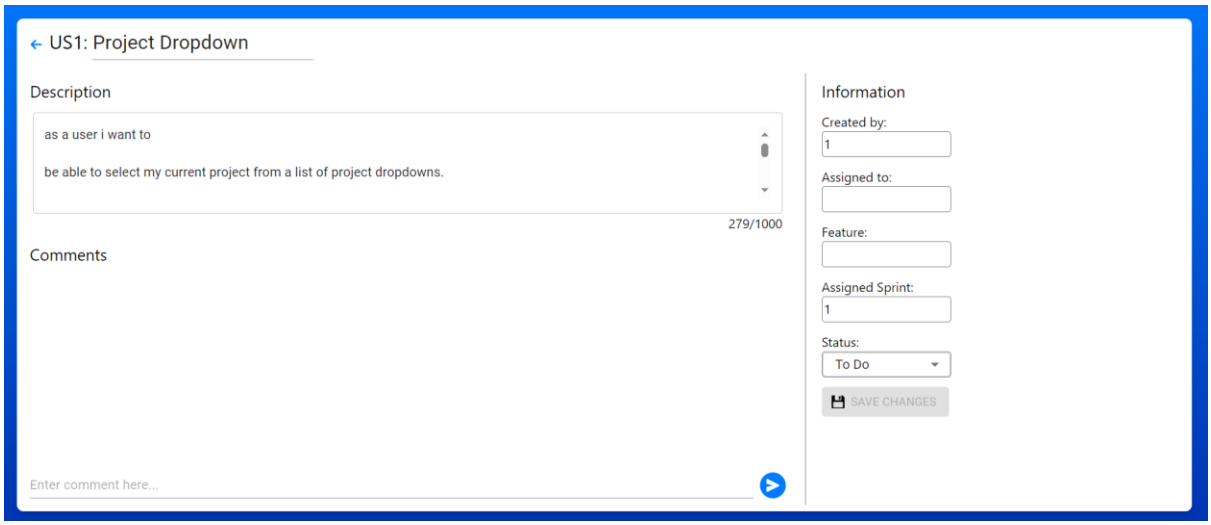
History Page



The bottom panel allows for one to choose the date and time for which the view of this board is updated to reflect the state at that selected date-time. This is a static display. The date-time selector has a calendar-clock format as such:



Expanded Ticket Page



This page includes more detailed ticket information (description, comments, user, sprint, board column) and is editable by clicking save changes after a change is made. Posted comments are automatically saved.

Testing Report

The testing report is available within the same Moodle upload as the User Guides and is a rudimentary report detailing our testing procedure. As an outline of this process we have employed the following testing methodologies: Whitebox Testing, Blackbox Testing, Integration Testing. This comes with a discussion of the testing approach and limitations.

- Whitebox testing involves writing unit tests in our Java backend using the Springboot testing framework.

- Blackbox Testing involves manually testing the application by interacting with the site after/during the implementation of a feature. For example, moving a ticket across the Kanban board to check for correct behaviour.
- Integration Testing involves testing the interaction of multiple implemented features together as a whole. For example, we can test ticket creation on the plan page, movement on the board page and then re-movement from the expanded ticket view page to check that changes are stored correctly when displayed on the site in comparison to the actions we have taken.

We discuss in our test report how unit testing and integration tests are run. We have laid our stages and steps on how they are conducted and run in our application. Our unit tests are run with the maven test command. Which includes our whitebox box behaviour. Our blackbox testing is a set of steps around the feature / parts of the program we edited or made changes to, to test that our changes did not affect other parts of our site or had unintended side effects. Finally our integration tests are done by hand and by following steps. More detail is mentioned in the Testing guide.

5.2 Software Qualities

Here we refer to and discuss the properties of which our software deliverable possesses both in relation to current and future contexts if we were to continue development.

Robustness

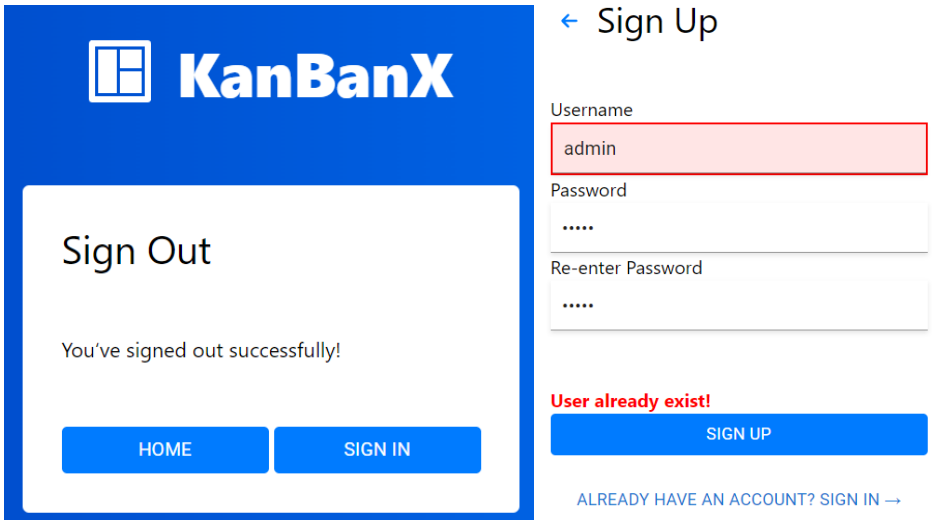
Robustness being the ability to cope with errors during execution, we have not detected any fatal error such as a crash when testing and using the platform. Largely, any error that occurs results in a no-action event. With specific reference to the saving of changes such as assigning a ticket to another sprint within the expanded ticket page simply resulting in no action. The other example is that of a disappearing action, such as the creation of a ticket from the board page, in which the ticket disappears from view after the board page is re-accessed. This is because it is not saved to the back end here. We outline these issues in the user guides and discussion of limitations.

In terms of inputs to ticket descriptions, exceeding the maximum description length disallows the user from committing the description, which is an intended action. Posting large comments allows the comment section to be scrollable such that the entirety can be read, and providing incorrect login information provides immediate feedback to the user that this information is incorrect without causing any non intended error. Robustness within the development environment is ensured through

use of Linting (ESLint) which visibly prevents the developer from further testing a feature until the detected issue is resolved. This greatly reduces the likelihood of coding errors persisting throughout the design.

Security

Security is enforced within the application through use of the login feature. However, given there is no ability to recover account information if lost or forgotten does pose an additional risk, that is the need to store retrieval information elsewhere external to the application, which may potentially not be secure. Access to projects is also restricted given only one user may interact with their project, which does reduce the risk of others meddling with or accessing potentially sensitive information. Furthermore, we have the logout feature such that when not in use, information cannot be accessed after logging out. No consideration to injection attacks has been given, nor a timeout feature to prevent rapid sequential brute force login attempts. Such an attack can occur by first attempting to identify an existing user through brute force feedback, then a brute force password guess, elaborated on in the Appendix under the title “Attacks”.



Usability

From the perspective of User Experience (UX) design principles such as inclusivity to those with visual impairments etc, we have not given consideration. From a simplified design perspective, we have taken the initiative to provide the user guides, as well as grouping similar features together (ticket information in expanded ticket, board information in board, sprints and planning on the plan page, and board history on the history page). We have taken the care not to excessively clutter these pages such that there is no visual overload taking place, nor the need to learn a large deal

in order to begin using the application. Buttons label their functionality and dropdowns list their contents directly.

From the perspective of manipulating the Kanban board, tickets are easily draggable and the ticket ID's come underlined in the well known and typical hyperlink fashion to visually indicate that this is a clickable item, and that this performs an action.

We have considered from Norman's 7 Principles in Usability (William Lay, 2022) the following:

- Discoverability: The ability to easily determine which action leads to the intended result. We could have improved this through the use of tooltips, but have attempted to increase discoverability by giving buttons a name, keeping design simple and consistent (conceptual model) and providing the User Guides.
- Affordances: The available/possible interactions relating an items' properties and the user's abilities. This has not been given much consideration, nor is it clear exactly how to cater to this principle other than to make the Kanban board look more like a board, and a Kanban card more like a ticket to derive more of a physical meaning to this online version.
- Constraints: Limiting the space of possible actions to reduce errors. This principle has not been given much consideration. Considerations for a future version of this project may allow for this by limiting Kanban card movement across columns (only allow movement by 1 to the right at a time) such that a card cannot be immediately considered 'Done' right after it has begun progress. Another useful function would be a prompt window 'Do you want to finish the current sprint?' as an intermediate stage in case this is done erroneously.
- Signifiers: Visual cues which indicate functionality. The board history date-time selector is a signifier that improves understandability, as it looks exactly as it's. intended functionality. The calendar to select date and the clock to select time. We make use of the profile circle at the top right of the screen to indicate that this item is related to actions corresponding to the user themselves.
- Feedback: An effect that occurs after an action that provides information about the current state. We implement feedback in the login and sign up page where incorrect information produces a response in red text to inform the user what issue has occurred. We also provide feedback on the Kanban board as the Kanban card is immediately located on the moved board column. Drop

down menus display as their first entry the currently selected item (such as the current project in the project selector).

- Conceptual Model: The mental map a user has of the application. We improve this by placing navigation in the same place with simple tabs that label their destination, grouping similar information together and not including too many functions on the same page.
- Mappings: Keeping similar information together. We display ticket information in a side panel together, group ticket comments together, board columns and displayed tickets all in the one window and on the plan page, label the columns in which the sprint information is displayed. The login and sign up page follow a similar format to each other to improve this mapping as well as the conceptual model.

Scalability

The code has been developed in a modular fashion, with board related items in the ‘board’ module, specific ticket related items in the expanded ticket module, planning related items in the plan module, sign up and login in their own modules etc. Every major feature which is sufficiently unique has its own module in the code which allows it to be considered somewhat separate from the rest (with the exception of dependencies, such as adding a new field to a ticket, meaning all functionality that will require this new field must be found and changed). This means that adding a new major feature can be achieved by creating a new module as a container for this new functionality, which is scalable in nature. For smaller nuanced changes, a greater amount of effort is required, by locating all dependent instances due to this change. Thus this code deliverable is scalable to an extent before significant restructuring is required. Restructuring would lean more towards the form of creating an interface for each class of related modules which require communication to external classes in order to reduce local complexity when adding functionality. The use of standard templates for web page design (a standard colour dictionary module as well) has assisted with keeping the design consistent, and more templates can be added and called upon in future if necessary.

Documentation and Maintainability

With regards to documentation, we have produced the User Guides (User and Technical) as well as the Testing Report. However documentation has also occurred within Github given that each commit during development is staged and uploaded to github, there is a clear development history for each branch. During the review process, comments are annotated to the code and feedback provided. Each of these comments when highlighted as an issue must be manually resolved. Thus from the

developer side, this version history provides a strong development context to assist with future maintenance and works. The code itself contains a sparse set of inline comments or docstrings describing functions/methods and their use, however modules and functions are given a descriptive name indicative of their functionality to make it apparent what they are designed to achieve.

From the user side, maintainability simply occurs through use of the application, manually updating project boards, adding and moving tickets etc. There is no external source of data that is required to be updated or merged in to replace any current working data and a static version of the software can be used. From the technical perspective, certain elements of the software stack may be deprecated in future such that a snapshot of the database would need to be captured and repopulated into the newer framework if a significant software change is required. However there is currently no provision to do so as this would be a very long term consideration.

6.0 Critical Discussion

This section discusses further the project outcome and whether the team believes we have met the acceptance criteria for this to be considered a successful project or not, as well as the reasons why/why not. Note that we have introduced the user acceptance criteria in Section 4.0: Methodology under the subsection ‘Initial Design’ in order to highlight the deviations introduced over the course of the semester. Here we also introduce the initially proposed functional and nonfunctional requirements.

6.1 Functional Requirements

1. The Kanboard web application should largely support an Agile methodology.
2. A web interface or application software with the ability to work collaboratively is sufficient to satisfy the outcome of this project.
3. The ability to view historical states of the Kanboard during some specified time period is necessary.
4. The ability to set goals in the future, showing ideal future states of the board on a specific date.
5. The ability to see how big a task is, that is, the weight of a task relative to the size of the project, alternatively, it is acceptable to consider weight relative to a subset of a project, such as an epic or feature.
6. The ability to assign names to tasks.
7. The ability to add tasks to the backlog that depends on another to be done before being able to add it to “Doing”. This is described as pre-committing, scheduling tasks that may be common or recurring.

6.2 Non functional Requirements

1. The project sponsors are teaching staff for the FIT3161 unit and the use of this Kanboard web application is intended to be a demonstration of the principles taught in the FIT3161 unit, with intended use limited to this demonstration. We do not expect a user base outside of this unit scope.
2. Rudimentary and basic product functionality as long as a demonstration of improvement upon existing Kanboard applications is shown.
3. Implementation in any programming language the development team chooses.
4. Use of a database to capture relevant data in a robust manner.

5. To be of no cost to the development team, project sponsor and end user (such that cost analysis can be ignored in all project scoping and proposal activities).

6.3 Analysis Of Project Outcome

With consideration to all three of the user acceptance criteria, functional and nonfunctional criteria, we believe this project to be partially successful in that we have been able to demonstrate an improvement in the form of that discussed in Section 3.0: Outcomes, in conjunction with subsets of these outcomes as ‘Functional Requirements’ 1, 2, 3, 6 and partially 7 (being able to commit to backlog). All ‘Nonfunctional’ requirements have also been met.

The extent to which requirements have been covered are not comprehensive, nor to the level of detail that would have provided a more engaging experience with the application. Particularly the limitation that multiple users cannot be working on the same project board together. We would have preferred to implement the user metrics feature as well as fixing some issues such as ticket creation and ticket saving to work flawlessly. Porting information from front end to backend proved to be problematic, therefore we will discuss possible changes to our tooling and methodology that in retrospect may have assisted in achieving a stronger outcome.

6.4 Changes

As a team we did not make any major changes in regards to features but in regards to added libraries. We added MaterialUI to our webclient which provides out of the box UI components which can be styled according to our needs. These UI components contained too much functionality that we couldn't ignore as it would go on to save us countless hours. We did hope to implement analytics although time did not permit. However this can be simply added with some SQL queries and formatting the data nicely on our frontend website.

7.0 Conclusion

Having covered the scope of our delivered project and provided the background on which it is based. We have discussed the project outcome within the key topics of Section 3.0: Outcome, Section 4.0 Methodology and Section 5.0: Software Deliverables. Our further discussion in Section 6.0: Critical Discussion included the functional and nonfunctional requirements in our analysis.

In Section 3.0, we had outlined in detail what had been implemented and determined that four key improvements had been achieved, these being:

- The ability to view historical states of the Kanban board.
- The allowance for planning and backlog to provide for future tasks in future sprints.
- The ability to divide projects into sprint intervals.
- Communication via commenting ability on tickets.

The extent of these improvements came with a set of limitations and a justification of main design decisions (based on the factors of time and skill, design complexity and user convenience), of which for a future version of the project we could further implement in order to enhance the product to a greater working standard.

In Section 4.0, we had dived into and explained our design methodology, both the physical designing itself, as well as the project management methodology used, which was largely the Kanban Agile approach. Here we identified a moderate deviation from the initial project proposal, which was necessary in order to develop the core improvements to a satisfactory standard rather than attempting to rush all facets of the proposal to completion in the form of nearly bare functionality.

In section 5.0, we present our discussion on the software deliverables, where we introduce the basic user side functionality of the application and what has been delivered as a final result. These being the software itself, both regular user and technical guides, and a testing report. We acknowledge that our software presents certain software qualities such as robustness through no action events, security only as a basic username-password and logout implementation, usability through modest consideration to Normans' 7 usability principles, scalability in the form of modular design and maintainability by adhering to a static software stack not requiring changes until later deprecation. Though we acknowledge that a more complete version of our work would address the gaps in these qualities based on our discussion.

In Section 6.0 we consider the initially proposed functional and nonfunctional requirements and note that we had satisfied all nonfunctional requirements, while partially satisfying functional requirements. Thus, our preference to have also satisfied the ability to track and display user metrics would have added to the value of our project in the perspective of the user who wishes to closely track developer contributions. On our team's behalf, software stack changes are required in order to provide a better product and assist with increasing the pace of development by simplifying our approach and learning curve.

We come to the conclusion that the project has been met with a partial success due the rudimentary and working web application delivered, which presents a set of improvements as outlined in the project brief. While we have swapped out the improvement detailing the ability to track user contribution with the ability to subdivide projects into sprint intervals, this itself is a unique replacement feature. While the team will not be implementing any further changes based on our previous limitations after the conclusion of this semester, we will undoubtedly take forth the lessons learned from the perspective of team management and that of project development itself. This has yielded the benefit of allowing the team members each to arrive at a greater level of experience and be able to reflect up and pragmatically apply these lessons to our future endeavours.

8.0 Appendix

a. User Guide Contents

Technical Guide Contents:

b. Code Sample

Here we have chosen a subset of our project code as a sample in order to demonstrate project requirements being fulfilled. We provide the code, then explain as follows:

```
// BasicDateTimePicker Module
// get date-time and DateTime picker widget imports
import React, { useState, useEffect } from 'react';
import { AdapterDayjs } from '@mui/x-date-pickers/AdapterDayjs';
import { LocalizationProvider } from '@mui/x-date-pickers/LocalizationProvider';
import { renderTimeViewClock } from '@mui/x-date-pickers/timeViewRenderers';
import { MobileDateTimePicker } from '@mui/x-date-pickers';
import dayjs, { Dayjs } from 'dayjs';
import 'dayjs/locale/en-au';

// Standardised interface of date time to send to for historical data retrieval
interface DateTimeProps {
    placeholderText?: string;
    onClose?: (newValue?: Date) => void;
    onChange?: (newValue?: Date) => void;
    defaultValue?: Date;
}

export function BasicDateTimePicker({ placeholderText, onClose, defaultValue }: DateTimeProps) {
    const [value, setValue] = useState<Dayjs | null>(dayjs(defaultValue));
    const [windowClosed, setwindowClosed] = useState(false);

    useEffect(() => {
        if (!windowClosed) {
            return;
        }
        const timeout = setTimeout(() => {
            if (value && onClose) onClose(value.toDate());
            setwindowClosed(false);
        }, 1);
        // Give it a time out to make sure when user click on cancel, mui have enough time to
        // change the date back to previous
        return () => clearTimeout(timeout);
```

```
    }, [windowClosed, value, onClose]);
        // Run above code immediately after windowsClosed/value/onClose updated
    return (
        <LocalizationProvider dateAdapter={AdapterDayjs} adapterLocale="en-au">
            <MobileDateTimePicker
                onClose={() => {
                    if (onClose) {
                        setwindowClosed(true);
                    }
                }}
                label={placeholderText}
                value={value}
                viewRenderers={{
                    hours: renderTimeViewClock,
                    minutes: renderTimeViewClock,
                    seconds: renderTimeViewClock,
                }}
                slotProps={{ textField: { size: 'small' } }}
                onChange={(newValue) => {
                    setValue(newValue);
                    setwindowClosed(false);
                }}
                maxDateTime={dayjs()}
                closeOnSelect={false}
            />
        </LocalizationProvider>
    );
}
```

```
// BoardHistoryModule
import React from 'react';
import '../styles.scss';
import { KanbanBoard } from '@/features/KanbanBoard';

// Allow Kanban Board to only view ticket data.
export const BoardHistory = () => {
    return <KanbanBoard viewOnly={true} />;
    // set KanbanBoard in view only (history) mode
};
```

```
// KanbanBoard Module
// … code excerpt (larger module)
export const KanbanBoard = ({ viewOnly = false }: KanbanBoardProps) => {
    const informationSlice = useAppSelector((state) => state.information);
    if (!informationSlice.currentProject.projectId) return <></>;
    const [isLoading, setIsLoading] = useState<boolean>(false);
    const [tickets, setTickets] = useState<FullTicket[]>([]);
    const [boardHistoryTime, setBoardHistoryTime] = useState<Date>(new Date());
    const [sprintInfo, setSprintInfo] = useState<FullSprint>({});

    const [getAllTicketsForActiveSprintQuery, _currentSprintTicketsResult, _lastPromiseInfo] =
        useLazyGetAllTicketsForActiveSprintQuery();
    const [getAllTicketsForhistory, _historyTicketsResult, _lastHistoryPromiseInfo] =
        useLazyGetBoardAtTimeQuery();
    const [GetActiveSprint, _sprintInfoResult, _lastSprintPromiseInfo] =
useLazyGetActiveSprintQuery();

    const [updateTicketPost, { isLoading: _isLoading }] = useUpdateTicketMutation();
    const [columns, setColumns] = useState<kanbanBoardColumn[]>([]);

    useEffect(() => {
        if (isLoading || !tickets) return;
        setColumns([
            { id: 'TODO', name: 'To Do', tickets: filterByColumn(tickets, 'TODO') },
            { id: 'PROGRESS', name: 'Progress', tickets: filterByColumn(tickets, 'PROGRESS') },
            { id: 'COMPLETED', name: 'Completed', tickets: filterByColumn(tickets,
'COMPLETED') },
            { id: 'ACCEPTED', name: 'Accepted', tickets: filterByColumn(tickets, 'ACCEPTED') },
        ]);
    }, [tickets]);
    // Load all the tickets
    useEffect(() => {
        if (!informationSlice.currentProject.projectId) return;
```

```
        setIsLoading(true);
        if (viewOnly) {
            // View only (history) mode, which is used in History
            void getAllTicketsForhistory({
                projectId: informationSlice.currentProject.projectId,
                atUtcDatetime: boardHistoryTime.toUTCString(),
            }).then((data) => {
                setTickets(data.data as FullTicket[]);
                setIsLoading(false);
            });
        } else {
            // Normal mode, which is used in board page
            void getAllTicketsForActiveSprintQuery({
                projectId: informationSlice.currentProject.projectId,
            }).then((data) => {
                setTickets(data.data as FullTicket[]);
                setIsLoading(false);
            });
        }
    }, [boardHistoryTime, sprintInfo]);
    // Run code above immediately once board time (History) or sprint (Plan) changed

    useEffect(() => {
        if (!informationSlice.currentProject.projectId) return;
        void GetActiveSprint({
            projectId: informationSlice.currentProject.projectId,
        })
            .unwrap()
            .then((data) => setSprintInfo(data));
    }, [informationSlice.currentProject.projectId])
    // Run code above immediately once project have changed
```

The date-time widget allows for selection of date-time to shuttle to the backend for later retrieval, such that when calling upon the board history module, the kanban board module can take the view only flag and then take a slice of the board information (related to current project and sprint) and display it in the standard fashion that the regular board display would do, except that we have invoked the view only functionality in KanbanBoard. Now tickets in this information slice are displayed and presented statically and cannot be moved/interacted with.

c. Attacks

Attacks can be run by first brute forcing username through the sign up page:

Brute Force:

Automate the process of entering in usernames from $\sum^{*}\{alphanum\}$ (Maheshwari et al., 2019, p.31). This defines the process of testing all combinations of inputs from alphanumeric characters. Then scraping feedback until “User already exists” is found.

Upon determining this another brute force of password from $\sum^{*}\{alphanum\}$ with the given username can be performed until successful login. However, do note that since this application is not truly online collaborative, this has to be performed on a specific machine which requires direct access. Thus it is not likely to occur. Instead of a pure brute force password tester, one could utilise a password database to try more likely options in a more reasonable amount of time.

SQL Injection:

This process can be found readily both online and through academic materials. It involves within simple unguarded systems running the following into the username or password box:

```
Select id from users where username='username' and
password='password' or 1=1--+
```

(Authentication Bypass Using SQL Injection on Login Page, 2020)

9.0 References

Beltrão, A. C., de Souza, B. P., Santos, P. S. M., & Travassos, G. H. (2018). On the benefits and challenges of using kanban in software engineering: A structured synthesis study. Journal of Software Engineering Research and Development, 6(1), 11. https://jserd.springeropen.com/articles/10.1186/s40411-018-0057-1

GeeksforGeeks. (2020, November 14). Authentication Bypass using SQL Injection on Login Page. https://www.geeksforgeeks.org/AUTHENTICATION-BYPASS-USING-SQL-INJECTION-ON-LOGIN-PAGE/

Hiranabe, K. (2008). Kanban applied to software development: From agile to lean. Retrieved from http://www.infoq.com/articles/hiranabe-lean-agile-kanban

Ikonen, M., Abrahamsson, P., Fagerholm, F., Kettunen, P., & Pirinen, E. (2011). On the impact of kanban on software project work: An empirical case study investigation. In 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) (pp. 3-10). IEEE. https://ieeexplore.ieee.org/document/5773404

Kamal, F (2020). Literature Survey on KANBAN: Opportunities and Challenges. https://www.researchgate.net/publication/347586912_Literature_Survey_on_KANBAN_Opportunities_and_Challenges

Kniberg, H. (2009). Kanban vs. Scrum: How to make the most of both. Retrieved from http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf

Lay, W. (2022). Notes On Norman's Principles: [PDF Notes]. Provided by Monash University. FIT3175 Usability.

Llorens, B., & Viñoles-Cebolla, R. (2020). The influence of the use of project management tools and techniques on the achieved success. In Proceedings of the International Conference on Project Management (ICPM 2020), Lecture Notes in Management Science (pp. 157-165). Springer. https://link.springer.com/chapter/10.1007/978-3-030-54410-2_12

Maheshwari, A., Smid, M., & Canada, O. (2019). Introduction to Theory of Computation. https://cglab.ca/~michiel/TheoryOfComputation/TheoryOfComputation.pdf

Pandey, A. (2020, April 26). 7 Best Project Management Tools. Gale Business Insights: Essentials. Retrieved from https://link.gale.com/apps/doc/A622079509/CDB?u=monash&sid=bookmark-CDB&xid=605bd8a0