Project Plan

IBM Kanban Board

Group 31

Name	Year
Aoife Marilla Simm	3rd year
Pavel Petrukhin	3rd year
Merlin Prasad	3rd year
Peadar Kenny	3rd year
Leah O'Connor	2nd year
Yongjia Li	2nd year
Wen Geng Lin	2nd year

Table of Contents

1. Project Goals and Objectives

1.1. Background

Our project is to develop a Kanban board. Kanban is the Japanese word for "*signboard*" or "*billboard*" and is an agile project management tool. Our client has highlighted the importance of the use of the CI/CD pipeline in the development of our project.

Our client for this project is IBM, a multinational technology company. IBM is known for their development of software, hardware and hosting services. In addition to this, IBM is heavily involved in the funding and facilitation of research projects, evidenced by them being awarded the most US patents for 28 years running (as of 2020).

1.2. Objectives

Our clients wish for us to focus on gaining experience with various industry-standard practices and technologies rather than creating a polished end product. Our group was given a set of tasks to complete, e.g., implement a multiservices design, implement a persistent backend, use of Continuous Integration and Continuous Delivery (CI/CD) pipeline, etc., and was given a suggestion for a potential and generic project idea that would allow us to incorporate all those elements. Our group selected the project idea that our client suggested to us; a Kanban board.

Our clients have stressed in our meetings that they value us taking our time and being thorough with our research into potential technologies to instrument over us spending our time writing reams of code and creating numerous features.

1.3. Goals

- Use a microservices design process that implements at least 2 application components.
- Learn about the industry standard software development process.
- Create a persistent database backend that can be restored onto a new system.
- Implement a CI/CD pipeline for project development.

- Use a public repository with an MIT licence.

- Create a thorough README.

2. Project Scope

2.1. Project Deliverables

The deliverables for this project consist of the submission of our code bundle which will contain our final product and all the documentation we completed as part of the module.

Code base:

The code for the kanban board will consist of 3 main parts; A frontend based on the React framework, a REST API written using Express and node.js, and a persistent MongoDB database. We plan to implement a CI/CD pipeline with container technology based on RedHat that covers unit-testing, code scanning and static analysis.

Documentation:

Along with the reports we are required to submit to Blackboard for this module our client, IBM, has requested us to provide some additional documentation of the project. This includes keeping an up-to-date README of the project as well as publishing a final article on LinkedIn and Github Pages with details about the project.

Completed Deliverables:

- Signed requirements documentation (Appendix 8.1.)
- Software Design Specification Document (Appendix 8.2)
- Project Plan
- Well documented README

Deliverables to be completed:

- Development Report
- Management Report
- Individual Reflective Essay
- Github page
- LinkedIn article

2.2. Project Boundaries

An important aim of this project is to gain familiarity with good development practices and technologies as well as gaining experience with following the CI/CD development process. Using this for guidance we laid out the following project boundaries to define what would be in-scope and out-of-scope.

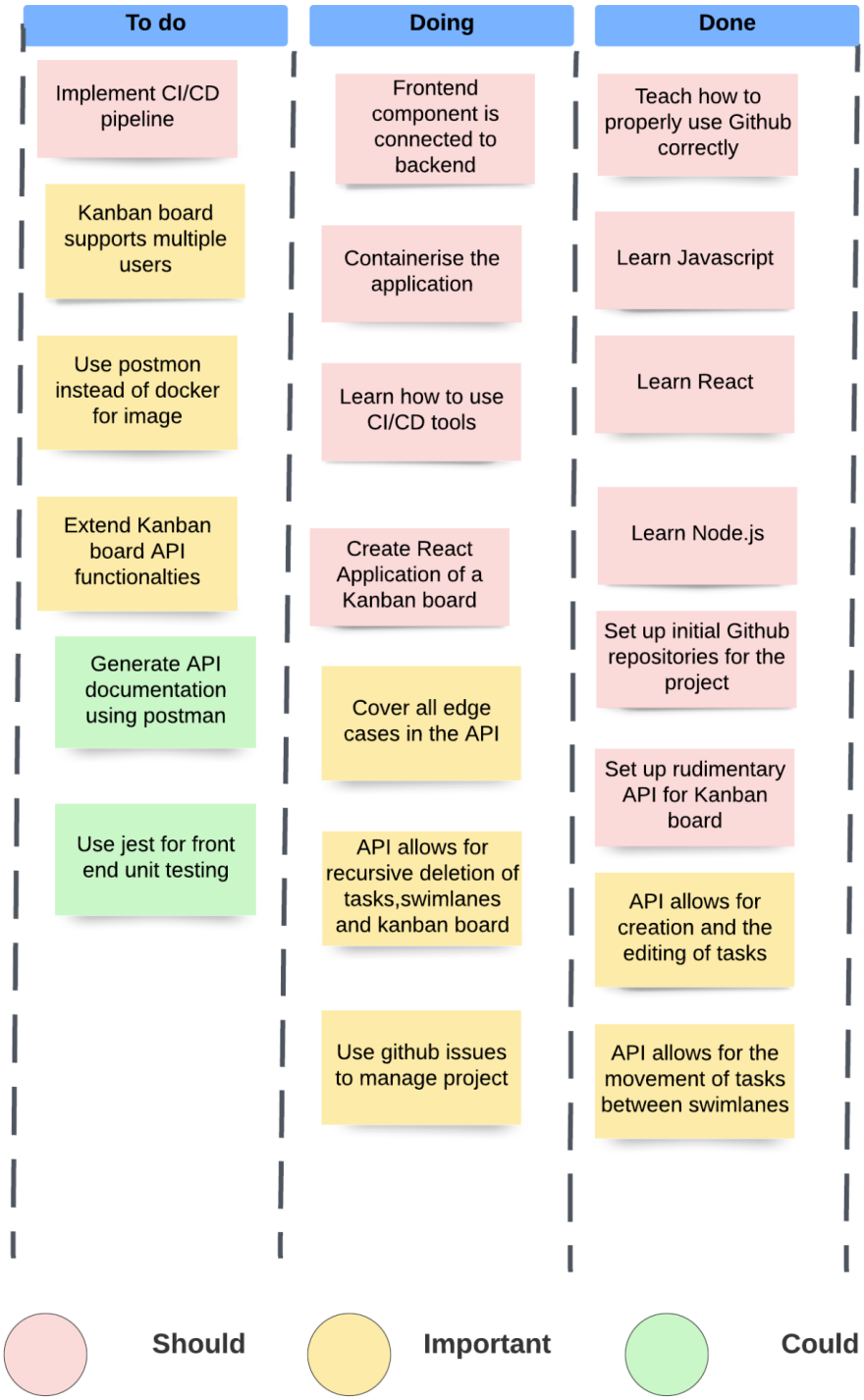
In-Scope:

- Develop a Kanban board API
- Visualise the Kanban board by querying the API
- Ensure the Kanban supports all basic functionalities required (create new boards, delete boards, update boards, move tasks between swimlanes and edit swimlanes)
- Implementing a CI/CD pipeline using container technology based on Red Hat
- Carry out unit-testing
- Documentation of all aspects of the project
- Complete the essential functional and non-functional requirements set out at the beginning of the project

Out-of-Scope:

Given the short project deadline, completing some of the stretch goals outlined in the requirements document such as using Ansible for more automation of the project will be considered out-of-scope at this moment.

2.3. Product Backlog



3. Project Approach

Our task is to develop a Kanban Board application with a persistent database and use it as an example project to build a CI/CD pipeline. The main idea is to deliver a minimal functionality product right away so that we can build the pipeline around it. Once the pipeline is in place we can refine the application while benefiting from all perks that a CI/CD pipeline brings such as automated testing and deployment.

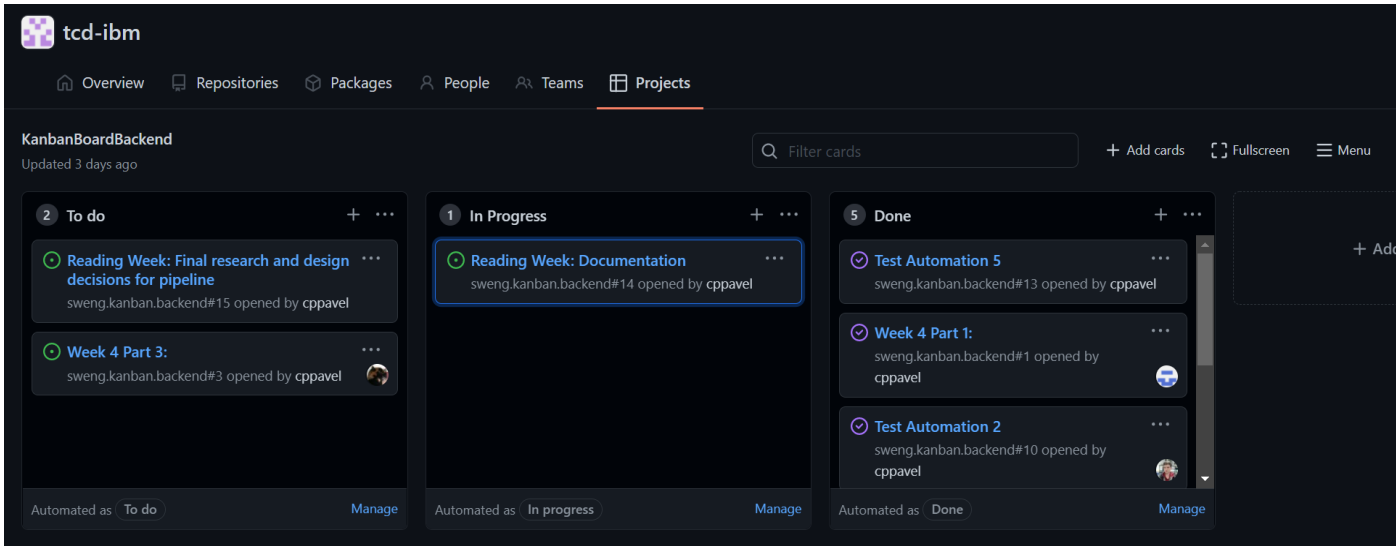
We split our team into 2 sub-teams, one for backend and one for frontend development. We made sure to split the teams based on the expertise and interests of the team members. We aim to achieve an agile process in our development.

3.1. Scrum Sprints

We used Github Projects to track all the issues in each sprint. For each issue we had an assignee and we ensured that the work was balanced evenly. We used automation provided by Github Actions to auto-assign issues to the relevant projects in Github. Here are the link to our project boards:

- https://github.com/orgs/tcd-ibm/projects/2 - Backend
- https://github.com/orgs/tcd-ibm/projects/1 - Frontend

Here is an example of what our Project Boards look like:



We decided to have week-long sprints, because our project is well-defined due to the amount of documentation and support we received from our clients in IBM. Having longer, two week sprints would have slowed down our pace because the project is easily split into small tasks that can be delivered in a short period of time. Hence there is more opportunity for rapid prototyping and feedback-driven development in a one week-long sprint scenario.

Normally sprint retrospectives happen at the end of each sprint, sometimes joined with sprint review meetings and even planning for the next sprint. In our case, we felt that sprint reviews and sprint planning is something that we will require to have on a weekly basis to adjust our development priorities in the agile backlog and to notice any issues that may be arising.

However, we chose the sprint retrospective, which focuses more on the individual reflections and suggestions for team operations, to happen once per two sprints. While this may seem unusual, similar schedules have been adopted in many successful tech companies. Firstly, having a sprint retrospective each sprint will inevitably lower the value of it, because it will be perceived more as a formality rather than an important time to reflect on the past performance. Secondly, the retrospectives which cover a week-long period may not be mature enough to create actionable items based on them. Finally, having less meetings allows the team to focus on the development more. Statistics have shown that many software developers find most of their meetings useless, which results in decreased productivity. Hence, sprint retrospectives are often called agile retrospectives to account for the fact that they do not necessarily happen at the end of each sprint.

Team Building and Tools Set-Up - Sprint 1
Objective: Meet together as a team, set up communication means, choose a project based on the expertise within the group and learning interests of team members. If we get the project early on, contact clients. Understand team's availability for meetings.

Start: 24th January 2022
End: 30th January 2022

Sprint Planning and Scrum: Tuesday 25th January 2022. We decided to have planning and scrum meeting at the same day, since our first stand up (scrum meeting) involved sharing our experiences and interests
Sprint review: Friday 28th January 2022
Sprint/Agile retrospective: N/A, since we are having one every 2 weeks, please see our rationale behind this in the above paragraphs.
Backlog refinements
Objectives achieved:

- Met together as a team
- Communication sorted out, all team members active (set up a Discord server and joined Slack with client)
- Project chosen
- Contacted client and set up a meeting

- Set up a doodle poll and found a number of times available for team meetings.

New objectives and their priority:

- Meet with client to learn more about the project and their perspective on it - Priority 1
- Look through the document provided by the client and do high-level introductory research on each of the technologies in preparation to the meeting - Priority 2
- Set up a github repo and teach 2nd years how to use git - Priority 3

Understanding Requirements and Choice of Technologies - Sprint 2

Objective: meet with the client and do research prior to that, capture requirements and start working on the requirements document, split the team into working groups, choose technologies and set up github repo

Start: 31st January 2022
End: 6th February 2022

Sprint Planning: Monday 31st January 2022
Scrum: Friday 4th February 2022
Sprint Review: Friday 4th February 2022
Sprint/Agile retrospective: Friday 4th February 2022

Backlog refinements

Objectives achieved:

- Met clients, discussed the project with them and requested more resources about the DevOps technologies involved. Understood that the main focus of the project is to build a pipeline, hence decided to build a prototype application as soon as possible and start building the pipeline around it
- Split up the tasks related to requirements document so that each team member contributes to it
- Split the team into 2 working groups: Frontend and Backend
- Had a training session with 2nd years about github, taught them basic development workflow, including branching
- Chose react js as the fronted technology for Kanban Board due to vast local expertise with this technology.
- Chose node js with express as a backend technology for Kanban Board, because it works well with react fronted and it can be easily built and deployed by almost any CI/CD tools available due to its popularity
- Chose MongoDB as the persistent database for the project, because it integrates well with express (mongoose library) and is easy to learn
- Chose to containerize the API and database to ensure better transition to pipeline at later stages

New Objectives and Their Priority Ranking:

- Teach React to 2nd years - Priority 1
- Teach Node JS, Express, mongoose and Docker to 2nd years - Priority 1
- Be on track to finish the requirements document before the next client meeting (15th February) to have it reviewed in time - Priority 2

- Set up github repos, as did not get a chance to do it within the previous sprint - Priority 3

Training and Documenting Requirements - Sprint 3
Objective: provide necessary training and examples for 2nd years and 3rd years to start working on the project, finish / nearly finish the requirements document by the end of the sprint and prepare the requirements presentation

Start: 7th February 2022
End: 13th February 2022

Sprint Planning: Monday 7th February 2022
Scrum: Monday 7th February 2022
Sprint Review: Friday 11th February 2022
Sprint/Agile Retrospective: N/A, since we are having one every 2 weeks, please see our rationale behind this in the above paragraphs.

Backlog refinements
Objectives achieved:

- Made a tutorial on basic HTML, CSS to 2nd years. Introduced concepts of JSX and states in React.
- Made a tutorial on Node JS, Express, MongoDB (mongoose) and Docker. Ensured that Node JS and Docker are installed on machines of each backend team member. Showed how to use Postman to query the API and generate the documentation for each endpoint.
- Created 2 repos for frontend and backend in the tcd/ibm organisation
- Finished most of the sections in the requirements document, with only a few diagrams left to be done
- Prepared a requirements presentation, since our team was chosen to present on the 14th of February
- Designed the basic API and database using Express and MongoDB, split up the rest of the work into 3 tasks. Met in-person on Friday as a backend working group to explain each of the 3 tasks and hence assign them.

New objectives and their priority:

- Finish diagrams in the requirements document before client meeting (15th February) and have the document reviewed and signed - Priority 1
- Complete 3 tasks for the backend development - Priority 2
- Continue React js training, design the components based on the data model and assign the development of each component to a team member - Priority 2

Developing an MVP and Finalising Requirements - Sprint 4
Objective: get the requirements document signed off and submit it, finish up the MVP for backend, provide more training on React JS, assign frontend tasks.

Start: 14th February 2022
End: 20th February 2022

Sprint Planning: Monday 14th February 2022

Scrum: Friday 18th February 2022
Sprint Review: Friday 18th February 2022
Sprint/Agile Retrospective: Friday 18th February 2022

Backlog refinements
Objectives achieved:

- Presented the requirements document to the client, took feedback into the account, got it signed off and submitted it in time.
- Completed 2 backend tasks out of the 3 planned, because the 3rd task was dependent on the first 2.
- Held another training session on React JS exploring how to split the UI into components more and learning about standard wrapper components that are used in most frontend projects such as cards.

New objectives and their priority:

- Finish the 3rd task in backend development - Priority 1
- Assign individual tasks to each developer in Fronted team - Priority 1
- Start researching more technologies for the CI/CD pipeline - Priority 2

Finalising an MVP and Starting the Detailed CI/CD Pipeline Research - Sprint 5
Objective: finalise the backend MVP and assign tasks for fronted team + develop a simple frontend prototype

Start: 21st February 2022
End: 27th February 2022

Sprint Planning: Monday 21st February 2022
Scrum: Monday 21st February 2022
Sprint Review: Friday 25th February 2022
Sprint/Agile Retrospective: N/A, since we are having one every 2 weeks, please see our rationale behind this in the above paragraphs.

Backlog refinements
Objectives achieved:

- Finished the 3rd task for the backend MVP (recursive deletion of kanban boards)
- Split up the frontend tasks into developing cards, designing swimlanes and API querying
- Got some understanding on how the pipeline will work, split up the pipeline research among the backend team members and created corresponding issues on Github. Tried out Jenkins with OpenShift Developer Platform

New objectives and their priority:

- Get feedback from client regarding pipeline research - Priority 1
- Develop a simple frontend prototype - Priority 2
- Finalise the pipeline architecture, make a list of technologies for each stage of the pipeline - Priority 2

CI/CD Pipeline Architecture and Finalising Frontend Prototype - Sprint 6
Objective: come up with a clear architecture for the CI/CD pipeline, taking the clients' feedback into account, make a list of technologies for each step of the pipeline: repo monitoring and image building including running tests and static analyses, image storage and application deployment, finalise the fronted prototype. Showcase our progress to Macu/Stephen.

Start: 28th February 2022
End: 6th March 2022

Sprint Planning: Monday 28th February 2022
Scrum: Monday 28th February 2022
Sprint Review: Friday 4th March 2022
Sprint/Agile Retrospective: Friday 4th March 2022

Backlog refinements
Objectives achieved:

- Got feedback on the pipeline architecture from the client's who suggested that we should explore more different technologies to choose ones that fit our needs best. Client also told us that some of the requirements from the initial project documents may be replaced with alternative technologies, however which should still focus on Red Hat tools.
- The backend team came up with an architecture of the pipeline, which cleared up most of the confusion that was there. Technologies for each step of the pipeline were identified.
- The frontend team created a simple UI prototype
- The team successfully showcased the progress to Macu

New objectives and their priority:

- Document our work in Software Design Specification and Project Plan during reading week - Priority 1
- Research technology options for each step in the pipeline - Priority 2
- Start connecting frontend to the API - Priority 3
- Fix small bugs in the API and test it a bit more - Priority 3

Documentation, Options for Pipeline Stages and a Little Bit of Development - Sprint 7
Objective: have the Project plan and Design specification ready at the end of the sprint, have a list of technologies for the pipeline with their advantages/disadvantages and identify top 3 technologies for each stage. If time allows, make the choice for each of the stages and document the design decisions. Start connecting the frontend to the API and fix small bugs in the API.

Start: 7th March 2022
End: 13th March 2022

Sprint Planning: Monday 7th March 2022
Scrum: Monday 7th March 2022 - did not record as it was not a requirement
Sprint Review: Sunday 13th March 2022
Sprint/Agile Retrospective: N/A, since we are having one every 2 weeks, please see our rationale behind this in the above paragraphs.

Backlog refinements
Objectives achieved:

- Finished the Project Plan and Design Specification documents
- Came up with a list of technologies for pipeline stages and identified potential best options for each stage
- Tested the API more and fixed a few bugs
- Frontend team researched API documentation
- Set up Github Actions automation to assign the issues to relevant projects on Github, hence no need to do it manually anymore

New objectives and their priority:

- Make final technology choices for the pipeline after getting feedback from client and start building the CI/CD pipeline - Priority 1
- Continue connecting frontend and backend - Priority 2

Start Building the Pipeline and Integrating Backend with Frontend - Sprint 8
Objective: Make final decisions on the pipeline technologies and start building it, work towards connecting frontend and backend as it will be important for demonstrations

Start: 14th March 2022
End: 20th March 2022

Planned schedule:

Sprint Planning: Monday 14th March 2022
Scrum: Wednesday 16th March 2022
Sprint Review: Friday 18th March 2022
Sprint/Agile Retrospective: Friday 18th March 2022
Backlog refinements - N/A since the sprint has not happened yet

Building the Core Part of the Pipeline, Main Functions in Backend and Frontend Connected - Sprint 9
Objective: Build the core of the pipeline, with only static analysis and automated testing left, have all the main functions of backend connected with frontend

Start: 21th March 2022
End: 27th March 2022

Planned schedule:

Sprint Planning: Monday 21th March 2022
Scrum: Wednesday 23th March 2022
Sprint Review: Friday 25th March 2022
Sprint/Agile Retrospective: N/A, since we are having one every 2 weeks, please see our rationale behind this in the above paragraphs.
Backlog refinements - N/A since the sprint has not happened yet

Adding Static Analysis and Automated Testing, Fully Connecting Frontend and Backend- Sprint 10

Objective: Add static analysis and automated testing to the pipeline, have basic frontend and backend fully connected

Start: 28th March 2022
End: 3rd April 2022

Planned schedule:

Sprint planning: Monday 28th March 2022
Scrum: Wednesday 30th March 2022
Sprint review: Friday 1st April 2022
Sprint/Agile retrospective:Friday 1st April 2022
Backlog refinements - N/A since the sprint has not happened yet

Adding More Functionality to the Kanban Board System - Sprint 11
Objective: Add more functionality to the Kanban Board system, improve test coverage

Start: 4th April 2022
End: 10th April 2022

Planned schedule:

Sprint planning: Monday 4th April 2022
Scrum: Wednesday 6th April 2022
Sprint review: Friday 8th April 2022
Sprint/Agile Retrospective: N/A, since we are having one every 2 weeks, please see our rationale behind this in the above paragraphs.
Backlog refinements - N/A since the sprint has not happened yet

Finalise Project Submission, Client Handover - Sprint 12
Objective: Finalise the project submission, handover the project to the client

Start: 11th April 2022
End: 17th April 2022

Planned schedule:

Sprint planning: Monday 11th April 2022
Scrum: Wednesday 13th April 2022
Sprint review: Friday 15th April 2022
Sprint/Agile retrospective: Friday 15th April 2022
Backlog refinements - N/A since the sprint has not happened yet

3.2. Agile Retrospective

Sprints 1-2

How did the last 2 sprints go for you? (😃, 😐, 😱, 😬)

- Wen Geng - 😃
- Yongjia - 😃
- Leah - 😃
- Peadar - 😃
- Merlin - 😃
- Aoife - 😃
- Pavel - 😃

What went well?

- Wen Geng - received our project, looking forward to getting started
- Yongjia - get the project, and meet each other
- Leah - met with everyone and agreed on a project together.
- Peadar - being assigned one of our top choice projects
- Merlin - set up communication with the whole team and picked a cool project.
- Aoife - Introductions and getting a sense of our team
- Pavel - met with the team, chose an interesting project

What did not go well?

- Wen Geng - had no experience with javascript
- Yongjia - no idea what react is
- Leah - No prior experience with React
- Peadar - Took too long to teach react basics
- Merlin - setting up a group meeting time slot due to different availabilities
- Aoife - Large knowledge gap with project content
- Pavel - took some time to connect as a team

What will you do better as an individual?

- Wen Geng - learn javascript basics
- Yongjia - install react, and start learning it
- Leah- start learning basics of React
- Peadar - streamline learning process for second years
- Merlin - figure out people's timetable to co-ordinate set meeting times
- Aoife - More in-depth initial research
- Pavel - communicate my ideas more, arrange more meetings and be proactive

What should we do better as a team (your suggestions)?

- Wen Geng - all looks good
- Yongjia - all good
- Leah - set regular meeting time that suits all
- Peadar - all good so far

- Merlin - all good so far
- Aoife - More documentation
- Pavel - document our decisions more

Sprints 3-4

How did the last 2 sprints go for you? (😃, 😐, 😱, 😬)

- Wen Geng - 😃
- Yongjia - 😃
- Leah - 😃
- Peadar - 😃
- Merlin - 😃
- Aoife - 😃
- Pavel - 😃

What went well?

- Wen Geng - first experience with Git and Javascript
- Yongjia - finish doc and start coding with react
- Leah - finishing requirements document & presentation, starting react components
- Peadar - Began teaching component based approach for react
- Merlin - was able to contribute a lot in building the API.
- Aoife - Creation and collaboration of Requirements document and presentation
- Pavel - did a lot of training with the backend working group, started writing code, everyone contributes equally

What did not go well?

- Wen Geng - Javascript
- Yongjia - put stuff as a statistic website
- Leah - took a bit of time to understand react, but got there in the end
- Peadar - Lots of information for second years to digest
- Merlin - did not do extensive error checking of code due to time constraints
- Aoife - Couldn't complete task 3 due to dependencies on prior tasks
- Pavel - did not finish task 3 for the backend, because it was blocked by tasks 1 and 2

What will you do better as an individual?

- Wen Geng - Get my tasks finished faster as not to block the progress of other team members
- Yongjia - add more cards to swimline
- Leah - try to add more features to card component than just the basics
- Peadar - be clearer in my communication
- Merlin - do more research on how to use the technologies required before pair programming sessions so I can provide more assistance.
- Aoife - Schedule my time to avoid lag caused by dependencies
- Pavel - give dependencies between the tasks more attention when splitting up the

work and planning pair programming sessions

What should we do better as a team (your suggestions)?

- Wen Geng - everything is going well
- Yongjia - good
- Leah - document as we go
- Peadar - Documenting as we go
- Merlin - start documenting earlier and have a more detailed project ReadMe.
- Aoife - All good
- Pavel - all good so far

Sprints 5-6

How did the last 2 sprints go for you? (😃, 😐, 😱, 😬)

- Wen Geng - 😃
- Yongjia - 😃
- Leah - 😃
- Peadar - 😃
- Merlin - 😃
- Aoife - 😃
- Pavel - 😃

What went well?

- Wen Geng - research on the CI/CD pipeline was completed
- Yongjia - completed project plan
- Leah - everyone completed their parts of project plan and design specification document
- Peadar - Initial MVP frontend completed with dummy data
- Merlin - the project plan and design specification documentation were completed with everyone contributing.
- Aoife - reducing knowledge gap though research and paired programming, completed task
- Pavel - CI/CD pipeline saw good progress, clear architecture defined

What did not go well?

- Wen Geng - still unsure on some elements of the CI/CD pipeline
- Yongjia - learning flexbox to add in more components and auto testing
- Leah - trying to test frontend components using jest
- Peadar - Lacking in testing infrastructure
- Merlin - my lack of experience with CI/CD development means I need to do more research .
- Aoife - Large volume of work to be completed to progress
- Pavel - initially pushed the team towards Jenkins, which turned out to be not the best option

What will you do better as an individual?

- Wen Geng - conduct more research on the CI/CD pipeline
- Yongjia - take a closer look at components in react and auto testing
- Leah - conduct more research into how to use jest
- Peadar - keeping on top of obligations for myself and second years
- Merlin - try to research more on Redhat technologies required for CI/CD pipeline
- Aoife - ask more questions to already experience group members/mentors to speed up researching and understanding process
- Pavel - prioritise research more

What should we do better as a team (your suggestions)?

- Wen Geng
- Yongjia - all good
- Leah - all good so far
- Peadar - Happy with progress
- Merlin - happy with the progress
- Aoife - All good
- Pavel - frontend and backend should start having integration meetings

Sprints 7-8 - Not yet completed the sprints

How did the last 2 sprints go for you? (😃, 😐, 😱, 😬)

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What went well?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What did not go well?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin

- Aoife
- Pavel

What will you do better as an individual?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What should we do better as a team (your suggestions)?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

Sprints 9-10 - Not yet completed the sprints

How did the last 2 sprints go for you? (😃, 😐, 😱, 😬)

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What went well?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What did not go well?

- Wen Geng
- Yongjia
- Leah

- Peadar
- Merlin
- Aoife
- Pavel

What will you do better as an individual?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What should we do better as a team (your suggestions)?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

Sprints 11-12 - Not yet completed the sprints

How did the last 2 sprints go for you? (😃, 😐, 😱, 😬)

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What went well?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What did not go well?

- Wen Geng

- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What will you do better as an individual?

- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

What should we do better as a team (your suggestions)?

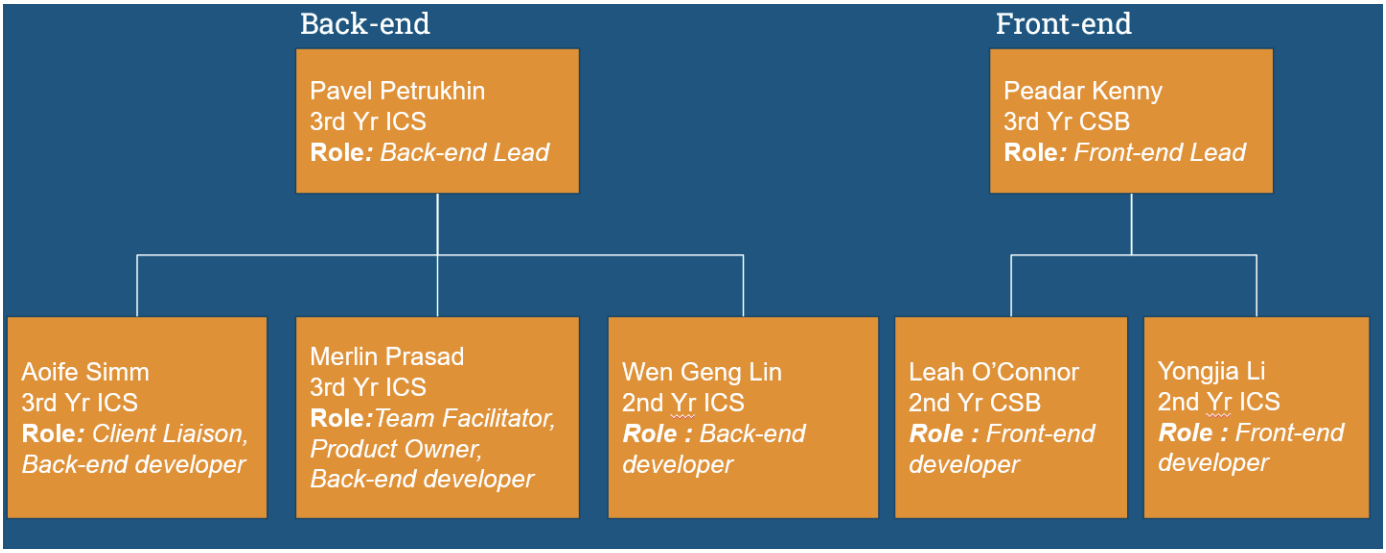
- Wen Geng
- Yongjia
- Leah
- Peadar
- Merlin
- Aoife
- Pavel

4. Project Organisation

4.1. Staff

Name	Projects	Skills
Peadar Kenny	SWENG 2021, Co-founder, Director, Project manager: Entity Esports, Summer internship: Mapall	React, Javascript, Python, Java
Leah O'Connor	Programming Project	Java, C
Yongjia Li	Programming Project	Java, C++, Python
Pavel Petrukhin	Programming Project, SWENG 2021, TCD CS Scholar, Google STEP intern	Java, Javascript, Python, C++, Devops, infrastructure
Merlin Prasad	Programming Project, SWENG 2021	Java, Python, C++, Javascript, Bootstrap
Wen Lin	Programming Project	Java, Python, C
Aoife Simm	Programming Project, SWENG 2021	Java, Python, C

4.2. Staff Chart



Sprint 1-2

Name	Role	Working On
Pavel	Product Owner	Requirements document, designate group hierarchy.
Merlin	Scrum Master	Requirements document, designate group hierarchy.
Aoife	Product Owner	Requirements document
Peadar	Product Owner	Requirements document
Leah	Member	Requirements document
Wen Lin	Member	Requirements document
Yongjia	Member	Requirements document

Sprint 3-4

Name	Role	Working On
Pavel	Product Owner	Assigning backend work
Merlin	Product Owner	Working on Kanban API
Aoife	Product Owner	Working on Kanban API
Peadar	Scrum Master	Assigning frontend work
Leah	Member	Working on Card component
Wen Lin	Member	Working on Kanban API
Yongjia	Member	Working on Swimlane component

Sprint 5-6

Name	Role	Working On
Pavel	Product Owner	Researching CI/CD Pipeline for backend. Project Plan
Merlin	Product Owner	Researching CI/CD pipeline technologies. Project Plan
Aoife	Scrum Master	Research CI/CD pipeline technologies. Project Plan
Peadar	Product Owner	Researching CI/CD Pipeline for frontend. Project Plan
Leah	Member	Documenting components and researching JEST. Project Plan
Wen Lin	Member	Documenting components and researching JEST. Project Plan
Yongjia	Member	Research CI/CD pipeline technologies. Project Plan

5. Risk Analysis

5.1. Risk Analysis

Risk Element	Impact (1 to 5)	Likelihood (1 to 5)	Risk Factor (I*L)
Unable to connect front-end to backend	4	1	4
Not implementing CI/CD pipeline	3	2	6
Insufficient project documentation	3	1	3
Mismanagement of the team	5	1	5
Miscommunication between the team and IBM	3	2	6

5.2. Risk Mitigation

Risk	Measures to Reduce Risk
Miscommunication between the team and IBM	We are scheduled to have biweekly meetings with our Client IBM since the project started. In these meetings we show them the progress of our project as well as get feedback. We also have a slack channel set up with IBM and their emails so we can ask them any questions we have and receive clarification quickly which helps prevent miscommunication.
Not implementing CI/CD pipeline	Extensively research CI/CD pipelines and follow online tutorials on how to use common technologies required to implement it. The back-end team is also focusing on having the pipeline in place early in the development stage of the project . Back-end team also asked for help from IBM in finding good resources to help implement the pipeline and also looked into their suggestions.
Mismanagement of the team	The 3rd years are split up into roles so that they can provide the best guidance to the 2nd years. Weekly scrum meetings also help mitigate this risk as we can update each other on our progress. Github issues are used by the team to keep track of the project .
Unable to connect front-end to back-end	Ensure clear communication between front-end and back-end teams regarding the type of data being sent and received. The front-end team will test the API designed by the back-end as soon as they can in order to find problems quickly and address them.
Insufficient project documentation	The team keeps an up to date Github ReadMe with all the technologies used and the architecture of the project. Code will be well commented and commits will be clear and consistent through the project development . Reports will be worked in conjunction with the project. We will also publish on Github pages as well as on LinkedIn once the project is finished.

6. Project Controls

Factor	Control Method
Progress	- Use of Github projects to view progress and assign issues. - Weekly meeting to discuss the project and what is due for each week.
Quality	- Discussion of what was done weekly. - Checking and critiquing others work within documentation and code.
Deliverables	- Leads assign parts of a document to each team member including themselves and take input from others on what part they believe they would be able to perform best in.
Deadlines	- Leads set deadlines for different project points which are also enforced by the managers.
Communication	- General communication and chat is done through discord. - Team calls are done using discord and google meet, calls with clients take place via Webex, while demonstrator meetings are done via blackboard. - Weekly team meetings and weekly meetings with our demonstrator.

7. Communication

7.1. Client Communication

We have two clients in IBM for this project; Mihai Criveti and Panpan Lin. Our group assigned one member the role of client liaison and they initiated contact with our client through email with all members and demonstrator, Alex Randles, cc'd. We arranged a bimonthly client meeting for Tuesday 5-6pm which is hosted by our clients on Webex. Additionally, our client invited our group members to a Slack group for less formal communication.

Client Meetings	
Week and Date	**Item(s) discussed**
01/02	- General project overview and discussion - Setting expectations, requirements, stretch goals, etc.
15/02	- Presentation to client of requirements document - Gathering feedback
01/03	- "Scrum"-like meeting

7.2. Project Team Meetings

Our group tends to have at least one general meeting a week to discuss the overall project and record scrum videos. The back- and frontend groups meet separately at least once a week. Our back- and frontend leads have hosted many paired programming sessions. We have an active discord to facilitate arranging meetings and smaller queries. We also meet our demonstrator Alex Randles every Tuesday at 6.30pm to discuss our progress with the project and to gather feedback.

General/Scrum Meetings	
Week and Date	**Item(s) Discussed**
24/01 - 7pm	Project selection
25/01 - 6:30pm	Scrum video
31/01 - 6:30pm	Requirements presentation and document discussion
04/02 - 11am	Scrum video

07/02 - 6pm	Scrum video
14/02 - 6pm	Requirements presentation run-through
18/02 - 12pm	Scrum video
21/02 - 6pm	Scrum video
28/02 - 6pm	Scrum video

Backend Meetings	
Week and Date	**Item(s) Discussed**
04/02 - 12pm	Simple API example run through
11/02 - 12pm	Q&A and troubleshooting regarding the simple API example
16/02 - 2pm	Allocated tasks for the Kanban Board API
25/02 - 12pm	High-level overview of CI/CD discussion
04/03 - 1pm	Discussion of research (Red Hat Quay, Clair)

Frontend Meetings	
Week and Date	**Item(s) Discussed**
07/02 - 5pm	Introduction to React.JS and Javascript
18/02 - 5pm	Component based approach and Initial work assignment
21/02 - 5pm	Integrating components into frontend with Dummy data
28/02 - 5pm	Documentation and testing framework meeting

Paired/Group Programming Sessions	
Week and Date	**Item(s) Discussed**
16/02 - 2pm + 17/02 4pm	Backend - Defined core data models and routes for the Kanban API
24/02 - 4pm	Backend - Introduced recursive deletion for Kanban Boards and fixed asynchronous bug

8. Appendices

8.1. Requirements Document

Table of Contents

1. Introduction

1.1. Overview - Purpose of system

Our project is to develop a kanban board for our client IBM. Kanban is the Japanese word for "*signboard*" or "*billboard*". It is an agile project management tool. Our client has highlighted the importance of the use of the CI/CD pipeline in the development of our project.

1.2. Scope

The minimum viable product that our client defined for us is a Kanban board that supports multiple users that is deployed with the modern CI/CD pipeline with container technology based on Red Hat. They stressed the importance of following the CI/CD development process and gaining development experience over a highly polished product.

A major goal of the project is to gain exposure to the design principles commonly used in industry as well as experience using open source development tools and practises as well as such as Github and automated testing.

1.3. Objectives and Success Criteria

While our client defined some objectives and success criteria with us, they also stressed the fact that they want us to focus on the development process and on gaining experience working with the CI/CD pipeline over the production of a high-quality end product.

Objectives:

- Follow a microservices design process.
- Implement at least 2 application components. In this the case of our project, a persistent database and a front-end.

- Implement a persistent database backend that can be restored onto a new system.
- Follow CI/CD pipeline for project development.
- - The project needs to be a public repository with an MIT licence.
- A well documented READ-ME.

Success Criteria

- The creation of a Kanban board that supports multiple users .
- Gain experience with CI/CD pipeline.

1.4. Definitions, abbreviations

Application Programming Interface (API)

A set of definitions and protocols for building and integrating application software.

Continuous Integration/Continuous Delivery Pipeline (CI/CD Pipeline)

This is a process of ongoing automation and continuous monitoring throughout the lifecycle of a development project.

Microservices Design

This is an architectural approach to developing a single application as a suite of small services. Each of these services is running in its own process and communicating with other services through lightweight mechanisms, often HTTP resource API.

Container(isation)

This is the packaging of software code and OS libraries and dependencies required to run the code in the form of an executable. Containerisation is favourable over virtual machines as it is more portable and resource-efficient. It allows for faster and more secure deployment of applications. It additionally keeps the occurrence of some bugs and errors down.

Unit Test

A way of error checking code through automated tests performed on small segments of code.

1.5. References

Kanban Boards

https://www.atlassian.com/agile/kanban/boards

https://www.ibm.com/garage/method/practices/culture/practice-kanban-method/

https://www.ibm.com/blogs/cloud-archive/2016/12/yes-we-kanban/

https://docs.github.com/en/issues/organizing-your-work-with-project-boards/managing-project-boards/about-project-boards

https://help.trello.com/article/708-what-is-trello

CI/CD Pipeline

https://www.redhat.com/en/topics/devops/what-cicd-pipeline

2. Current system

There currently exists numerous Kanban board applications because it is an efficient and simple to use team management tool.

Physical Board

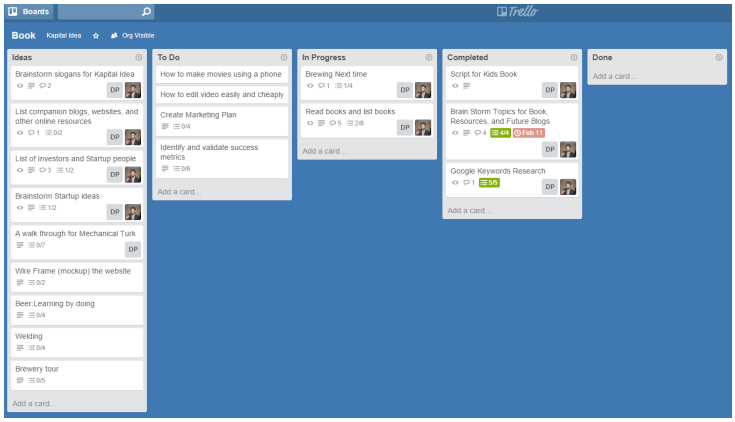
The easiest Kanban board involves using a physical board split into vertical columns. Using sticky notes and whiteboard teams can move around tasks as is necessary to track their progress.

Github Project Boards

Github allows users to create project boards consisting of issues, pull requests and notes in order to support agile scrum methodology and collaboration. Project boards come with a range of templates such as basic kanban and automated kanban with review which means the user can create as complex a kanban board as required.

Trello

One of the most popular Kanban board applications, it allows you to categorise your project into multiple boards. The advantage of systems like trello over traditional boards is that it allows you to be able to access it anywhere on the web. The ease of access means teams can be aware of tasks at all times. Trello lets the user add attachments, website links and pictures to their cards which allow for more customisation over physical boards as well.


Boards
Trello
Book
Kapital Idea
Org Visible
Ideas
Brainstorm slogans for Kapital Idea
List companion blogs, websites, and other online resources
List of Investors and Startup people
Brainstorm Startup ideas
A walk through for Mechanical Turk
Wire Frame (mockup) the website
Beer:Learning by doing
Welding
Brewery tour
Add a card...
To Do
How to make movies using a phone
How to edit video easily and cheaply
Create Marketing Plan
Identify and validate success metrics
Add a card...
In Progress
Brewing Next time
Read books and list books
Add a card...
Completed
Script for Kids Book
Brain Storm Topics for Book, Resources, and Future Blogs
Google Keywords Research
Add a card...
Done
Add a card...

3. Proposed System

3.1. Overview

We plan to develop the frontend and backend in a more independent way than that of a usual React + Express project. The reason for that is that we want to allow our API to be used by other clients. We have 3 major components: frontend based on React, REST API based on Express and a persistent MongoDB database.

The main plan is to develop a minimal version of the application first, so that we can focus on building automation and CI/CD pipelines after that. We plan to delegate some part of our time on improving the application itself once some pipeline is in place. This way we will be able to benefit from the project the most.

Most of the work is anticipated on the automation and infrastructure part of the project, which can be seen in more detail in the requirements section.

3.2. Functional Requirements

Functional requirements for our project are determined by the application used as an example for the CI/CD pipeline. Since we are developing a kanban board, there are a number of essential requirements. We plan to develop the API and client separately to make sure that our API can be used by other clients and/or via direct HTTP requests, hence it is reasonable to define front-end and back-end requirements.

Back-end requirements:

Essentials:

- The API should provide end points to access, create, update and delete kanban boards
- The API should allow the user to create new swim lanes within the kanban boards as well as create and move the tasks in these lanes

- Create documentation for the backend

Stretch goals:

- Implement user sessions so that users can register and login. This will require partitioning the database and implementing authentication
- Automatically generated documentation for the API.

Front-end requirements:

Essentials:

- Visualise the kanban boards by querying API
- Allow the user to create new boards, delete and update existing ones
- Allow the user to move the tasks from one swimlane to another either by drag and drop or using some other form of UI
- Kanban board supports multiple users.

Stretch goals:

- Have pages for registering and logging in
- Manage authentication data such as JWTs (JSON Web Token) in local storage on the client

3.3. Non-functional requirements

Non-functional requirements constitute the core of our project, because the focus is put on software development practises and automation.

Essentials:

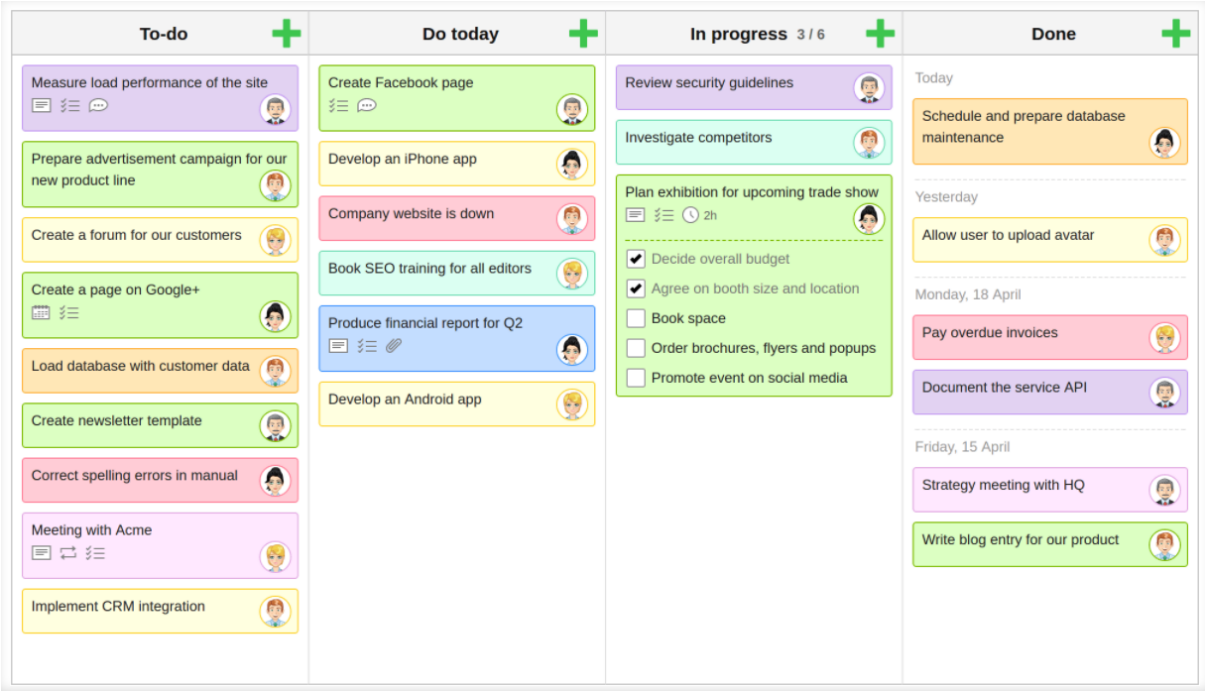
- The application we develop has to be containerised. In our case, we will have 2 containers for the database and the API. The images for containers have to be built using podman
- Application has to store persistent data that can be restored onto a new system
- Images have to be based on Red Hat Universal Base image and should take the up-to-date security updates into account
- We should build a CI/CD pipeline with static analysis, unit test coverage collection and code scanning. The pipeline has to be triggered automatically when the code changes, for example Github Actions
- Project code has to be hosted on Github
- We should also ensure that automated deployment is part of the pipeline. The suggested technology for that is OpenShift
- Use Red Hat Quay + Clair as the container registry
- We need to provide documentation in the form of markdown files, Github pages and publications on LinkedIn

Stretch goals:

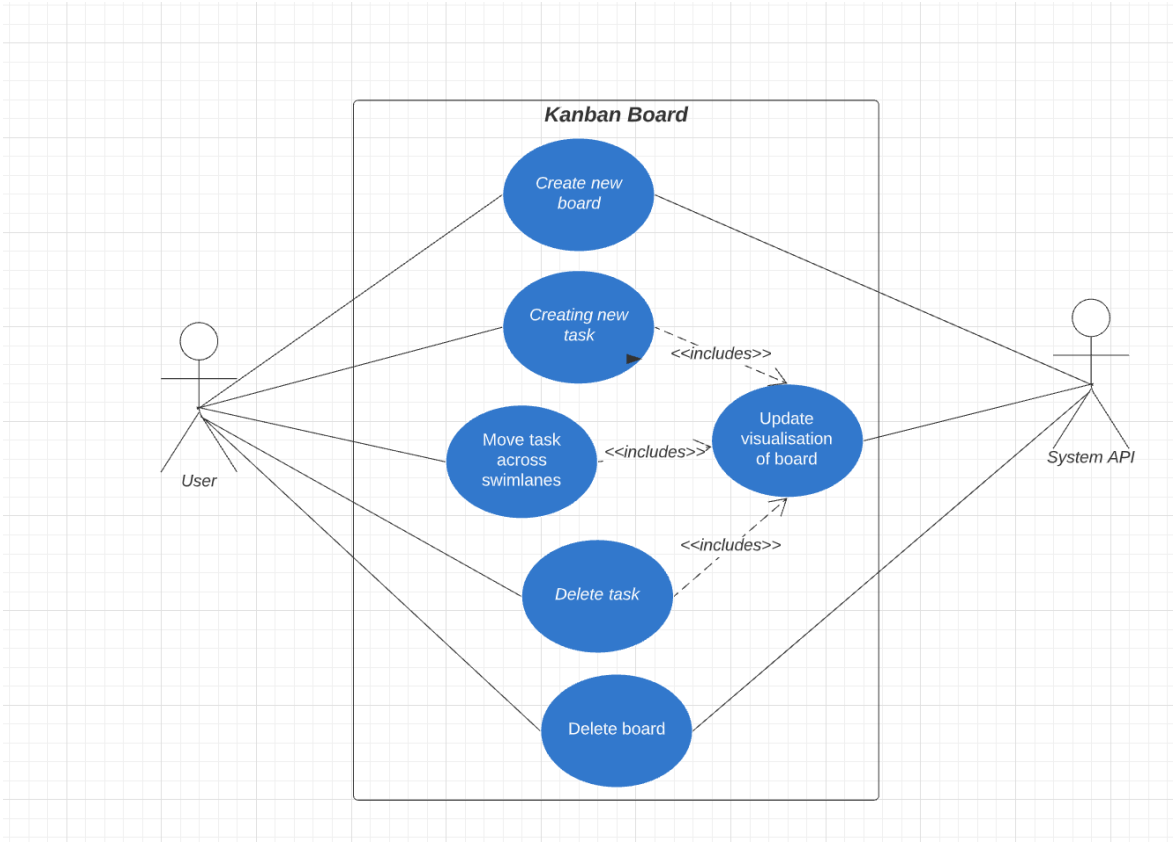
- Use Ansible for more automation
- Test of multiple different devices such Linux, Windows etc.

3.4. System prototype

3.4.1. User interface mockups



3.4.2. Use Cases



Name	Create new board
Participating Actor(s)	User & System API
Entry Condition	Application is accessible & option to create new board exists
Exit Condition	New blank Kanban board is created
Normal Scenario	- User selects 'create new board' - User gives new board a name - New blank Kanban board template created
Error Scenario	System unable to create new board

Name	Creating new task
Participating Actor(s)	User
Entry Condition	User has Kanban board in use
Exit Condition	Kanban board has new additional task
Normal Scenario	- User selects ‘add new task’ - User writes specific task - Task is added to first ‘To do’ swimlane
Error Scenario	System unable to create new task and does not appear on board

Name	Move task across swimlanes
Participating Actor(s)	User
Entry Condition	User has Kanban board with at least one task in use
Exit Condition	One or more task(s) have moved swimlane and visualisation of Kanban board is updated
Normal Scenario	- User wants to move certain task from one swimlane to another - User selects said task and moves to desired swimlane using drag and drop
Error Scenario	Task does not move or user is unable to drag task

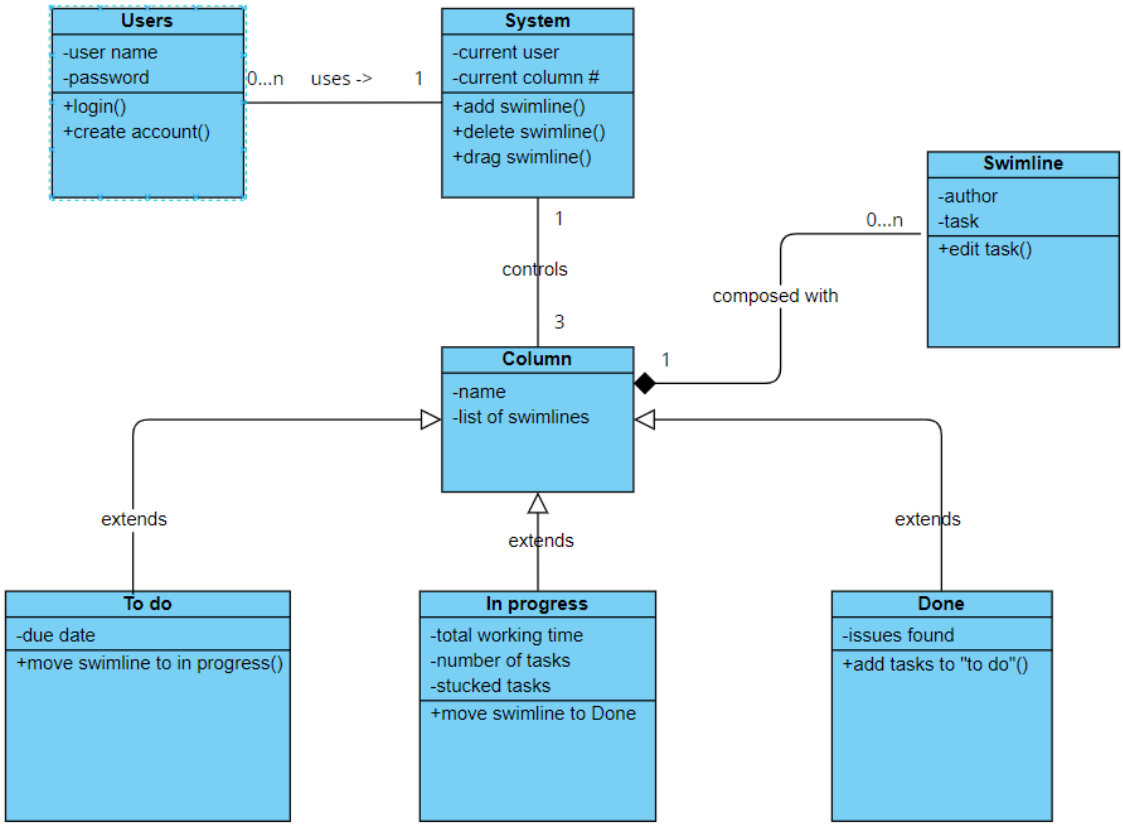
Name	Delete task
Participating Actor(s)	User
Entry Condition	User has Kanban board with at least one task in use
Exit Condition	One or more task(s) have been deleted and visualisation of Kanban board is updated
Normal Scenario	- User decides which task to delete - User selects said task - User selects ‘delete task’ widget

Error Scenario	Visualisation of board is unable to update and therefore task is still visible

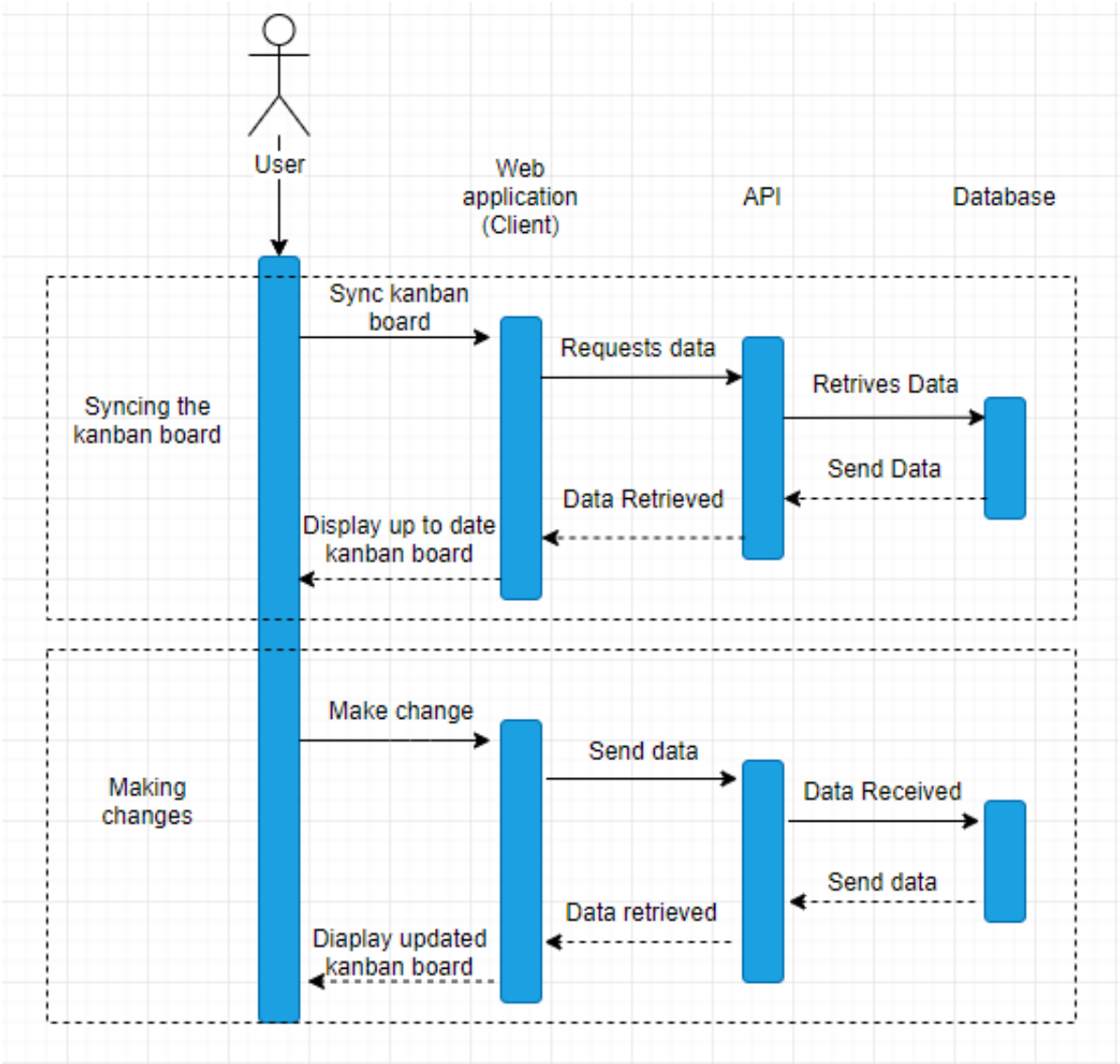
Name	Delete board
Participating Actor(s)	User & System API
Entry Condition	User has Kanban board in use which they wish to delete
Exit Condition	Board has been deleted
Normal Scenario	- User has board in use - User decides to delete entire board - User selects 'delete board' widget - User is presented with pop-ups making sure they want to permanently delete entire board - User selects 'yes' and board is deleted from system
Error Scenario	System unable to delete board

Name	Update visualisation of board
Participating Actor(s)	System API
Entry Condition	User has Kanban board in use and is making changes
Exit Condition	Visualisation of board has been updated
Normal Scenario	- User decides to make change to board, (add task, delete task, move task across swimlane) - User makes said change either using drag and drop or widget - System API updates visuals of board in line with change made by user
Error Scenario	Visualisation of board does not update correctly – task does not get added/deleted/moved

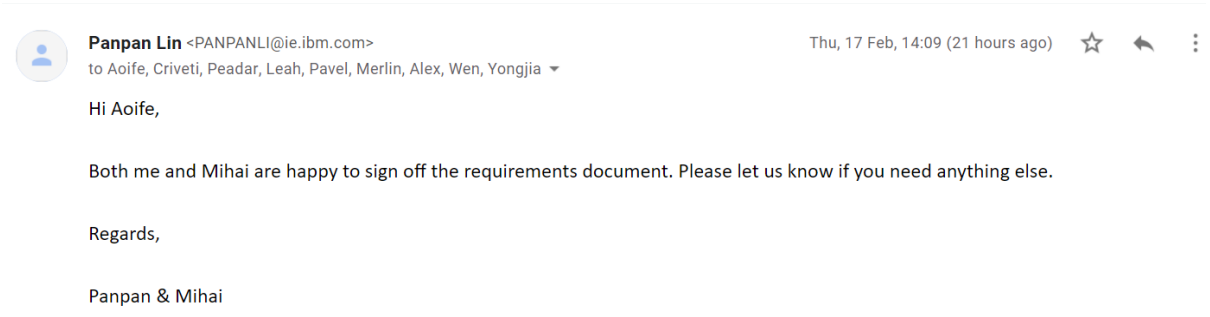
3.4.3. Object Model



3.4.4. Dynamic model



4. Proof of Sign-Off by Client



Panpan Lin <PANPANLI@ie.ibm.com> Thu, 17 Feb, 14:09 (21 hours ago)
to Aoife, Criveti, Peadar, Leah, Pavel, Merlin, Alex, Wen, Yongjia

Hi Aoife,

Both me and Mihai are happy to sign off the requirements document. Please let us know if you need anything else.

Regards,

Panpan & Mihai

Table of contents

1. Introduction

1.1. Overview - Purpose of system

The project our group has been assigned is to develop a kanban board which can support multiple users. A kanban board is an agile project management tool, or essentially a shared space in which teams can visually and asynchronously manage their work.

The design proposed by our team is to create a user-friendly kanban board, designed with a simple swim-lane layout, which would allow users to make changes by moving cards with drag-and-drop functionality. Our client, IBM, has also specified that the system is to be deployed using a modern CI/CD pipeline with container technology based on Red Hat.

While a highly polished end product is always important, our client has put more emphasis on the importance of following the CI/CD development process and gaining development experience, which is what we, as a team, will focus on.

1.2. Scope

The aim of our project is to build a Kanban board that supports multiple users and stores ticket data on a persistent NoSQL database. The project will be built using open source development practices and tools. We will build a CI/CD pipeline with code scanning, unit test coverage and static analysis stages. The application will be deployed on Openshift. The build containers will be based on Red Hat Universal Base Image with Red Hat Quay + Clair as the container registry. The image will be built with podman.

In Scope	**Out of Scope**
- a CI/CD pipeline with code scanning, unit test coverage, and static analysis stages - Create an OpenShift Deployment, Operator or Helm chart with instructions to deploy the application to OpenShift - The Openshift deployment need to be automated and gated by the CI/CD pipeline - The build containers need to be based on the Red Hat Universal Base Image - Use podman to build the image - Use Red Hat Quay+Clair as the container registry - Create a basic Kanban Board app as an example	- Using Ansible for automation of the project.

for CI/CD pipeline	

1.3. Definitions, abbreviations

CI/CD: Continuous integration(CI), is a software development process in which developers all merge code in a central repository and continuous delivery(CD), which adds the practice of automating the entire software release process.

Container: A package of software that contains all of the necessary elements to run in any environment

Container registry: A repository used to store and access container images

Image: An unchangeable, static file that includes executable code so it can run an isolated process on information technology (IT) infrastructure

Red Hat: self-described provider of enterprise open source solutions

1.4. References

- Atlassian. 2022. *What is a Kanban Board? | Atlassian*. [online] Available at: https://www.atlassian.com/agile/kanban/boards [Accessed 7 March 2022].
- Planview. 2022. *Why Use Kanban Boards?*. [online] Available at: https://www.planview.com/resources/guide/introduction-to-kanban/use-kanban-boards [Accessed 7 March 2022].
- Google.2022. *What are containers?* [online] Available at: <https://cloud.google.com/learn/what-are-containers> [Accessed 8 March 2022]
- Red Hat. 2022. *What is a container registry?* [online] Available at: <https://www.redhat.com/en/topics/cloud-native-apps/what-is-a-container-registry> [Accessed 8 March 2022]
- Semaphore. 2022. *CI/CD pipeline* [online] Available at: https://searchitoperations.techtarget.com/definition/container-image [Accessed 11 March 2022]

2. System Design

2.1. Design Overview

Our system will enable users to query data through api. When they make queries, the api will deliver data to the website. Users can checkout three swimlines: to do, doing, and done with such queries. They can also query to add and delete cards in each swimline. This process is done by accessing api and fetching data from the database, and finally flows back to api. The user-application is updated real time by updating data from the api. The api is auto tested and deployed with CI CD pipeline.

2.1.1. High-level overview of how the system is implemented, what tools, frameworks and languages are used

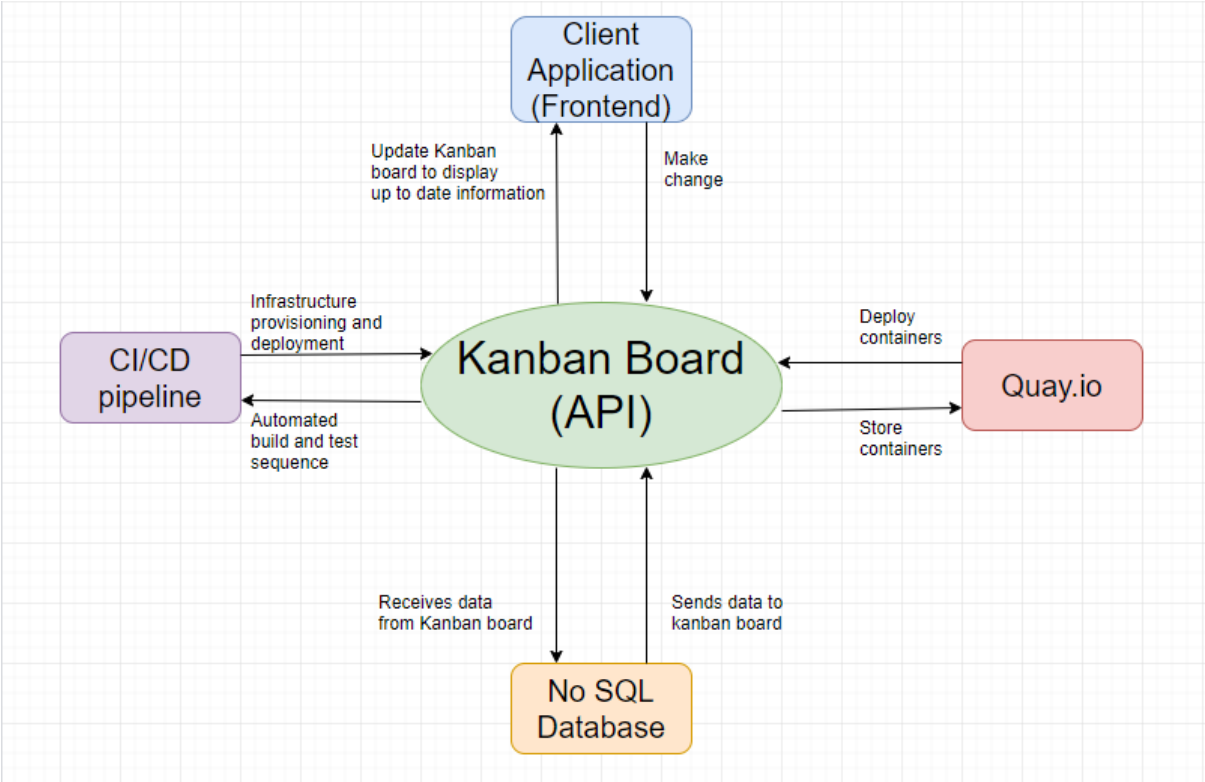
Our system consists of 3 main components - frontend using React, REST API based on Express and a persistent MongoDB database.

The tools and languages used are as follows:

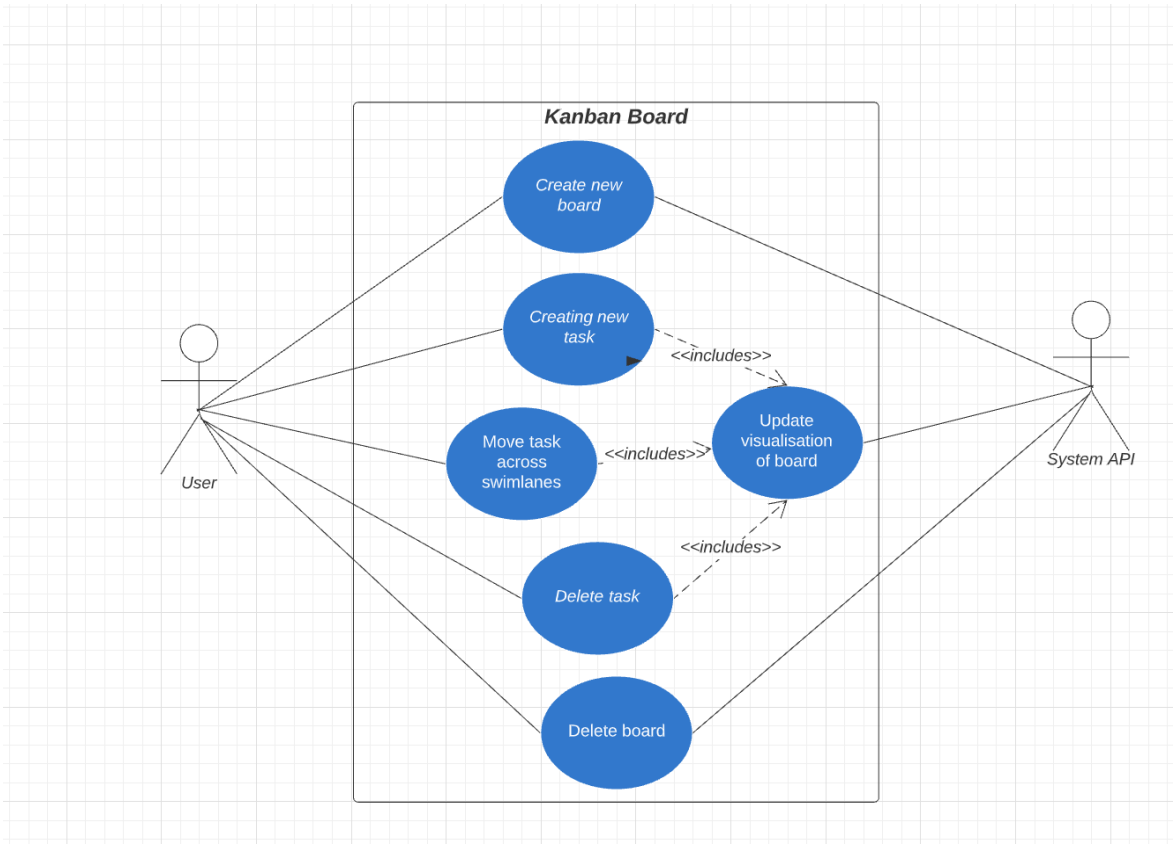
- Runtime environment Node js + Express
- React.js
- JavaScript
- MongoDB
- Docker
- Red Hat Quay
- Jenkins + OpenShift
- Visual Studio Code
- Github + Github Actions
- IBM Cloud
- Postman
- Jest

2.2. System Design Models

2.2.1. System Context



2.2.2. Use cases



Name	Create new board
Participating Actor(s)	User & System API
Entry Condition	Application is accessible & option to create new board exists
Exit Condition	New blank Kanban board is created
Normal Scenario	- User selects ‘create new board’ - User gives new board a name - New blank Kanban board template created
Error Scenario	System unable to create new board

Name	Creating new task
Participating Actor(s)	User
Entry Condition	User has Kanban board in use
Exit Condition	Kanban board has new additional task
Normal Scenario	- User selects ‘add new task’ - User writes specific task - Task is added to first ‘To do’ swimlane
Error Scenario	System unable to create new task and does not appear on board

Name	Move task across swimlanes
Participating Actor(s)	User
Entry Condition	User has Kanban board with at least one task in use
Exit Condition	One or more task(s) have moved swimlane and visualisation of Kanban board is updated
Normal Scenario	- User wants to move certain task from one swimlane to another - User selects said task and moves to desired swimlane using drag and drop
Error Scenario	Task does not move or user is unable to drag task

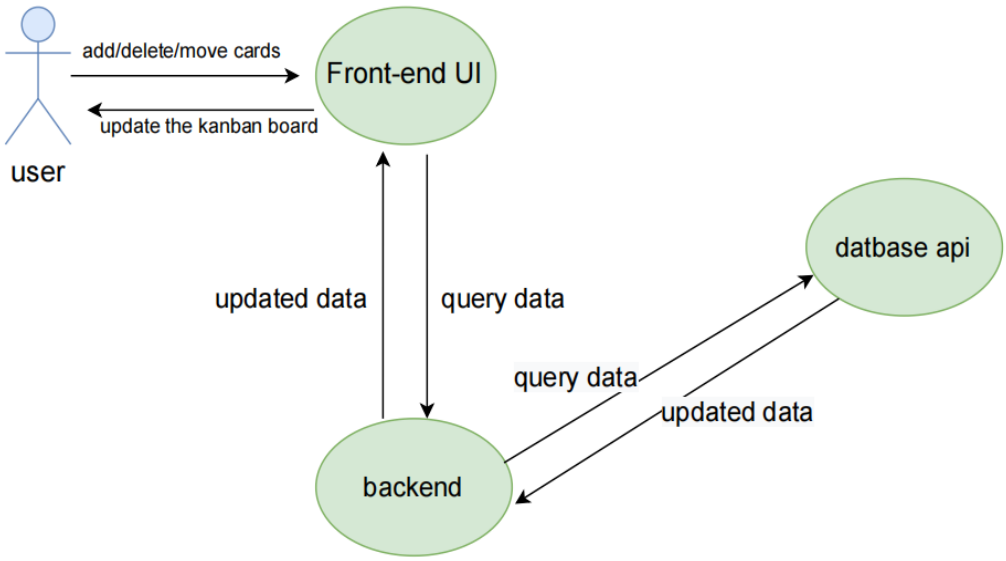
Name	Delete task
Participating Actor(s)	User
Entry Condition	User has Kanban board with at least one task in use
Exit Condition	One or more task(s) have been deleted and visualisation of Kanban board is updated
Normal Scenario	- User decides which task to delete - User selects said task - User selects ‘delete task’ widget

Error Scenario	Visualisation of board is unable to update and therefore task is still visible

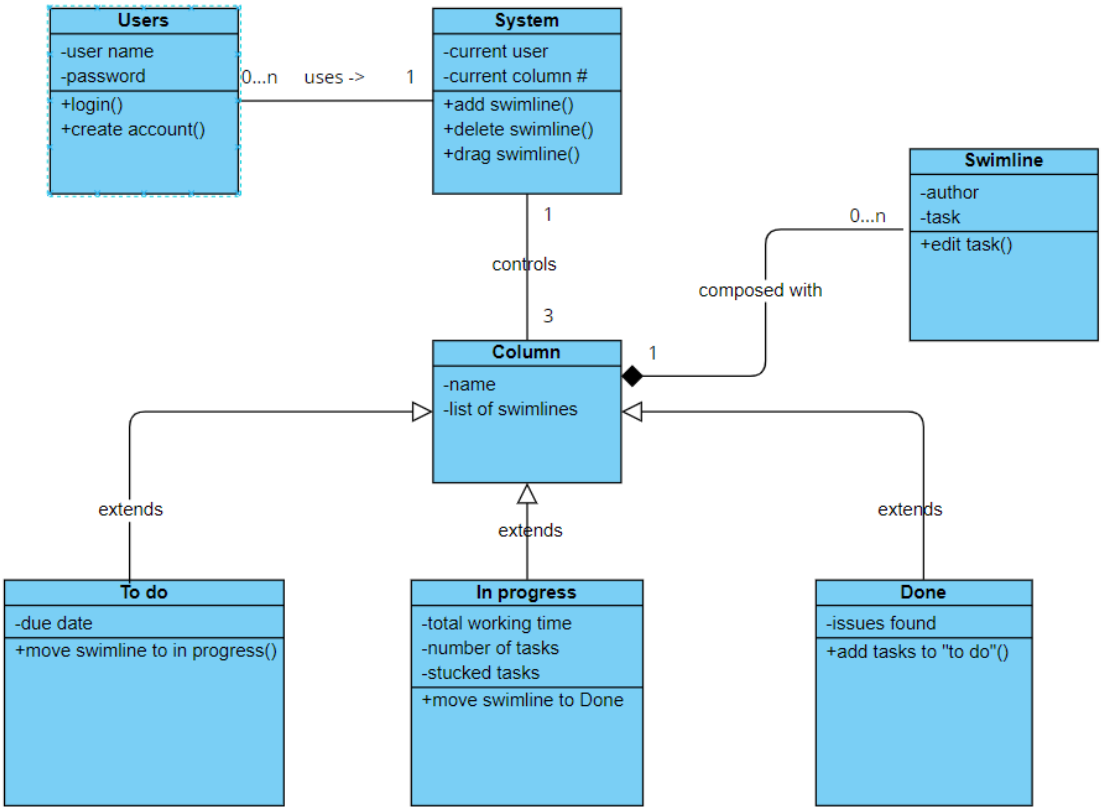
Name	Delete board
Participating Actor(s)	User & System API
Entry Condition	User has Kanban board in use which they wish to delete
Exit Condition	Board has been deleted
Normal Scenario	- User has board in use - User decides to delete entire board - User selects 'delete board' widget - User is presented with pop-ups making sure they want to permanently delete entire board - User selects 'yes' and board is deleted from system
Error Scenario	System unable to delete board

Name	Update visualisation of board
Participating Actor(s)	System API
Entry Condition	User has Kanban board in use and is making changes
Exit Condition	Visualisation of board has been updated
Normal Scenario	- User decides to make change to board, (add task, delete task, move task across swimlane) - User makes said change either using drag and drop or widget - System API updates visuals of board in line with change made by user
Error Scenario	Visualisation of board does not update correctly – task does not get added/deleted/moved

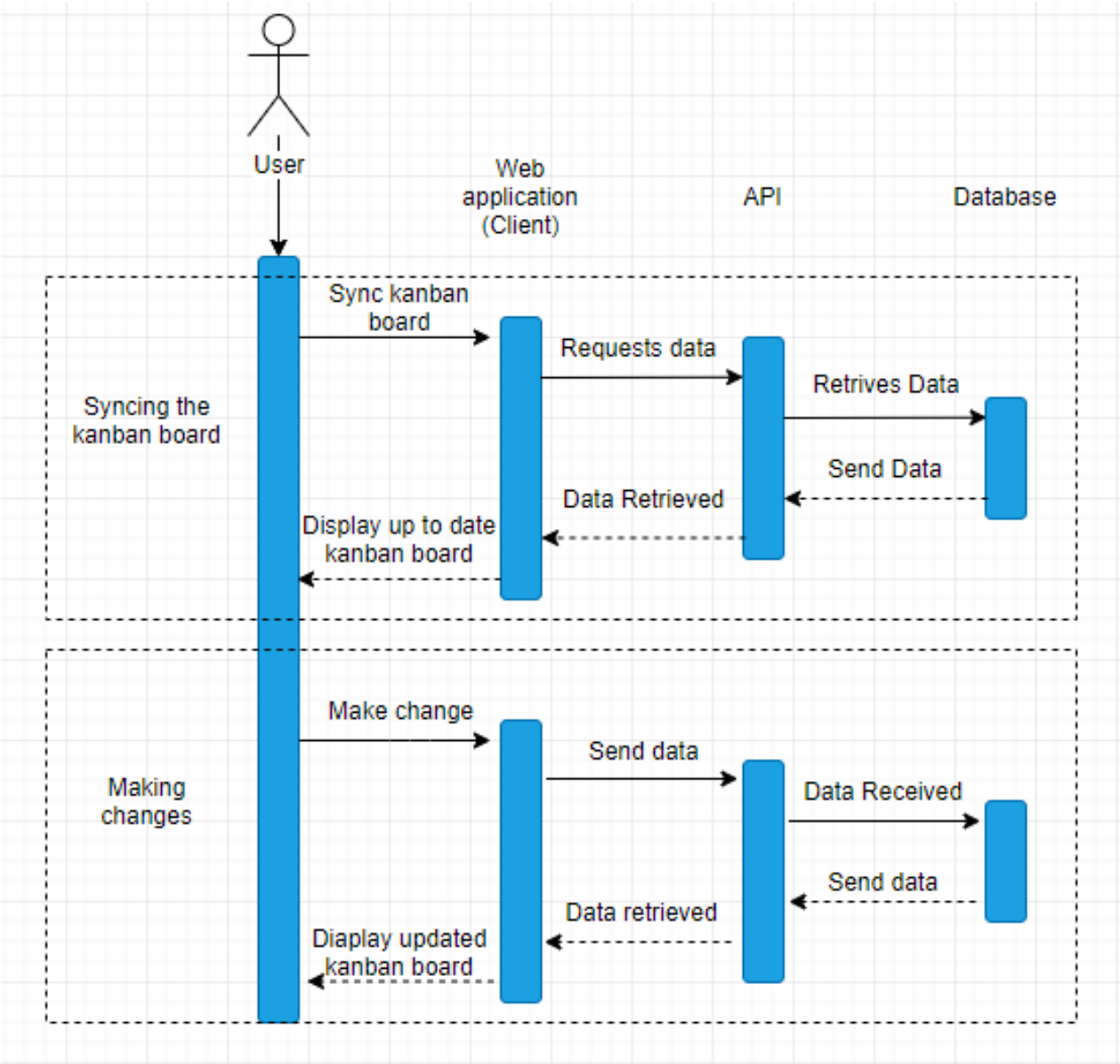
2.2.3. System Architecture



2.2.4. Class Diagrams



2.2.5. Sequence Diagrams



2.2.6. State Diagrams