Sami Haapanen

Developing a project management system that suits the development team

Metropolia University of Applied Sciences

Bachelor of Engineering

Information technology

Bachelor's Thesis

19 April 2021

Abstract

Author:	Sami Haapanen
Title:	Developing a project management system that suits the development team
Number of Pages:	27 pages
Date:	19 April 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and communication technology
Professional Major:	Software Engineering
Instructors:	Jari Majaniemi, Chief Technology Officer Janne Salonen, Principal Lecturer

The purpose of this study was to create an option for a development team's possible future project management tool. The solution was created for Mousewell Oy's development team, that has not been fully happy with their current project management system.

The development team's work was analysed to find any possible software improvements. There were also some needs from the business department and those were taken into account during the project. The goal was to create a project management system that could solve some of the development team's problems and have beneficial features over their currently used software.

In this study a prototype version of a project management system was created. The developed prototype is a light frontend-only solution to demonstrate a higher number of features in a short time scope. The prototype can be further developed by the development team. The result proves that it is possible to develop a project management system that suits the development team's needs in a short time frame.

Keywords: project management, development team, software

Tiivistelmä

Tekijä:	Sami Haapanen
Otsikko:	Kehitystiimin tarpeisiin soveltuvan projektinhallintajärjestelmän kehittäminen
Sivumäärä:	27 sivua
Aika:	19.4.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistokehitys
Ohjaajat:	CTO Jari Majaniemi Tutkintovastaava Janne Salonen

Opinnäytetyön tarkoituksena oli luoda ohjelmistokehitystiimille mahdollinen tuleva projektinhallintatyökalu. Projekti tehtiin Mousewell Oy:n kehitystiimille, joka ei ollut täysin tyytyväinen heidän tämänhetkiseen projektinhallintajärjestelmäänsä.

Kehitystiimin työtä analysoitiin mahdollisten ohjelmistoparannusten löytämiseksi. Myös liiketoimintaosastolla oli tarpeita, jotka otettiin huomioon projektin aikana. Tavoitteena oli luoda projektinhallintajärjestelmä, joka voisi ratkaista tiimin ongelmia ja jossa olisi hyödyllisiä ominaisuuksia nykyiseen järjestelmään verrattuna.

Opinnäytetyössä kehitettiin prototyyppi projektinhallintajärjestelmästä. Kehittettyyn prototyyppiin tehtiin ainoastaan käyttöliittymäpuoli, jotta voitaisiin demonstroida manhdollisimman paljon ominaisuuksia lyhyessä ajassa. Prototyypin jatkokehitys on kehitystiimin vastuulla. Lopputulos todistaa, että lyhyessä ajassa on mahdollista kehtittää ohjelmistokehitystiimin tarpeisiin sopiva projektinhallintajärjestelmä.

Avainsanat: projektinhallinta, kehitystiimi, ohjelmisto

Contents

List of Abbreviations

REST:	Representational state transfer
API:	Application programming interface
POC:	Proof of concept

1 Introduction

Modern software development teams usually use some kind of project management system to plan and keep track of their work. There are many to choose from and teams try to pick the one that best suits their needs. They vary from simple Kanban boards to more versatile software with a greater range of project management tools.

Choosing the correct project management system is not always easy as the number of options can be overwhelming and getting familiar with each software takes time. Sometimes development teams might notice a short time after use that their project management system does not satisfy or perfectly suit their needs.

Mousewell Oy is a software company that develops SLP Group Oy's services, one of which being an invoicing service for light entrepreneurs, Ukko.fi [1]. Mousewell's development team uses Atlassian's Jira for project management and the team has not been always happy with the project management system.

Switching to another system would be time-consuming and there is no assurance that the new one would end up being any better. One solution could be making an own project management system from scratch. That way the team could decide themselves about all the features and the functionalities.

The goal of this study is to create a demonstratable prototype version of a project management system for Mousewell's development team. The project management system should have some improved features over Jira.

2 Project initialization

2.1 Background

Before starting to plan a new project management system, the development team's current working methods had to be taken under consideration. Mousewell is a software development company that specializes in agile development [1]. As one of their working methods Mousewell's development team uses scrum.

2.1.1 Scrum

Scrum is a popular agile working framework used for iterative development. Scrum is usually used by software development teams, but its ideas can be applied to any kind of teamwork. [2.] Scrum is not a strict rulebook that should be followed word for word. It can and should be tailored to fit the organization's needs.

The work in scrum is split into short time periods called sprints. In Mousewell's case, sprint is a bi-weekly cycle. Each sprint begins with a planning session where the team takes a set amount of work from the product backlog to the sprint backlog. Product backlog is essentially the team's to-do list for the whole project and sprint backlog a list of tasks that the team will try to get done during a sprint. [2.]

At the end of each sprint is a review session where the team analyses the past sprint. During the review session, the sprints goals are reviewed and any demonstratable new features that got done during the sprint can be presented. After the review session is the retrospective meeting where the team documents and discusses all things worked and what did not work during the sprint. It is a place where the team can focus on all the positive things and try to solve any problems they have in their daily work. [2.]

2.1.2 Jira

Mousewell’s development team’s current project management system Jira has been listed a few times in the retrospective meetings and usually in the negatives. The usual complaints were about Jira’s long loading times cutting the workflow and some simple core features being frustratingly complicated to use.

The ways the team uses Jira for project management had to be analysed. As most project management systems, Jira has a backlog of upcoming features of the project. Features are separated into doable tasks and during a sprint, tasks are placed on a board that shows their progress. As in most cases the board view is the main feature developers use.

The team also uses Jira for sprint planning. One thing noticed in the team’s planning sessions is that the team very often commits to more work on the than they can get done in one sprint. By looking at the past sprints the team can estimate how much work they usually get done. Although Jira saves statistic of the past sprints, the information is not easily available where the planning is done and looking at history statistics in every planning session takes time.

Getting familiar with other available project management systems did not really help, as the plan was to make improvements that suit the team’s working methods, so most of the ideas came directly from experience. The idea behind analysing the ways the team uses Jira was to take the things that work and improve the ones that could be done better.

2.2 Initial Plan

The initial plan was to create a POC (proof-of-concept) version of a project management system for Mousewell’s development team. The prototype could later be used as a base a software that replaces Jira as the development team’s project management system.

One of the initial ideas was to use Jira’s API to provide all the data that has been gathered over a long period of time. That would solve many problems concerning the data transfer while switching to a new software and also remove the need of a database and backend. The project was developed keeping the utilization of Jira’s API in mind.

The project’s goal was restricted into a frontend-only version to get the maximum amount of beneficial features done in the limited time scope. The target was to create a software that proves its worth with new useful features that Jira does not offer.

3 Development Process

3.1 Planning

The development process started by making rough sketches of all the ideas of features that came to mind. These sketches do not need to be perfect representations of the final product, just good enough to make the ideas behind them clear. Figure 1 shows an example of a rough sketch that helps visualize the idea.

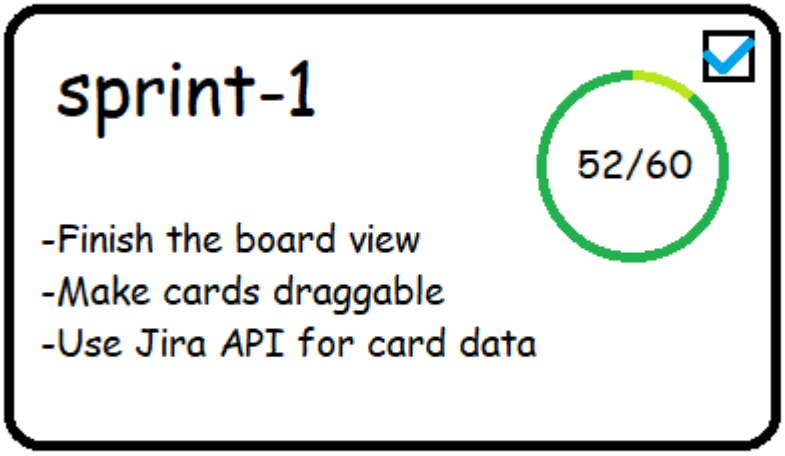

Figure 1. A sketch of the sprint card UI component.

After sketching, the ideas were presented to the development team which then gave feedback. The team liked some of the ideas, but some were dumped at the very beginning. This was a very important step as it reduced redundant work. If some of these features were developed further and then noticed to be unnecessary or useless, a lot of time would have been wasted.

SLP Group's business department were also asked if they would have any needs regarding development team's project management system. Only one

thing came up. It was to find an improved way of monitoring the development team’s progress.

3.2 Implementation

When there was a good amount of planned features, the implementation started. In the beginning the feature ideas were put into a priority order on a backlog. Also new ones were added to the list during the actual development as ideas came to mind. The final versions of the features had sometimes changed from the original sketches as the ideas clarified during the implementation.

The features were coded one at a time and when finished added to the project. Before adding to the project, the feature’s code was always reviewed by at least one frontend developer of Mousewell’s development team. Frontend developers read the code through and commented if they found any mistakes or anything that could be improved. This minimized the amount of accidental errors and improved the overall quality of the code.

4 Features

The new features that were added or the ones that were reimplemented from Jira had to fit scrum and the team’s other working methods. The new features should show improvements over Jira. All needs for the development team and business were attempted to be fulfilled.

4.1 Reimplementation of the Core

The goal was not to copy the old project management system before adding some extra features to it but to rather reimplement only the most important parts. In this project’s case, the point of a POC version was to prove that the new features would work and could be easily implemented.

The main functionality that was reimplemented was the board view. The board view is the core of most project management systems, and it is the one developers use the most. The board is a project management tool that helps to visualize the work to maximize the development team's efficiency [3]. Figure 2 shows the project management system’s board view. The board view splits the team's work into columns that represent different phases of work. The phases could for example be:

- to-do
- planning
- implementation
- in code review
- done.

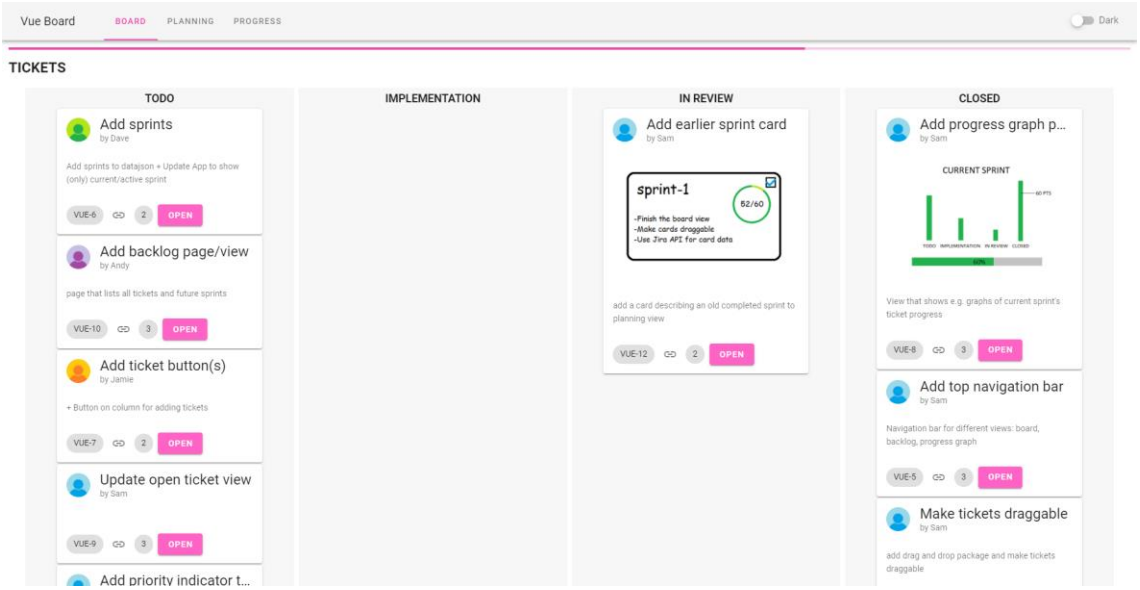

Figure 2. The board view showing all tasks in columns that represent the work phases.

As in most agile boards the tasks have a drag and drop functionality, meaning that the tasks can be moved around the board by dragging them with the mouse. Drag and drop is not a unique functionality to just agile boards and it is not the simplest feature to code out of scratch either. To implement the drag and drop functionality, a package called vuedraggable, created by David Desmaisons, was added to the project [4]. Fortunately, in software development there is usually no need to reinvent the wheel. Especially, when it comes to these kinds of features, someone has probably already done it.

Before adding features for sprint planning and monitoring development team's progress, it was important to note and reimplement how work was measured. For measuring work the team uses story points that describe a task's complexity.

This has noticed to be a lot better practice than using time as measurement. It is impossible to estimate accurately enough how long a task takes and even if it were possible, the time would differ depending on the assignee of the task. For

example, developers with more working experience or knowledge of the codebase would need less time than others. Putting a time limit on a task would also give developers unneeded stress and even though the time limits would be estimates, someone could easily misunderstand them as deadlines.

As the team has noticed story points to work well for them, there was no reason to try improved it. Story points were reimplemented as work's measurement and as a pre-requirement for adding the sprint planning and monitoring features.

4.2 Added Features

4.2.1 Sprint Planning

One of the main features that the development team would benefit from was the added sprint planning view (Figure 3). It helps the team plan their next sprint by calculating a suggested amount of points from history data.

The view uses sprint card components that represent the past sprints. The sprint card shows how many points the team got done out of the initially committed amount. It also lists the sprint's goals to make all sprints more easily recognizable.

The sprint planning component takes all selected sprint cards and for each sprint calculates the average points for one member. This is important as sprints may have different amount of members. After that, it calculates the average of all sprints' one member averages. User then selects the members of the upcoming sprint and the suggested point amount is calculated by multiplying the average of one member of all sprints by the amount of members.

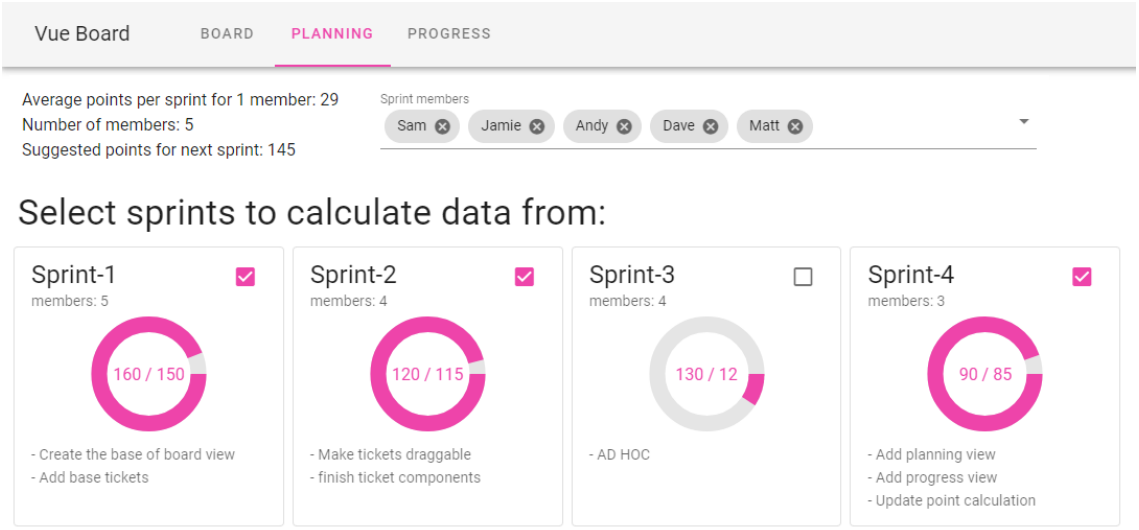

Figure 3. Sprint planning view calculating suggested points for the chosen members from selected history sprints.

Sometimes sprints can fail due to external factors such as serious bugs that need quick attention, or other unexpected things that have higher priority than the sprint goals. These failed sprints create anomalies to the statistics and disrupt the average calculation, therefor the ability to leave some sprints out of the calculation was added. The team may also want to leave old history sprints out of calculation as those might not accurately resemble the current point amounts.

4.2.2 Monitoring Progress

SLP Group's business department had some wishes for features that would help the development team's progress. Although the board view shows exactly the progress of each task on the sprint, it is made for developers and can have an overwhelming amount of information for someone who does not use it in everyday work.

To monitor the ongoing sprint’s status, a progress bar was added to the top of the board view as seen in Figure 2. Progress bar shows the current situation of

the board's tickets based on each ticket's column and point amount. The progress bar fills up as tickets move from left to right and is full when all tickets are in the last column. It is not only a way to monitor the progress sprint but also a good way to give developers the feeling of accomplishment without giving them too much stress. In the best case it might even help push the team to try and reach the sprints goal.

Although the board view's progress bar helps to visualise the status of the current sprint, it does not give out that much useful information. In order to give more in-depth information of the current sprint's status, the progress view was added. The progress view has a graph that displays the combined sum of points of tickets on each column (Figure 4). It also has the same progress bar component, that additionally to the one on board view, shows the percentage value.

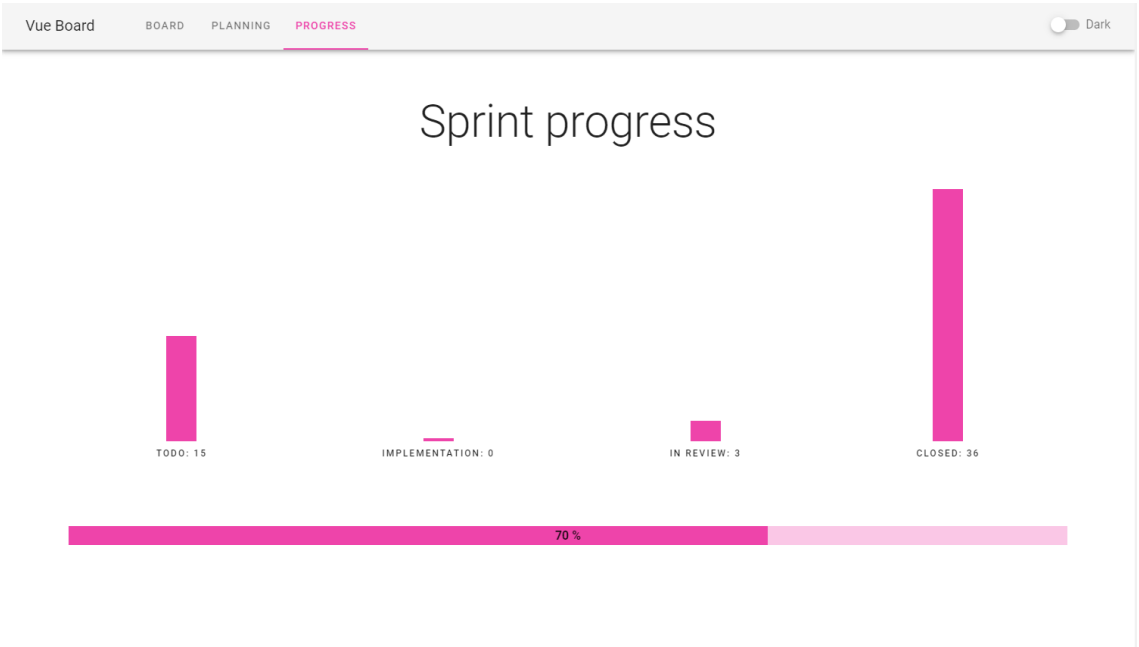

Figure 4. Progress view showing a graph and a progress bar that describe the sprint's status.

With a quick look of progress view anyone can get a clear sense of the current sprint's status. If needed other graphs and more detailed information can be easily added to the progress view.

4.2.3 Quality of Life Features

There are also quality of life or nice to have features that are not necessarily crucial but can still improve the user experience of a software a lot. These features are sometimes considered to be premium and can help the software to stand out from the competition.

One quality of life feature that has become very popular in modern applications is having the option to choose light between light and dark theme. A dark mode switch was added to the project management system using a Vuetify's ready-made component. The implementation took just minutes while having most visual components made with Vuetify. Figure 5 shows how the project management system looks with dark mode enabled.

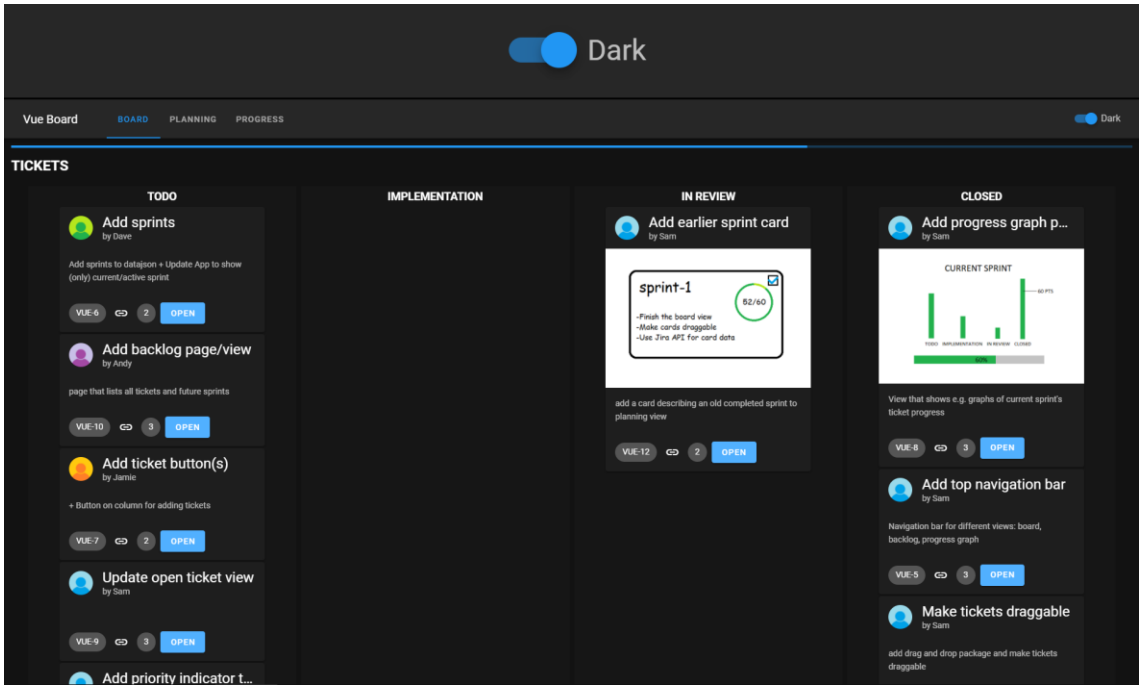

Figure 5. Dark mode switch and the board view in dark mode.

People usually have preferences over these kinds of things, so it is good to give them the option. But it is not always about preference. The user might choose their theme due to working environment factors. For example, working in a low light environment dark mode can be less likely to cause eye soreness but on a

bright day it can cause annoying screen reflections and become less usable. This feature is not crucial part of the software's core functionality, but it shows that adding features that improve the user experience, can be fast and easy.

4.3 Future Features

4.3.1 Jira Integration

Jira offers a REST API (Representational State Transfer, Application programming interface) for its users to interact with the application remotely. The REST API is created to enable development of any kind of integration for Jira. [5.]

One of the initial ideas was to utilize Jira's API in the project. With the API Jira could be used as the data provider of the new project management system and neither a separate backend nor database would be needed. It would also enable simultaneous use of Jira and the new project management system before all features are complete.

The API can be used for fetching, creating, updating, or deleting any offered data from the Jira project. For example, all tasks of the project could be fetched from the API and placed on the board of the new project management system or a new task's data could be inputted in the application and sent to Jira's API to create a new task. Listing 1 shows an example of a possible JSON response of calling the API's get issue endpoint.

```
{
  "id": "5",
  "self": "https://your-domain.atlassian.net/rest/api/3/issue/5",
  "key": "VUE-5",
  "fields": {
    "watcher": {
      "self": "https://your-domain.atlassian.net/rest/api/3/issue/EX-
1/watchers",
      "isWatching": false,
      "watchCount": 1,
      "watchers": [
        {
          "self": "https://your-
domain.atlassian.net/rest/api/3/user?accountId=5b10a2844c20165700ede21g",
          "accountId": "5b10a2844c20165700ede21g",
          "displayName": "John Doe",
          "active": false
        }
      ]
    },
    "attachment": [
      ...
    ],
    "sub-tasks": [
      ...
    ],
    "description": {
      "type": "doc",
      "version": 1,
      "content": [
        {
          "type": "paragraph",
          "content": [
            {
              "type": "text",
              "text": "Add top navigation bar"
            }
          ]
        }
      ]
    },
    "project": {
      ...
    },
    "comment": [
      ...
    ],
    "issuelinks": [
      ...
    ],
    "worklog": [
      ...
    ],
    "updated": 1,
    "timetracking": {
      ...
    }
  }
}
```


Listing 1. A possible JSON response from Jira's API's Get issue endpoint showing the data of 1 issue. Tasks are called issues in Jira. The response has been shortened to show only the most relevant data for the project management system's current state.

Unfortunately, the utilization of Jira's API did not fit in the time scope of the thesis project, but it was always kept in mind while developing the project's other features. The Jira integration will definitely be one of the next steps of the application's further development.

Another option is to create an own backend and database for the software. As the team probably does not want to lose project data that is currently in Jira, this would require a large data transfer. The data transfer could possibly be done by utilizing Jira's API, but it would require creating some sort of data transfer tool that would mangle the data into usable form for the database. This would take a lot more development time, not only for the data transfer tool but the team would also have to develop and maintain the backend project. In the other hand it would give more freedom for future development as all the features would not be dependent on Jira's data models. In the end the team will decide together, in which direction they want to take the project.

4.3.2 Sprint Retros

As of now there is no logical place for retrospective meeting notes in Jira and the team uses other tools for that. This means that all the sprint data is not stored in one place. Having to search for illogically located data wastes time and can cut the developers' workflow.

As a possible future feature there could be a place in the system to create retrospective meetings' notes. The notes could be added to sprint history data when completing a sprint. This way the data would be more easily accessible as it is stored in the same place with everything it relates to, making the project management system more suitable for the team's working methods.

4.3.3 Other Future Development

There are still many ideas on the project's backlog for future features and also a lot of features that could be reimplemented from Jira. Everything could not fit in

the time scope, so the features had to be prioritized and the ones that were considered to be most beneficial for their development time were implemented to the POC prototype.

After the completion of the prototype, the further development will be in the team’s hands. There are still many planned ideas and unimplemented functionalities that are necessary to make the project management system usable. The team will also most likely get new ideas for features and improvements in the future, so they will have to decide which ones have the highest priority. The project management system will probably not get released fully at once. More likely single features are going to get finished and added to the team’s working methods one at a time.

5 Technologies

5.1 Vue.js

The thesis project was created using a popular JavaScript framework Vue.js. Using Vue for the project was an easy choice as it is used as Mousewell's main frontend framework for most of their software projects.

Vue.js is a progressive JavaScript framework that focuses mainly on the view layer and built is to be easily adaptable. Vue.js is an open-source project meaning anyone in the community can contribute to its codebase. [6.] Vue is used for creating web interfaces and single page applications but can also be used for developing mobile and desktop apps [7]. Vue.js was created by Evan You and was initially released in February 2014. Vue is still a fairly new framework and has gained a lot of growth in popularity after its release. According to StackOverflow's survey made in 2020 Vue.js is the 7th most popular of all web frameworks (Figure 6) [8].

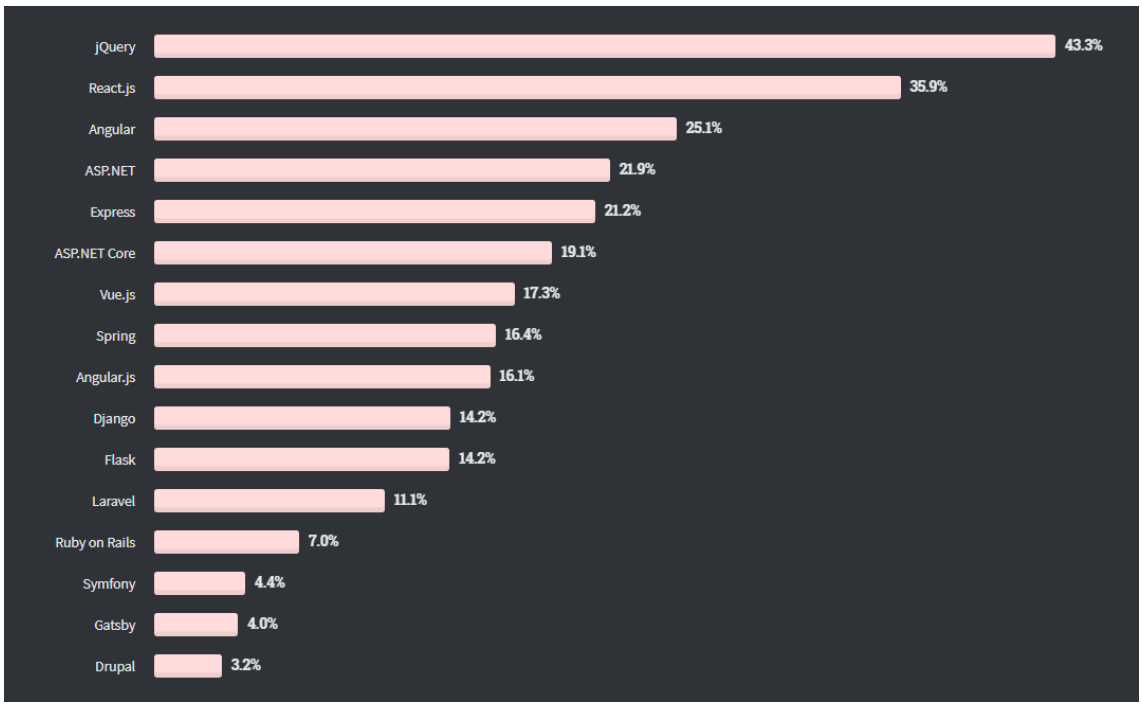

Figure 6. StackOverflow's survey’s results listing the most popular web frameworks of 2020 with 42 279 responses.

The survey also has also listed the most loved and most wanted frameworks. The loved list shows the percentage developers who are using the technology and want to keep doing so and wanted developers shows the amount of developers who are not using but are interested in the framework. Vue is placed third in the most loved list and second on the most wanted. [8.] Vue's growing popularity and the community's interest towards it make Vue a safe choice of frontend framework for the future too.

Vue has many great features that make the development process more enjoyable. The quick installation process and comprehensive documentation make Vue easy to get started with. The code in Vue is split into single-file components that resemble pieces of interface. Single-file components are pieces of code that can be easily reused wherever needed. Splitting code into smaller pieces improves the code's readability and maintainability. [7.] Figure 7 shows a component being used in 2 different views.

```
<template>
  <div class="pa-3">

    <progressBar :epics="dataJson.epics"></progressBar>

    <epic v-for="epic in dataJson.epics" :key="epic.name" :heading="epic.name">···
    </epic>

  </div>
</template>
```

```
<template>
  <div style="max-width:1400px" class="mx-auto mt-16">

    <div class="text-h2 text-center">Sprint progress</div>

    <progressGraph :epics="dataJson.epics"></progressGraph>

    <progressBar class="ma-10" :showPercentage="true" :epics="dataJson.epics"></progressBar>

  </div>
</template>
```

Figure 7. The progress bar component being used in the Board and Progress views. The optional property showPercentage defines how the progress bar looks like in the UI.

Vue offers a developer tools browser extension that makes debugging Vue applications easier. The extension offers a set of utility tools that help with developing Vue applications. [9.] The components tab shows an element tree of the page's Vue components in the same hierarchy as in the code. By selecting a component, all of its data can be viewed as it is in the component's current state. The events tab lists all events caused by actions made in the UI and shows all changes made to components' data by the events (Figure 8).

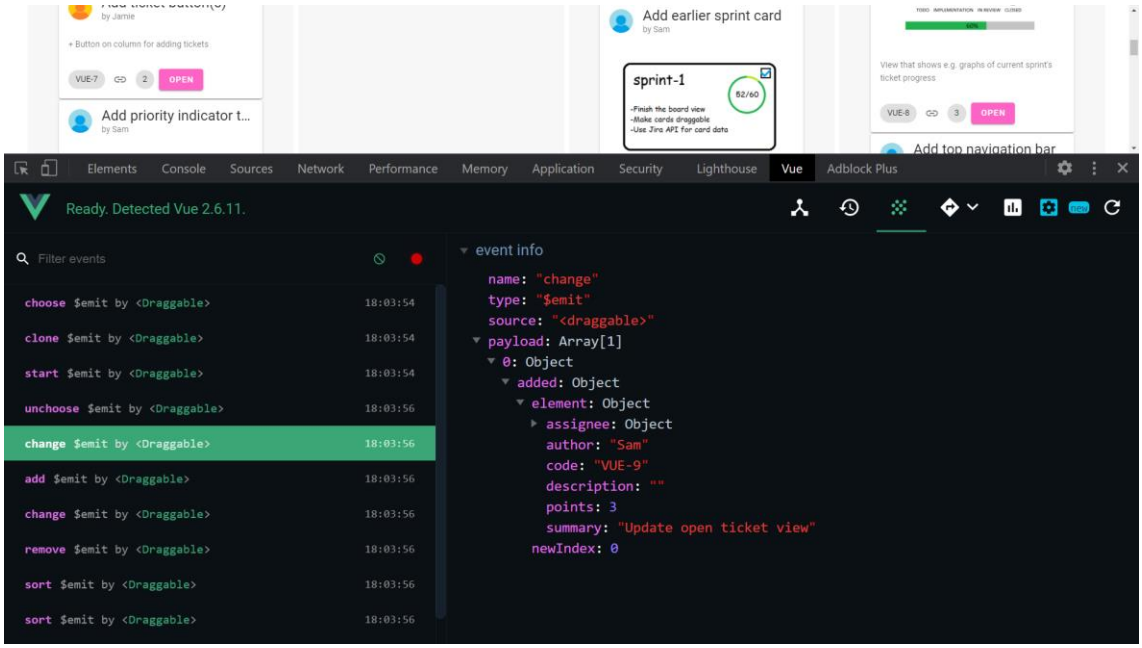

Figure 8. The Vue developer tool events tab showing all events caused by dragging a task to another column. The selected change event shows the task added to the new column with all its data.

Both the components and events tabs were utilized while developing the thesis the project. But that is not all. The developer tools extension offers also a lot of other useful features and mastering them will take Vue development to the next level.

5.2 Vuetify

Along with Vue.js, Vuetify was used in the thesis project. Vuetify is a UI framework with hundreds of ready-made Vue components that are crafted using

Material Design specification. Vuetify is built to be easy to learn and it aims to give developers all the tools they need to create great user experiences. [10.]

Vuetify's versatile UI component library has a lot of different components for different uses. The components are presented in an accessible way by showing how they react to user actions in the UI and how the components look like with different parameter values (Figure 9). Code for template and script is available for all components in the library.

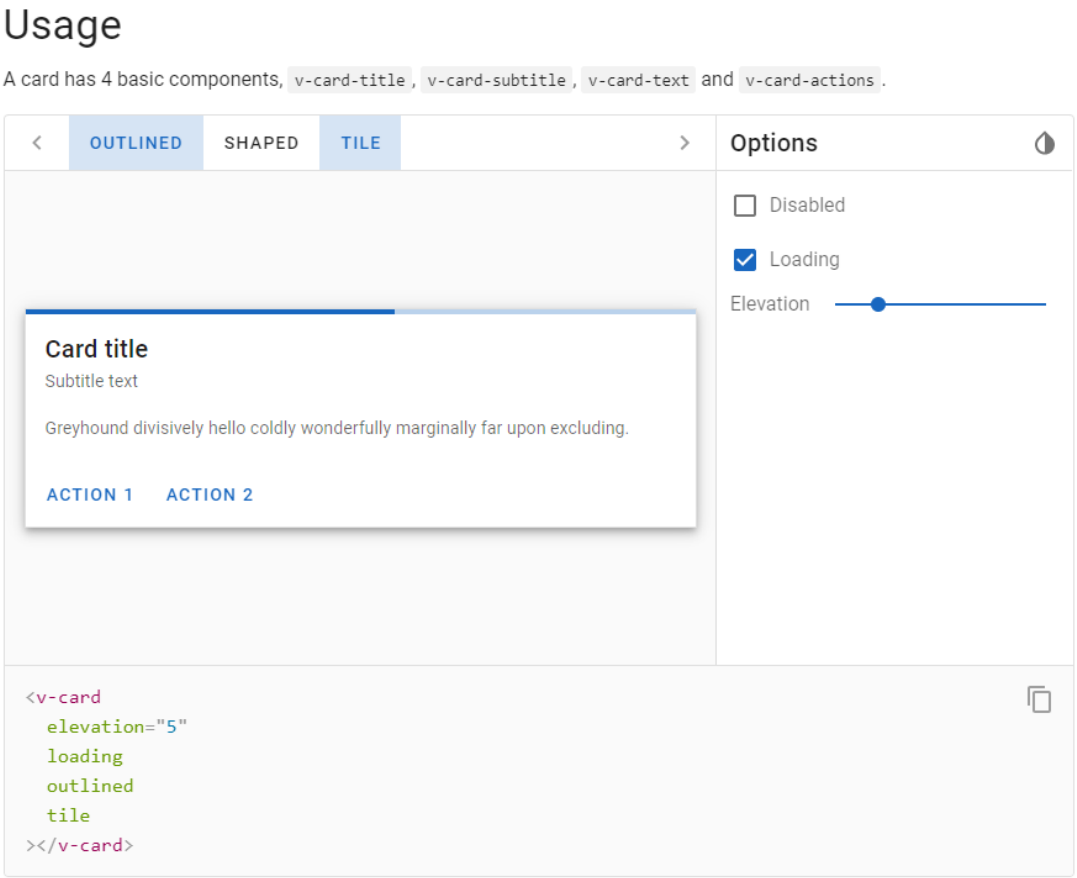

Figure 9. Card component from Vuetify's UI component library.

Vuetify also offers ready-made customizable CSS utility classes (Figure 10) that work similarly to a popular CSS framework TailwindCSS. These classes offer for example spacing, colors, flexbox settings and display breakpoint helpers. The utility classes help with creating the UI and finetuning layouts.

```
<div class="d-flex justify-space-around pa-10 mt-5">
```

Figure 10. Vuetify’s utility classes defining the spacing and flex settings of an element.

With Vuetify's ready-made components and CSS classes, developers can easily create beautiful user interfaces without the need for a UI designer. Creating single-file components using Vue and Vuetify together made the development process of the project a lot faster.

5.3 Data

One idea was to use a JavaScript library called Mirage to provide the data by mocking Jira’s API. Mirage is a tool that enables testing the application just as if it were communicating with a real server. Even though using Mirage would have been handy, it was noticed to be unnecessary in the POC version as Jira’s API did not fit the scope either. Instead, the data was just simply place into JSON files in the same format the application uses it (Listing 2).

```
{
    "epics": [
        {
            "name": "tickets",
            "columns": [
                {
                    "name": "todo",
                    "tickets": [
                        {
                            "code": "VUE-9",
                            "summary": "Update open ticket view",
                            "description": "",
                            "points": 3,
                            "author": "Sam",
                            "assignee": {
                                "img": "user-1.png",
                                "name": "user-1"
                            }
                        }
                    ]
                },
                {
                    "name": "implementation",
                    "tickets": [
                    ]
                },
                {
                    "name": "in review",
                    "tickets": [
                    ]
                },
                {
                    "name": "closed",
                    "tickets": [
                        {
                            "code": "VUE-4",
                            "summary": "Make tickets draggable",
                            "description": "add a drag and drop package",
                            "points": 8,
                            "author": "Sam",
                            "assignee": {
                                "img": "user-1.png",
                                "name": "user-1"
                            }
                        }
                    ]
                }
            ]
        }
    ]
}
```


Listing 2. JSON data used in the Board and Progress views. Data has been reduced to show only 2 tasks. As can be seen while compared to Jira's API's JSON response in Listing 1, data of a single task is in much simpler form.

This just sped up the development process as data was already formatted to be easily handled by the application. It was good enough solution for this state of the project. Overengineering should be always avoided and especially when developing a prototype.

5.4 Project Management

The board view that was implemented to the thesis project was actually used itself to keep track of the project's work. That way the features were constantly tested throughout the project and if something were lacking it would have been noticed right away.

Using the project for its own project management was also a really good way to provide realistic data. There was no need use fake data generators or copy dummy text like Lorem Ipsum. This saved some time and also removed the need of an external project management system.

5.5 Version Control

The popular version control system git was used for version control in the thesis project. The project was initialized in Mousewell’s private Github repository like almost all Mousewell’s other projects. This means it is not available publicly.

The coding was done locally in feature branches and when a feature was complete it was pushed to the remote git repository in Github. In Github the code was reviewed by a Mousewell’s frontend developer using pull requests. Pull requests are a great way to show other developers of the team the changes made to the repository and that the feature is ready. In pull requests the code reviewer can approve, comment or request changes to the code. Suggestions done in code reviews are a quick and efficient way to make small fixes to the code (Figure 11).

Figure 11. Suggested change in a code review done for the Progress view feature's pull request.

When the reviewer had accepted the code, the feature branch was merged to master branch meaning the feature then becomes part of the application. Usually in software projects, there are release branches that merge many features to master at once but in small scale projects it is unnecessary.

6 Results

The result was a small POC version of a project management system that proves developing a software with beneficial features good-looking UI can be fast and worth the time. Although all the initial ideas were not implemented, the application has new useful features and the project met most of its goals.

The software has 3 views 1 of which was reimplemented and 2 that were completely new features over the old project management system. The planning view was created to help the development team with sprint planning and the progress view for business department to help keep track with the team’s progress.

There is still a lot of features on the project’s backlog ready for the team to develop in the future. The POC project could further be developed to work simultaneously with Jira or even into its own complete project management system. Ultimately, it’s up to the team to choose how they want to advance with the project.

7 Conclusion

If a development team is not happy with their project management system and feel like switching to another one will not solve the problem, developing their own project management system can sometimes be the best option. That way the team can try to solve any issues they currently have and build a project management system that perfectly to suit their needs.

In Mousewell's case, the development team was not fully satisfied with their current project management system. It was also noticed that there could be software solutions that would improve their working methods with agile development too. And not only the development team, but the business department too could benefit from the creation of a new improved project management system.

In this thesis project a POC prototype version of a new project management system was developed. The prototype offers features that could potentially solve the development teams issues with agile development. It also introduces a beneficial feature for the business department.

One of the initial plans was to used Jira's API as the data provider. It couldn't unfortunately fit the time scope of the thesis project, but its implementation is just a question of time as the future of the new project management system is in the hands of all the team's developers.

The goal was to create a prototype of a new project management system that shows improvements over the current software and that goal was definitely met. The prototype proves that by creating a project management system out of scratch the team can have a software that perfectly fits the way they work.

References

1 Mousewell. Online source. <https://www.mousewell.fi/>. Read 4 April 2021.

2 What is Scrum? Online source. Atlassian. <https://www.atlassian.com/agile/scrum>. Read 2 April 2021.

3 What is a Kanban board? Online source. Atlassian. <https://www.atlassian.com/agile/kanban/boards>. Read 2 April 2021.

4 David Desmaisons. Online source. Vue.Draggable. <https://github.com/SortableJS/Vue.Draggable >. Read 7 April 2021.

5 REST API. Online source. Atlassian. <https://developer.atlassian.com/cloud/jira/platform/rest/v3/>. Read 9 April 2021.

6 Vue.js. Online source. <https://vuejs.org>. Read 4 April 2021.

7 The Good and the Bad of Vue.js Framework Programming. 2019. Online source. Altexsoft. <https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/>. Read 4 April 2021.

8 2020 Developer Survey. Online source. StackOverflow. <https://insights.stackoverflow.com/survey/2020#technology-web-frameworks-all-respondents2>. Read 4 April 2021.

9 Peter Mbanugo. 2018. Getting Familiar with Vue Devtools. Online source. Telerik. <https://www.telerik.com/blogs/getting-familiar-with-vue-devtools>. Read 11 April 2021.

10 Vuetify. Online source. <https://vuetifyjs.com/en/>. Read 4 April 2021.