LiU-ITN-TEK-A--20/021--SE

Data Visualization for User Experience - Implementation and Evaluation of a Customer Portal with Time and Project Management Data in Real Tim

Jacob Nyman

Viktor Sandberg

2020-06-15

Department of Science and Technology
Linköping University
SE-601 74 Norrköping, Sweden

Institutionen för teknik och naturvetenskap
Linköpings universitet
601 74 Norrköping

LiU-ITN-TEK-A--20/021--SE

Data Visualization for User Experience - Implementation and Evaluation of a Customer Portal with Time and Project Management Data in Real Tim

Examensarbete utfört i Medieteknik
vid Tekniska högskolan vid
Linköpings universitet

Jacob Nyman
Viktor Sandberg

Handledare Niklas Rönnberg
Examinator Camilla Forsell

Norrköping 2020-06-15

Linköping University | Department of Science and Technology
Master's thesis, 30 ECTS | Medieteknik
202020 | LIU-ITN/LITH-EX-A--2020/001--SE

Data visualization for user experience

- implementation and evaluation of a customer portal with time and project management data in real time

---

Datavisualisering för användarupplevelse
- implementering och utvärdering av kundportal för tids- och projektutvecklingsdata i realtid

Jacob Nyman
Viktor Sandberg

Supervisor : Niklas Rönnberg
Examiner : Camilla Forsell

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

In this report the work behind creating a web application for agile time management data for the company *Angry Creative AB* will be presented. The aim of the application is to create a customer portal with the intent to simplify the communication between Angry Creative and their customers. The research questions of this master thesis are based on how to communicate complex economic data to people who are not necessarily familiar with any types of visualization tools.

The work of this thesis started with a pre-study; where a literature study and data exploration was conducted. The work process started with creating a prototype to get a vision of what the end product might be. The prototype was used as a backbone in the development of the web application. The implementation of the web application was done in *React.js* and included the components from the prototype as well as new functionalities that became essential during the project's course. When all the desired functionality had been added, a usability test was conducted on the personnel of Angry Creative. The user test was initially intended to be held on the customers of Angry Creative, but due to the on going Covid-19 pandemic this was not a possibility.

The result of the master thesis is a web application in the form of a dashboard. This dashboard contained a number of graphs, interactions and filtering possibilities. All choices made regarding the dashboard and visual representations were based on scientific articles in related areas.

The implemented dashboard got a high *System Usability Scale* score, which can be translated to that it was easy and intuitive to use. Estimations and predictions that were made for the different graphs did not become as exact and reliable as had been hoped for. The graphs did not have the customizability that was needed to be able to create certain interaction techniques. The data was not as extensive as was originally promised either, but the visual representations that was created covered for the missing data.

Acknowledgments

We would like thank Angry Creative AB for giving us the opportunity to do our master thesis at their company. Jimmy Rosén and Malin Kylegård for helping us to get setup at the office and for constantly giving us feedback and suggestions about the development of the application.

We would also like to thank our supervisor Niklas Rönnberg for always being there helping with questions regarding the thesis as a whole and giving feedback. As well as guiding us in the development of the dashboard.

Finally we would like to thank our examiner Camilla Forsell, who have also helped us during the processes with answering questions and giving relevant sources for the master thesis.

Contents

List of Figures

List of Tables

1 Introduction

This master thesis was conducted at the web agency *Angry Creative AB* during the spring of 2020.

1.1 Background

Relationships between companies are created everyday through the sales of services and products. These relationships can be crucial for both their growth and survival as a company. It is therefore important to maintain these relationships and see to that both parties get what they want. Relationships between client and seller can easily get broken up due too lack of communication. The parties may find better options elsewhere or do not think that they get what they want from the other party. Most of these problems can be solved through communication and transparency, with the help of technical tools this can be made easier.

1.2 Motivation

When a company is about to purchase a product or service there are multiple factors that affect the decision. The decision may revolve around how well known the company is, but usually it is the economic factor that decides if the deal goes through or not. Projects where a technical service or product are being developed are usually hard to set a fixed cost/rate on due to the unpredictable nature of those kinds of projects. This can be facilitated through an agile product development approach [1]

Agile workflow is something that is commonly used in software development today and allows for changes during the projects life cycle. Using this workflow allows for a greater transparency to be held, which means that the end product can be very different from what was initially planned. Allowing for changes is preferred during software development due to the lack of knowledge of both time scale and pricing in the beginning of the project. When conditions and goals changes, it is essential that there is a clear communication between company and client. The client often has a time frame that needs to be followed while at the same time it is difficult to be precise and accurate with time in agile software development due to new goals and functionalities that recurrently comes up. Due to the unpredictable changes in the time plan it is necessary to communicate regarding the current status of the project which

subsequently would allow for the client to know what is possible to implement during the time frame of the project when considering the budget.

To ease the communication between the two parties, a system (e.g a *dashboard*) can be developed that mediates information continuously throughout the project. Combining the systems for tickets and time reporting in a system that can display relevant information about the state of the project in real time for the client. This kind of interface can visualize time reports, what tickets are currently being worked on and estimate the time consumption for the client in an intuitive way. The portal would also give the client opportunities to see their previous ordered projects.

1.3 Aim

The aim of this master thesis is to create a customer portal for the company Angry Creative with the intent to simplify the communication between them and their clients. This simplification will lead to larger transparency between the companies which subsequently will strengthen their relation and trust. These relationships are important for Angry Creative as it will lead to better businesses and returning customers. The purpose of the portal is to present intuitive data visualizations so that people that do not have a direct affiliation with the project still can understand.

1.4 Research questions

1. How can an end price be estimated for an agile project with regard to new events and time changes?
2. How can interactions be implemented in an interface for showing agile project data?
3. How should economic data be visualized and with what techniques in order to make someone that is not familiar with the data able to understand the contents of it?
4. How should an interface be designed so that a client can quick and easily get an overview of the progression of a project?

1.5 Delimitations

The project will be centered around creating a dashboard for computers meaning that there will not be any intentional time spent on working with responsive user interfaces for smart phones and tablets. The development will only focus on newer browsers meaning that compatibility with older browsers such as *Internet Explorer* will not be considered. As this master thesis is during the time of the pandemic Covid-19, the user tests will be solely held for Angry Creative's personnel at a distance.

2 Theory

In order to be able to create a customer portal for Angry Creative extensive research has been done. The research areas that have been studied are: information visualization, dashboards and interaction design.

In this chapter relevant theory for creating a user friendly interface will be presented. To be able to create a customer portal with the focus on interaction design and information visualization, it is important to understand how to create such a portal. It needs to be customized so that anyone can use it whether a technical background or not. The research that has been done in these research areas will be presented in this chapter. It will also cover why these subjects are important for the master thesis.

2.1 Agile Development

Agile development is a software development mythology where the project's end goal is reached through smaller iterations that results in new components or functionalities building up to the final product. Each iteration, called a *sprint*, follow the phases of the System development life cycle (SDLC), that consist of Planning, Analysis, Design, Implementation, Testing & Integration and Maintenance. Working agile allows for quick changes in the project plan, fast development, as well as ensuring the quality of the project [2]. Figure 2.1 gives a visual representation of the life cycle.

Figure 2.1: Diagram of the system development life cycle.

A few agile development methods commonly used in companies today are *Scrum*, *Kanban*, *XP* among others, were all methods follow the same concept of the *Agile Software Development Manifesto* [3]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Which can be summarized with working well together in a team, is more important than the task or project the team is working on. Get the software to the customer is of highest priority, not to write an extensive documentation before any development has begun. Even though the contract is of most importance, working and communicating with the customer during the development can generate more value for both parties. Following the prior mentioned method, when new information is gained the team should dynamically update the development plan to take in to account for the new changes.

2.2 Time management

For software developing companies to succeed on the market, efficient and low-cost solutions needs to be adapted by their employees [4]. T. Macan [5] states that if people receives the perception that they have control over their own time, it will be both beneficial for their workflow as well as giving them job satisfaction. Working agile with time management can be done in different ways, but a common way for working with software development is to solve projects with the use of a task board (Kanban board).

A Kanban board is a virtual (or physical) board where different tasks (called *tickets*) are placed in different columns depending on their status. These columns differ from board to board, but usually consists of at least a column for "Planned tickets", "In progress tickets" and "Done tickets". A ticket on the board could be anything from helping co-workers to developing new functionality and it is up to the assigned developer to place the task in the correct column [6]. When working with a task it should be placed into the "In progress tickets" and when done placed into the "Done tickets" column.

The tickets are usually created in the beginning of each sprint from the specification of the product to be developed and then either assigned to a specific developer or just put out for anyone to take. Another common thing to do at this stage is to estimate the time of completion of the ticket [7]. Usually there is a priority on what tickets should be done, but for the most part the developer choose and have control over what tickets to do. There are a few popular Kanban software that helps companies with these matters and some of them are: *Jira*, *Trello*, and *Redmine Backlogs* [6].

2.3 Information visualization

Storage space becomes cheaper and easier to acquire every day which inevitably leads to larger databases and more stored information [8]. This information however, is usually stored in a way which is not understandable for a human being, and the data needs to be processed to make sense to both the human eye and mind [9]. This is were information visualization techniques becomes crucial. Yi et. al [10] states that there are two different important aspects when it comes to information visualizations; representation and interaction, among which will be discussed in this section.

2.3.1 Representation

There are several visual representations and techniques that can be used to represent data. Depending on what type of data that is analyzed, some representations and techniques are more suitable than others. For example, plots are good for simple x and y- values, maps are good for representing coordinates, and streamlines for showing timelines.

Line Charts

Line charts are one of the most versatile charts; drawing lines between data points to display quantitative information. The chart is usually divided into three different representations depending how the line is drawn between data points; *segmented*, *smooth* or *stepped*. Each with their own benefits and use cases. The Segmented line graph is when each data point is connected by a straight light. The Smooth line graph is on the contrary interpolated between each data point, giving a continuous curve. The Stepped line graph differs from the segmented line graph with each data point being displayed in the center of a horizontal line between the points [11]. Figure 2.2 shows these three types of the graph.

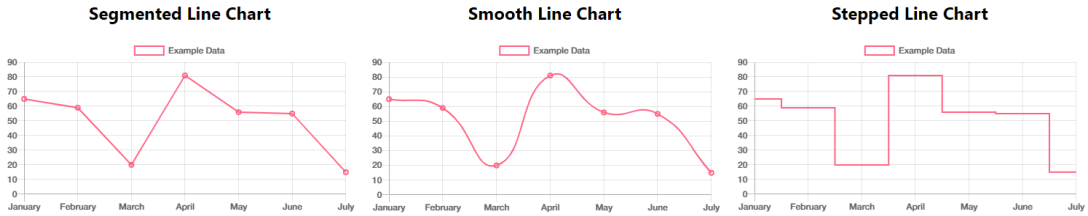


Figure 2.2: A representation of the three different types of line charts.

The stepped line chart if often used when plotting time discrete data where no values are defined in between data points. The smooth curve is commonly used when the exact value is not the main goal, but rather to give an approximation of the data. The Segmented line chart is the simplest of the three, used when to display the exact values and connection between the data points.

Bar Charts, Column Charts & Stacked Column Charts

If showing the exact value of the selected data is the objective, Bar Charts or Column Charts are a good choice. Due to the graphs characteristics, such as their geometric shape, they are trivial to interpret and it is easy to read their exact values. The charts are suited for displaying discrete data in conjunction with a category scale on the other axis [11]. The charts' differences are that Bar charts are represented with horizontal bars and Column charts with standing.

The Stacked Column Chart is a variant of these two charts where the bar is divided into different sections but still adding up to 100%. The graph is commonly used when the total is not the main focus, but the dividable parts compared to the total. Figure 2.3 shows an example of the bar chart when displaying both as standing and laying.

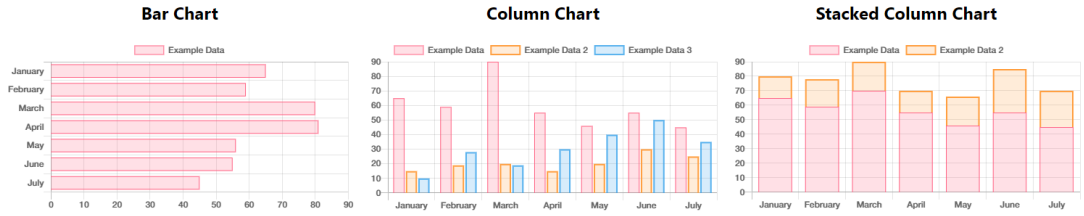


Figure 2.3: A representation of the three different types of bar charts.

To have less confusion in the remaining of the report, the Column chart will be referred to as Bar chart. This due to that their characteristics are almost the same, but it's more common to refer Column charts as Bar charts.

Burn Charts

A good way of displaying agile work progress is by the usage of a *Burn Up* or *Burn Down* charts. G. Dinwiddle states that burn charts are easy to comprehend, and make it quick to monitor the progress as well as simple to create a visual prediction of the project's scope [12] . A Burn up chart is built using multiple line graphs, plotting the current work progress against how much work is left. Figure 2.4 shows a simple version of the Burn up chart.

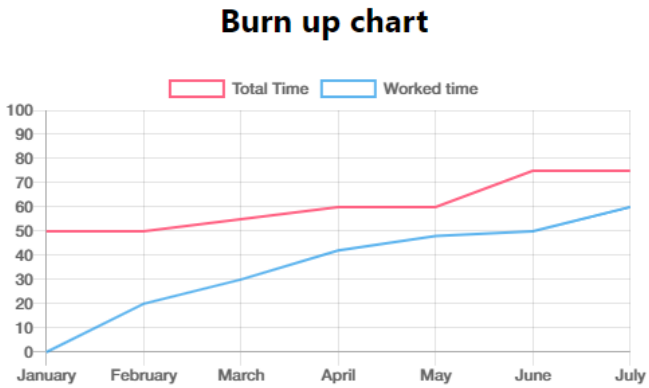


Figure 2.4: Burn up chart showing **Worked time** as a progress towards the total remaining work.

2.3.2 Interaction

One of the main goals for the customer portal that will be created in this project is to create a successful way of communicating with the user in an intuitive and interactive way. B. Shneiderman [13] talks about the *Visual Information Seeking Mantra* which are guidelines of how information should be displayed to achieve this type of interaction. The mantra's meaning is that the user should Overview first, then zoom and filter, then get details-on-demand:

Overview, is according to Shneiderman that the user should first be given a view of the entire data collection. This means that all the data should be presented to the user without any alterations to it. **Zoom** should be a feature for the user to use when looking at the visualized data. The zoom functionality let's the user get a better view of interesting data. **Filter** is a functionally much like the Zoom, that lets the user filter for items of interest whilst removing the unwanted data. **Details-on-demand** lets the user select a data point to get more information about it. A common way is to use a pop-up window to display the data corresponding to that point.

The mantra has laid a foundation for how information visualization should be and many researcher has used it in their work. For example, Yi et.al [10] states that in order for a data to be easy to understand, the user should be given 7 ways of interacting with the visualized data, much like B. Shneiderman's mantra [13]. Following these interactions techniques can help the user to get a deeper understanding about the data and these techniques are: **Select**: the user should be able to click on a data point. **Explore**: the user should be able to explore subsets of the data. **Reconfigure**: the user should be able to rearrange the data. **Encode**: the user should be able to choose to see another representation of the data. **Abstract/Elaborate**: the user should be able to set the amount of information given. **Filter**: the user should be presented with data based on chosen conditions, and **Connect** means that the user should be shown related data.

2.4 Dashboard

The dashboard terminology originates from the vehicle dashboard that displays the necessary information for the driver with an overview of the state of the vehicle. In a similar way a dashboard in a vehicle give you an overview, a performance dashboard can give you that for the data of a company. Dashboards are a great tool to improve the decision making processes within companies. They make it easier for decision makers at companies to get an overview of their data and help them draw better decisions and conclusions. Performance dashboards help remedy the *information overload*, which is when companies are presented with too much information to process. This is a problem many companies face with various different Business Intelligence (BI) and Enterprise Resource Planning (ERP) systems [14].

2.4.1 Design

Rahman et.al [15] states that dashboards are used differently throughout organisations. They state that there are three types of dashboards; strategic, tactical and operational. The developer need to be aware of these three different types of dashboards before starting the design of the dashboard. The first type of dashboard is the strategic dashboard which helps executives monitor the implementation of strategic objectives, communicate strategy and review performance. The second type of dashboard is the tactical dashboard which shows more detailed information than the operational dashboard. The tactical dashboards is used to manage specific projects or the performance of a department e.g.. The third type of dashboard is the operational dashboard which helps the front-line employees to view the current information needed to manage and control operational processes. It is important to have these three kinds of dashboards to battle information overload. Different users at a company have different needs regarding what they need to view in a dashboard. They need to view and have access

Table 2.1: Summary of 26 different reviews of dashboards that have been categorized into one of three types: strategic, tactical and operational.

Level (Frequency)	Purpose	Features (Frequency of appearance)
Strategic (4)	• Consistency – Improve business process – Track KPI • Monitor – Monitor organizational performance • Planning – To plan the organization future	• Visual Features – Fit single screen (2) – Grid overlay (1) • Functional Features – Graphical Presentation (Bar Chart, Pie Chart, Graph, Gauge Chart)(5) – Time horizon (1)
Tactical (7)	• Consistency – To standardize the service • Monitor – Self-monitoring the performance of the management. – Understand employee's performance – Summarize information by departmental. – Monitor trend over the period • Communication – Communicate with the operational level. • Analysis – Improve decision making among the departments.	• Visual Features – Fit Single Screen (2) • Functional Features – Graphical Presentation (Fusion, historical, bar, gauge chart) (5) – Drill down (2) – Scenario analysis (1) – Drag and drop (1) – Hide/flag component (1) – Report (2) – Alert mechanism (2) – Print (1) – icon (1)
Operational (14)	• Consistency – Increase speed and consistency of analysis. – For information transparency. • Monitor – Monitor individual or group information. – Monitor activity. – Monitor and detect relevant information. – Measure individual performance. • Communication – Provide feedback on their performance. – To extract information among the team members. • Analysis – Analyze learning analytics. – Analyze user's own information. – Analyze effects.	• Visual Features – Fit Single Screen (2) • Functional Features – Percentage indicator (2) – Graphical Presentation (bar, line, pie, network, spider, trend and gauge chart) (18) – Concept map (1) – Table (2) – Filter (2) – Badge (1) – Zoom (1) – Rating (1) – Calendar (1) – Alert mechanism (2)

to information related to their work. If the developer would try to create a dashboard that would fit all the different audiences for it at a company, this would lead to information overload. The overload would happen because it would need to suit too many different needs of its users.

Rahman et.al also states that the developer also needs to study the features that suit the dashboard's purpose before designing it. Rahman et.al did a review of 26 different dashboards to identify which type they belonged to and what features that are included in those implementations. This review can bee seen in Table 2.1 [15].

According to Noonpakdee et.al [16] by applying a dashboard to 40 different small and medium enterprises (SMEs) they could reduce their costs and ultimately help them make

better decisions. Dashboards are therefore an easy way for a company to make better decisions and to get a better overviews of their data.

2.4.2 Layout

A dashboard is a support system in the decision making process. It supports the decision making process by presenting the information in a certain way for the observer. Therefore a dashboard must be evaluated by the design aspects of it and how you interact with it [14]. Toasa et.al states that dashboards can provide a unique and powerful way of presenting information, but no matter how great the technology is, the dashboards success is a results of it's design and how well it displays information clear and direct.

2.4.3 Interface frameworks

To be able to create beautiful and intuitive user interfaces a good foundation is *intelligent borrowing*. This means that a user interface should be designed with reference to other peoples successful work, rather than coming up with new ideas. For example Apple and Google encourage that you develop applications and interfaces with their design language. If a user already knows a certain design language, they expect certain things from similar designs. Therefore, following a design language can ease the experience for the user [17]. An example of such a design is *Material design* which was designed by google and can be used for this exact purpose [18].

2.5 Effect Managing

Often when projects start, the descriptions about the expected benefit of the product are explained vaguely. This can often be due to the fact that the effects of the product have not been defined more than: "This new product will change our company" describes Ottersen and Balic in *Effektsyrning av IT: Nyttan som uppstår i användningen* or the English title: *Effect Managing IT* [19]. Effect managing is a concept that helps companies to structure the planning part of a project to help all parties in the development to understand the full process of the project. The main purpose of this concept is to understand that the benefit of the product arise when in use. With this in mind, a plan can be set in an early stage of the project and draw a map with all the important effects and how these are connected with each other. This map is called an *Effect Map* and it answers the following questions:

- What is the purpose and why should we build this project?
- Who are the people that can create the desired effects?
- What is needed to achieve these effects?
- How should the product be designed, and what other actions need to be taken to achieve this?

The creation of this map will help to keep transparency in the project and create an unanimity about the desired effects. Conclusively, this helps to create better user experiences because all parties have been taken into consideration.

2.6 User tests

A user interface cannot be determined to be good or bad unless people use it. Before a system is finalized user testing needs to be done. To be able to do this the users need tasks to perform. The best users to test are those who reflect the target audience of the application. If they

cannot be used as test users, it is important to keep in mind that the result may not be what the actual end users will perceive [17].

Lewis et. al [17] writes about the importance of an iterating process. The purpose of testing is not to prove the interface, but to improve the overall design. Therefore, using a *low fidelity prototype* of the full system is a good way of testing the initial idea. A low fidelity prototype is a simple version of the system with limited to none of the planned interaction, and some times even in the form of rough sketches [20]. After creating and testing the prototype, the process is then to continue working on the actual system.

During the test, it is important to stress that it is the system that is being tested and not the user. Selecting tasks for the users to test should reflect what real tasks should be like. The testing of the system should be done early in the development process. During testing, if users do what is expected of them: let them keep doing so, but if they deviate it is important to record this as it is valuable data. It is valuable data as the reason for the test is to find a discrepancy between what was expected by the user and what they did do.

While doing user tests it is important to collect data. There are two types of data: process data and bottom-line data. Process data refers to observations of what the test users are doing and think while doing the tasks. Bottom-line data on the other hand is a summary of what happened during the tests such as how long the users took to complete the tasks [17].

Think aloud method is a very simple method where you ask the user while performing the task to talk about their thought process. The test giver is supposed to ask questions similar to what are you trying to do? Why did you do that? and other questions related to the work the test person is doing. The thinking-aloud method works with mock-ups as well as prototypes of a system.

Qualitative tests are tests that utilize a relatively small group of test users (5-10). These tests are usually done in one-one sessions where the test user is given tasks to perform on the system. It is also likely that the test user will be asked to use the think-aloud method. The test giver records the test user's behavior and responses to questions and how they are performing the tasks. The most important metrics for the test giver to record are issues. Another metric that can be important to collect are self-reported metrics such as the *System Usability Scale* (SUS) [21].

In a usability study testing five user gives the best benefit to cost ratio. This finds almost as many usability problems as testing more users would do according to Nielsen [22]. User tests are expensive and time consuming so there is no reason to be doing more tests than necessary. Nielsen made a case study where 83 cases of different number of tests users were done and came to the conclusion that 5 test users gave the best ratio of usability findings to test users.

Quantitative user tests are a good way to get numerical data about the user experience of the system. Qualitative user tests are often more widely used as they are less cost and time consuming, but they also give vaguer results in a data perspective. Quantitative users tests needs more participants often 4 times more than a qualitative user test. A quantitative user test is good to use as it gives a concrete number of the usability of the system and it also makes it easier to compare to previous versions of the system [23].

2.6.1 Standardized questionnaires for usability and workload

There are many standardized questionnaires that are being used today when evaluating products and software. These forms are used extensively and can be found on websites as popups or as question purposely asked when evaluating systems. The purpose of these questionnaires is to get a quantitative score, other than the subjective results from other user tests, so that the results can be compared and evaluated [24]. Two of these systems are the System Usability Scale (SUS) and the post-test questionnaire *NASA Task Load index* (NASA TLX).

System Usability Scale - SUS

One way of getting a metric on the usability of a product, system or service is to use the System Usability Scale. The system works by letting users answer a set of 10 simple questions about the person's subjective assessments of usability using the product, by giving a grade from 1 (strongly disagree) to 5 (strongly agree) [25]. In Appendix B.1 the questions used in the test are listed. When the answers have been collected, the final score can be calculated with the help of Equation 2.1.

$$\frac{((\sum Score_{even} - 1) + (\sum 5 - Score_{odd})) \cdot 2.5}{n} \tag{2.1}$$

Where $Score_{even}$ is the score of all even questions and $Score_{odd}$ is the score of all the odd questions received. Also, n is the number of participants that have taken the questionnaire.

The score inherited from the equation gives a value between 0 and 100. Figure 2.5 gives an overview of how the score is then graded, where an average usability is around 70 when converted using the graph [26].

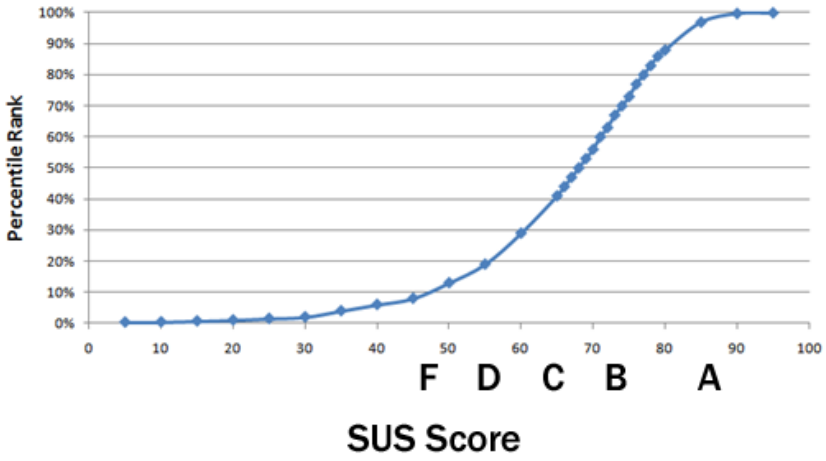


Figure 2.5: A graph used to get the final SUS evaluation score.

NASA Task Load Index

NASA TLX (NASA task load index) is a widely used questionnaire used to evaluate the user's subjective perceived workload when using a system, performing a task or team effectiveness [27]. The rating received from the test will give a score of how mental, physical, temporal and performance demanding the task was, as well as how much effort was needed and if the task caused frustration [28]. The test lets the user give a rating from 0 to 100 (%) in a 21 graded scale (See Appendix B.2). The test is split into 2 parts, where the first part is to conduct what categories has a majority of the workload. The second part is to evaluate the magnitude of each category. The categories are: mental demand, physical demand, temporal demand, performance, effort and frustration. When the values are summed together from all the conducted tests, it is possible to see where the workload of the system lies.

3 Method and design process

During the course of this project many crucial steps had to be accomplished during a period of 20 weeks. A plan was therefore developed to be able to deal with the different tasks at hand. The tasks that had to be done were: pre-study, data exploration, implementation of low fidelity prototype, implementation of the application, users tests, user test analysis and updates on the dashboard.

3.1 Planning

The master thesis spans over 20 weeks and there were many time consuming task that had to be done in order to succeed with the project. Setting up a plan for the project in the form of what to focus on during different time periods of the project was therefore necessary.

1. Pre-study - Effect managing and Data exploration
2. Low fidelity prototype
3. Implementation of application
4. User test of application
5. User test analysis and updates on the dashboard

3.2 Pre-study

During the beginning of the project a literature study was carried out in order to be able to back up design choices and to get a deeper understanding of the design elements of the application. The research fields that were investigated were Information visualization, Dashboards, UX-design and User tests. From the literature study many research articles and books were found that covered the targeted subjects. The literature study created a good basis to start developing and designing the application.

3.2.1 Effect managing

In an early stage of the project, an effect map of the project was made. This gave an overview of all the desired effects, as well as who and how these can be achieved. After meetings with the product owner, the map could be created so that all parties had the same idea about the effects of the project. This map became a backbone in the development of the product, as it was something to fall back on if decisions were unclear and needed to be made.

3.2.2 Data exploration

To be able to create the desired dashboard with the visual representations wanted by Angry Creative, data exploration had to be done. Angry Creative had already started to develop a platform where their time logging and ticket systems got integrated. From this system, calls to the server could be done to get some of the desired data. When creating the API calls from the backend to the frontend it was important to think about either having larger *endpoints* or smaller more specific ones. Where an endpoint is the name of a call to the server for receiving specific data. If smaller endpoints were developed the parsing of the data would be done on the backend and if the endpoints where larger the parsing had to be done on the frontend. At a later time in the project, it was realized that the backend system was not a finished product, but a prototype that did not have all the desired data for the planned visual data representations. Therefore a requirement specification was made for the personnel that worked with the system so that they knew exactly what was needed to be able to create the customer portal with the desired graphs and data representation. By doing the requirement specification, less time was needed to be spent discussing data and implementation choices with people responsible for the system at Angry Creative.

The required data was decided through a few discussions of the requirements of the customer portal with the product owner and the supervisor for the application at Angry Creative. Through these discussions an overview of what was needed in terms of data representations for application was established. After these discussions the data Angry Creative had through their systems Jira and Harvest was explored. In Figure 3.1 the data that was required to create the desired application can be seen.

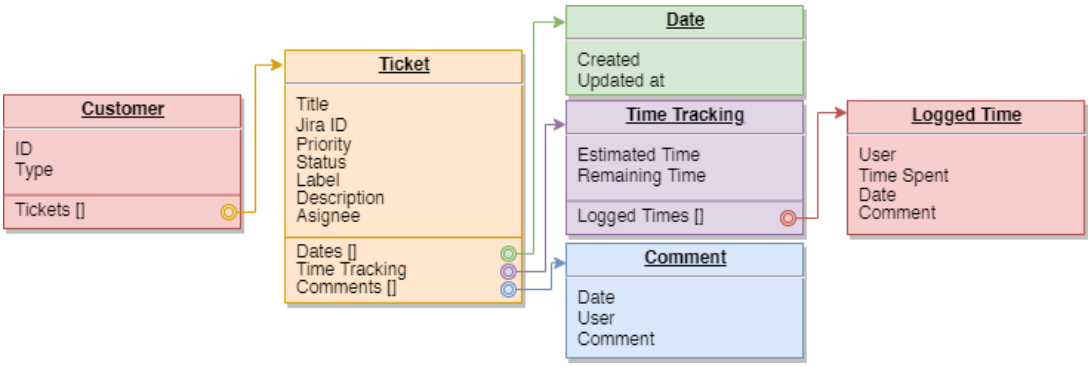


Figure 3.1: The desired structure of the data for the application.

3.3 Low fidelity prototype

A first prototype of the dashboard was created in the user experience design tool *Adobe XD*. The purpose for creating this prototype was to be able to visualize the ideas around the dashboard and to be able to see if a dashboard was the right way to go for this kind of application. The prototype could then be shown to the product owner and supervisor to get their opinions

regarding if the initial ideas of the application were the same as the supervisors. In the Figure 3.2 an overview of the prototype can be seen.

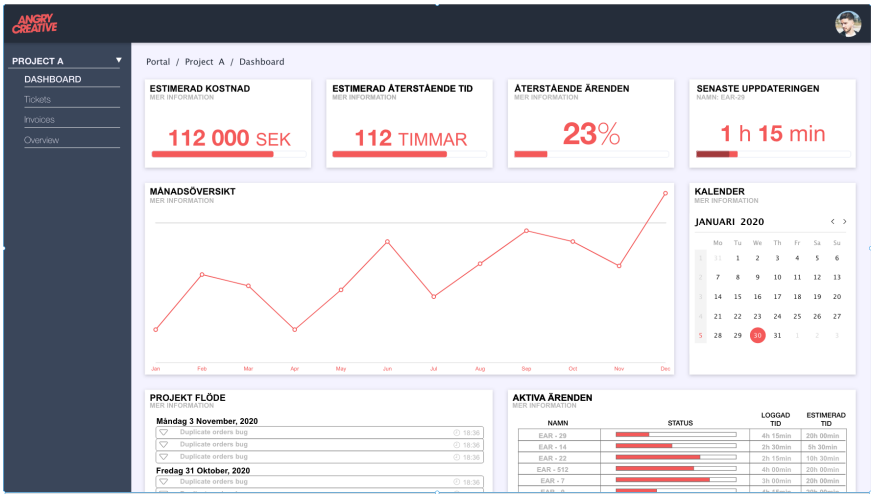


Figure 3.2: Image of the prototype dashboard created using Adobe XD.

3.3.1 Design choices

The design choices behind the dashboard was based around having a good balance of visual complexity as well as easy to digest visuals for delivering information [14]. In the Figure 3.2 the different visual aspects of the dashboard can be seen. In the prototype a line graph can be seen that is made to be the more visual complex part of the dashboard. The line graph will need to be analysed in order to get the information, rather than giving direct information and insight.

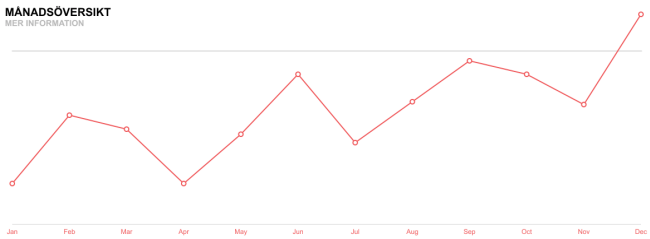


Figure 3.3: Image of the line chart in the *Adobe XD* prototype.

A part of the prototype dashboard that can be considered an easy way to deliver information can be seen in Figure 3.4. This is one of the four information boxes that are supposed to deliver information in a fast and efficient way to the user. The boxes show very explicit detailed information while the line graph shows more complexity. The information shown in the box can also be seen in the line graph but not as apparent. Yi et. al [10] means that by reconfiguring or by encoding the way the data is presented it will provide different perspectives on the data set. The prototype dashboard does not have the interactivity and details that the application later will have, but it shows an overview of what can be done the goal with the dashboard can be mediated. The goal with the dashboard is to be able to show the data set in different ways so that the user can get different perspectives of the data and by that being able to make better decisions regarding it.

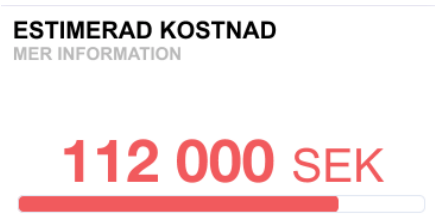


Figure 3.4: Image of one of the information boxes in the Adobe XD prototype.

Another feature of the dashboard that was seen as important was to be able have interactivity that would allow the user to drill down information to obtain further details [14]. This is also something that B. Shneiderman [13] and Yi et. al [10] brings up as important features when visualizing data. Shneiderman writes about *Details-on-demand* which means that by selecting an item or group of items that you can get details about when needed. Yi et. al also writes about this but as *Abstract/Elaborate* meaning that the user can go between different levels of detail in the data. Both by getting a more abstract overview as well as more detail. Therefore much of the focus of the application will be on details as well as abstraction. In the Figure 3.5 a detailed view of a ticket can be seen .

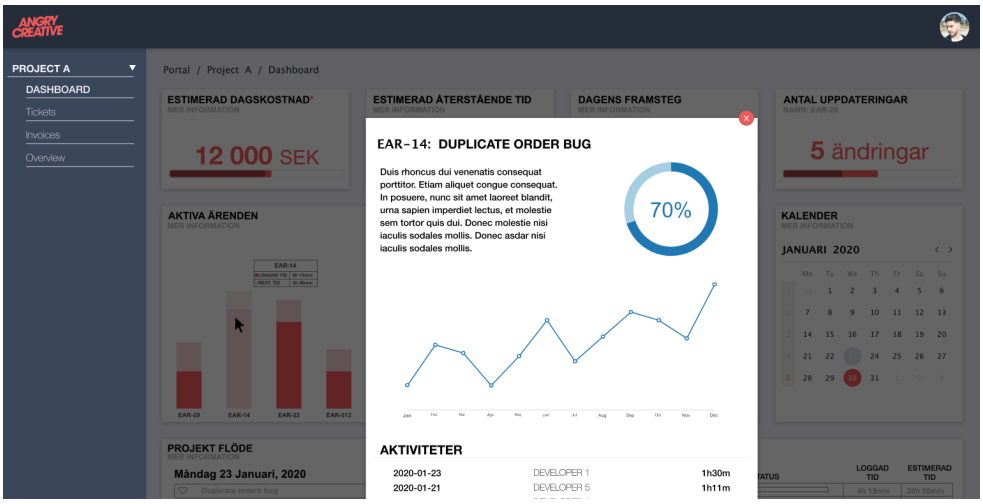


Figure 3.5: Image of the ticket detail view of the Adobe XD prototype.

By having the ticket detail view as can be seen in Figure 3.5 the user can easily go between an overview to a detail view by selecting a ticket. The user can also go back to the overview as well which Yi et. al and Shneiderman states as important features when visualizing data.

4 Implementation

The implementation of the project consisted of implementation on both the backend and the frontend. The following sections will explain the development process.

4.1 Backend development

The backend system of the application was created in the *Php* framework *Laravel*. This system integrated Angry Creatives ticket and time handling systems Jira and Harvest. When this master thesis started Angry Creative had already done much development on this system so that much of the needed functionality already existed. What still needed to be developed on the backend were endpoints for the suggested customer portal. The endpoints were designed in a way to give as much data as possible, which was not the most efficient solution due to that all the parsing had to be done on the frontend.

4.2 Frontend development

The dashboard was implemented using *React.js* as a frontend framework. Angry Creative chose the development to be in React because the framework is powerful but also easy to learn and simple to modify [29]. React is also a good choice for building web applications like dashboards due to the component architecture the language uses. This helps to keep the code structured, with all individual elements in the application being a component of its own with data and state. Where a component's state is its local dynamic variables that changes during its lifetime. When building web applications, there are other frameworks that uses component-like structures like React. *Angular* [30] and *Vue.js* [31] could also have been good candidates to use as frontend frameworks. Both frameworks are popular and with their own advantages. Angular is the oldest of the three and is maintained by Google and Vue.js the newest but has gained lots of new popularity in the last few years.

Using a global state management system, like *Redux*, was also decided. Other than each components having their own state, a global state management system could be an advantage when having larger applications. This way, when the global state changes, all components that uses that state re-renders with the new values [32].

4.3 Chart libraries

For the dashboard to be able to give a quick overview of the data, it is essential that a clear and intuitive diagram is given to the user. There are many libraries and plugins for creating charts in JavaScript and many of these are also wrapped to work with React. A study was made to analyze what plugins are most suitable for this project's needs. The purpose of this evaluation was to test and see how different chart frameworks work, and how easily customizable the plugins were. The tested libraries were *Chart.js*, *Recharts*, *C3.js* and *D3.js* which were all chosen because of their popularity, features and customizability. The following tests were made in isolated environments were all libraries could be tested with the same data and by using a simple line graph displaying the result.

4.3.1 D3.js

D3.js is the one of the largest information visualization plugins for web programming. The reason why it is the largest is because of the large amount of features as well as customizability. The D3 community is also large, with over 30.000 tickets on *StackOverflow*. Nevertheless, because of the customizability options, D3 is difficult to use and it needs a lot of boilerplate code in order to run. That is the reason why most of the interaction elements are quite difficult to implement. Usage with React is very similar to the regular usage of the D3 library.

4.3.2 Chart.js

Chart.js is a library that focuses on responsiveness, compatibility with browsers and style. The library is not developed for React, but with the React wrapper the code is almost the same as for vanilla JavaScript and it was easy to get up and running. The results are very stylish, animated and the out of the box interaction was versatile. Chart.js is the second most popular library compared to the others. The library also offers many options for customization.

4.3.3 Recharts

Recharts is the least popular library out of the evaluated ones, but despite the other ones Recharts is made explicitly for React. Because it is made for React, the implementation is simple. All extensions that the user want to add to the chart, is added like a component. The only thing that needs to be added is the actual data. The only downside to the library is the limited customizability. The library comes out of the box with great animation, interaction and stylish design.

4.3.4 C3.js

C3.js is a library with almost the same customizability as the D3 library, but no need to write the complex boilerplate code that is needed for D3. The chart is created from a *JSON* object with all the wanted settings and all the configuring happens in that object. There are not many animations out of the box with the C3 library, but the support for animations is large. The library is easy to set up and despite not being popular, is probably a great candidate for the dashboard.

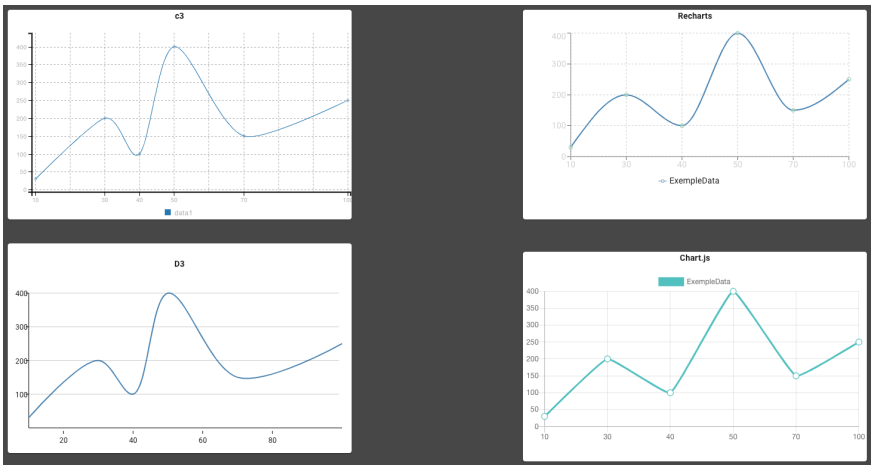


Figure 4.1: A comparison between the different graph frameworks, using the same data points.

4.3.5 Premium libraries

In this evaluation a few premium libraries were also researched, these plugins were *Highcharts*, *FusionCharts* and *amCharts*. What they all have in common is that they are extensive with a large variety pre-built charts to choose from, let alone their compatibility with older browser, add-ons and animations. The downside to them is that because of all their features they are premium plugins, meaning licences are needed to use them for financial purposes which is not an option in this project.

4.3.6 Plugin decision summary

From this evaluation Chart.js was chosen. It was chosen because of its popularity, intuitiveness, and the stylish results that the framework had out of the box. It also had many options in term of customization compared to e.g. Recharts. It did not have as much customizability as the D3 library, but that was not seen needed for the graphs that were created.

4.4 Dashboard components

In this section all the components implemented on the dashboard will presented with their purpose as well as how they were implemented.

4.4.1 Filtering Calendar

An essential part of the dashboard was to be able to filter the data to only show a selection. As stated in Section 2.3.2, B. Shneiderman [13] and Yi et. al [10] mentions the importance of filtering data to get a deeper understanding of the content. The filtering calendar made it possible to select a range between two weeks of the project to update the whole dashboard to only show data from that time span. Due to the projects' differences and lengths, it was important to implement different feedback systems for the the filtering. Because some weeks did not have any time logs on tickets but only project managing, the graphs would become empty if such filtering was done. To avoid confusion, a feedback messaging system was implemented to give the user information about that type of filtering. The system gave the user possibility to reset the filter as well as stating that the current filter was not containing any logged ticket data. The filter calendar was also supplemented with another messaging

system, noticing the user when any filter had been done i form of a dialog appearing the bottom left corner.

4.4.2 Information Boxes

The information boxes were implemented on the dashboard to give a quick overview of the project. The boxes also provided an easy way to get more information about the projects current cost, remaining time and latest update. This was an important part, because as previously mentioned in section 3.3.1 B. Shneiderman [13] and Yi et. al [10] states the importance of details on demand.

4.4.3 Budget Graph

The most important individual part of the dashboard was the budget graph as it would be the best indicator for the customers of Angry Creative to see where the end price of the project would result in. To be able to make such estimation, it was needed to analyze the logged time of a ticket when it was declared closed, because no data was included in the data about ticket status. The budget graph had 4 different lines. *High projection*, *Estimation*, *Actual* and *low projection*. The first iteration of the budget graph can be seen in Figure 4.2.

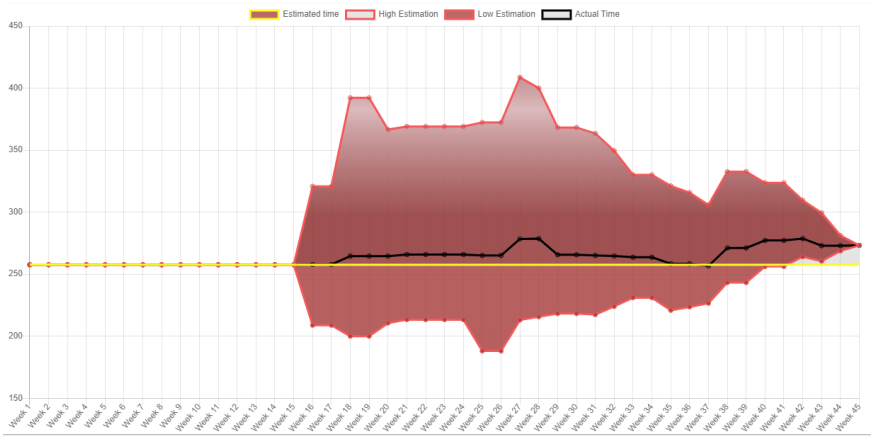


Figure 4.2: An image showing the budget graph of one of Angry Creatives prior projects.

The Estimation line is calculated through the sum of the estimated ticket times. This line can change depending on the scope of the project and the amounts of tickets the project have at a certain time. With this line the customer can clearly see the estimated budget of the project and what the ideal end cost would be.

The Actual time line is calculated from the estimation line. It changes when a ticket is closed and the tickets closing time is then added or subtracted, depending on if it was over or underestimated. When the project is done the Actual line will show exactly what the project ended up costing. As it become more accurate depending on how many tickets are closed.

The High projection line is calculated through the average overestimation time and multiplying that with the amount of open tickets. By calculating the high estimation this way it will be more accurate the more tickets that close. This is because there will be more data to calculate the average overestimation and there will be less open tickets to multiply with. Low estimation line is calculated in the same way as the High estimation, but instead of overestimation it will calculate the average underestimation.

4.4.4 Progression Graph

The progression graph showed the progression of the project. This graph was created so that Angry Creative's customers could see the progress of the project during a certain week and compare it to prior weeks. The graph also contained estimations for the project based on the worst, average and best week. These lines would help the customer get an insight regarding how the project was going other than looking at the logged time for a specific week. The progression graph can be seen in Figure 4.3.

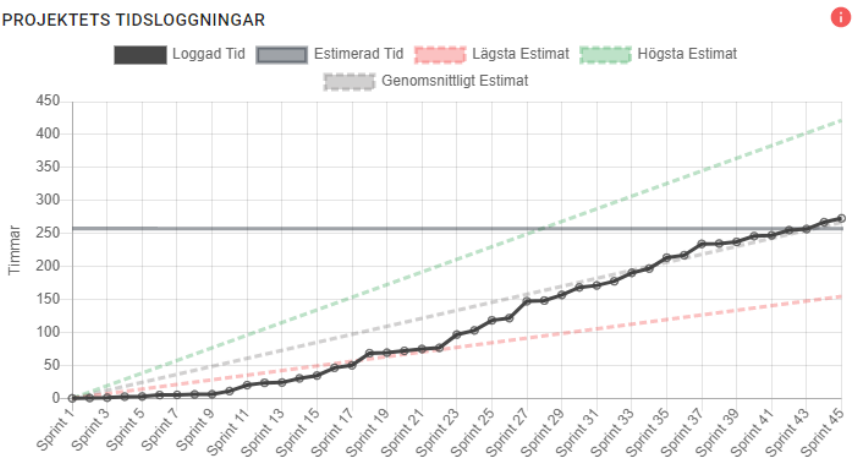


Figure 4.3: An image showing the first iteration of the progression graph of one of Angry Creatives prior projects.

The *Loggad Tid* line in the graph showed the logged time for each week and how much time had been logged so far. This line was import to include so that the customer understand how much time had been logged during a specific week and it clearly shows the progression of the project. This line in relation to the *Estimerad tid* line gave the customer insights regarding when the project would finish.

The estimation lines are based on as previously mentioned the worst, average and best week. These lines are used to get an understanding how the Loggad Tid line is doing during the current week. These lines also relate to the Estimerad tid line as at the cross section with the Estimerad Tid line will show when the project would be done if it followed the estimation lines.

4.4.5 Ticket Graph

To be able to get an insight regarding what time was logged on what tickets, a bar chart showing each ticket was implemented. Each bar in this graph represented a ticket and to be able to show the status of each ticket the bar graph needed to be stacked. All bars could then show both the estimated and logged time of a ticket. By showing this the user got a quick overview of the status of the ticket. If the bar chart only showed one bar for each ticket it meant that the ticket had gone over estimation. While if the bar was stacked the user could see how many hours has been logged on a ticket in relation to the estimated time.

The reason behind only showing one bar if the ticket was overestimated was so that the customer would not get worried if it saw all tickets that had gone over estimation at a quick glance. The bar still showed how much over estimation the ticket was if the user hovered over a specific bar. As the user needed to actually look at the specific ticket to know it was over estimation some of the concerns were removed.

The bar chart can also be sorted so the user can get a better overview of the tickets. The bar graph can be sorted either by logged time in acceding/descending order, alphabetically, by date or by estimated time. By letting the user sort the graph by these categories the user can get an even better overview of the information they are looking for.

When the user use the Filtering calendar 4.4.1 the data of the ticket graph will also change after that selection. Another stacked bar will show if the there is time logged on that ticket before the filtered weeks. A ticket will then show how much time was logged before and under the filtered weeks as well as how it relates to its estimated time.

4.4.6 Ticket Table

The ticket table was added as a supplement to the bar chart. The table lists all the tickets in the project, as well as showing the current ticket status with data such as ticket name, latest update, how much time that has been logged, the time estimate for the ticket as well as having a progress bar showing the remaining estimated time. Another strength of the table was that the user can filter by searching for a ticket, sort each column by different data and expand each ticket for more information.

4.4.7 Ticket View

An important part of the dashboard was to let the user be able to see the specifics of certain tickets. This was an important part to implement as it would make it possible for the user to get a deeper knowledge about the ticket rather than on only on an abstract level. The information that was displayed in the ticket view was the description of the ticket as well as the times logged in various forms. As previously mentioned Yi et. al [10] means that by abstract and elaborate the way the data is presented it will provide different perspectives on the data. This was something that was thought about especially on the ticket view as being able to see the data in different forms would make it easier for the user to get an overview as well as insight about the ticket data.

4.4.8 Full screen mode

In order to give the user even more choices in how to view the data a full screen mode was added. This full screen mode was added to all the graphs and the table. This was done to give the user another perspective of the data and a more focused one. Yi et. al [10] and B. Shneiderman [13] both states the importance of this feature. Yi et. al writes about the importance of being able to Select and Abstract/Elaborate graphs as important interaction features a graph should have. B. Shneiderman also writes about the same subjects but in the form of being able to Zoom and Extract data in order to get better insight from the data.

5 Evaluation of the dashboard

The evaluation of the dashboard was done by conducting a user test for the employees at Angry Creative, as well as analysing the results from the test. This chapter will explain that process and how it was done.

5.1 User tests

To be able to evaluate the functionality and usability of the application a usability test was performed. This usability test was done with eight employees at Angry Creative. The employees that participated in the tests had different roles at the company. Four of the participants were product owners and their feedback was seen as especially important as they work closely with the customers and give them information regarding their projects. They were also important to do tests on as they work with the data around projects on a daily basis and know what is important information for the customer and not. Two developers were interviewed to get an insight regarding how a developer experienced the application. One of the test participants was a UX-designer and was tested to get insights regarding the interaction design and design aspects of the application. The last participant was an Agile Coach with a background as a developer. All the test participants were chosen to give certain insights regarding the application.

The usability test was done using the thinking-aloud method. The tests were conducted over video communication due to the ongoing Covid-19 pandemic. The test was conducted by letting the test participant share their screen while they were looking at the application and performing various tasks. The projects members had different roles during the user tests. One gave out tasks and asked questions while the other team member wrote down the participants thoughts and answers. The tests were performed on the participants one at a time and the test took approximately 45-60 minutes. The test started with the test leader explaining the purpose behind the test and that it was a test of the application and not a test of the participant. The participant was then asked to think aloud while doing the different tasks. The first tasks of the usability test were intended for the participant to familiarize themselves with the application. During the first tasks the user got to try all different functionality of the application. The reason behind this was that a real user of the application would use it more than once and would be familiar with how it worked.

The reason behind having user tests on participants with different skill sets was that it would give insights regarding different aspects of the application. E.g the UX-designer would give insights regarding color and interaction choices. To be able to get these different insights regarding the application all follow-up questions to the tasks were asked in regard to the participants answers and thoughts regarding it. After the usability test, general questions regarding the application was asked in order to get the overall impression the participant had of the application.

5.2 SUS - The System Usability Scale

In addition to the main user tests (see section 5.1), the SUS system was also used to obtain a quantitative score from each participant about the usability of the dashboard. The reason why SUS was used and not the NASA TLX was because at this stage of the development, the workload was not of interest but rather how intuitive and quick the users comprehended the dashboard (see section 2.6.1). The tests were held after the completion of the main user test and the interviewee was asked to fill out a questionnaire with the original SUS questions without the interviewers interfering. These questions can been seen in Appendix B.1. After all interviewees had answered the questionnaire, the usability score was then calculated to achieve a final score.

5.3 Analysis of the user tests

The analysis was conducted by looking through the answers of the tasks from the user tests, rather than looking at the SUS score. The general questions gave an overview of how the user experienced the application more in detail, compared SUS. The analysis was done by selecting the relevance of the answers given by the users. Some answers, eg. compliments, did not weight as much as answers that pointed out specific details. The answers from the questionnaires was not used explicitly for finding flaws in the design but rather as a measurement to ensure the quality of product.

5.4 Design proposal based on analysis of the user tests

The last part of the project was to go through the analysis retrieved from the user tests. Then, conduct a plan for what ideas and implementations there were time to add to the final product, as well as what to put aside for future work. The feedback was added to a backlog and evaluated on how much time each task would take to implement, also evaluating their importance for the last iteration of the dashboard. After the evaluation, a last development process was engaged. This process was mainly about doing small changes to the dashboard to implement smaller features, but also refactoring of the code.

6 Results

In this chapter the results of this master thesis will be presented. The main outcome of the project is a web application in form of a dashboard. This chapter contains the results of the pre-study for the project, the implementation of the web application as well as the results of user tests.

6.1 Pre-study

The goal of the pre-study was to get as prepared as possible to be able to create the application Angry Creative wanted. An effect map was created for the project which can be seen in Figure 6.1. This map made it possible to see the different needs of the application and became the backbone for both the decision making as well as the development of the application.

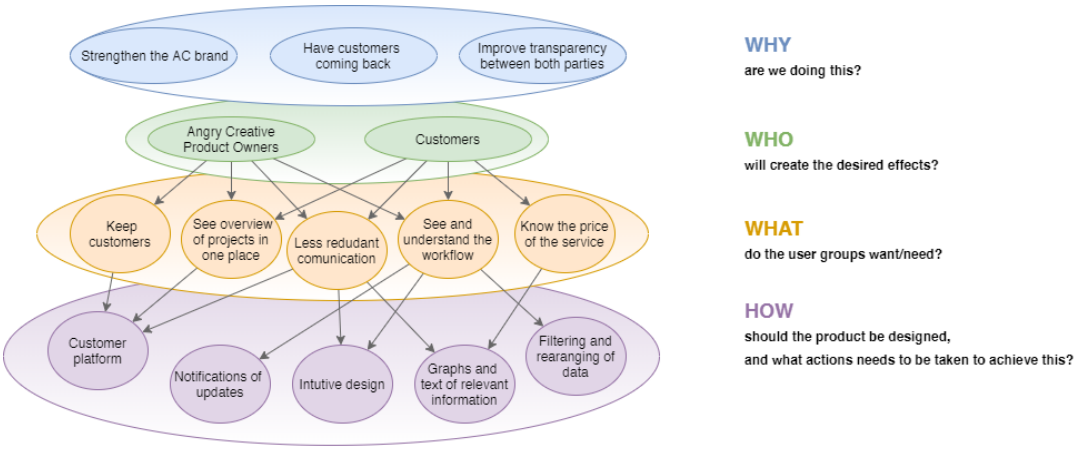


Figure 6.1: The effect map of the project with answers to the effect managing questions.

The scientific research made about the implementation was also a big factor for the development. Many resources found, especially *The 7 interaction techniques* explained by Yi. et al

[10] and Schneirderman's *Visual Information Seeking Mantra* [13] written about in section 2.3.2 became a staple of the development. Having these references made it easy to know where to take the development next and to know what type of interactions should be added to the dashboard.

Data exploration made it possible to see what could be done with the data Angry Creative had. In Figure 3.1 the desired data of the application can be seen. Having this structure simplified the process of both implementation and parsing of the data.

6.2 Dashboard

One of the main parts of this project was to build a web application in form of a dashboard. Before the implementation in code was done, a prototype was made in Adobe XD (see Figure 3.2). After the prototype was been created, it was presented to the CEO of Angry Creative to see if the visions aligned as well as to get feedback. The prototype was static, but all the interactions was planned out for the implementation of the actual dashboard and these were explained in the presentation for the CEO.

6.2.1 Web application

The web application that was created for Angry Creative was a dashboard that used project management data from Angry Creatives ticket and time management programs Jira and Harvest to show that data in an informative and intuitive way. The dashboard contained multiple visual representations of the data; information boxes for general information about the project and detailed information with five different components to visualize the data. These components were a budget graph, a progression graph, a ticket bar chart, a ticket table and a specific ticket view (see Figure 6.2). The figures shown in this section are the results of the dashboard after the user tests.

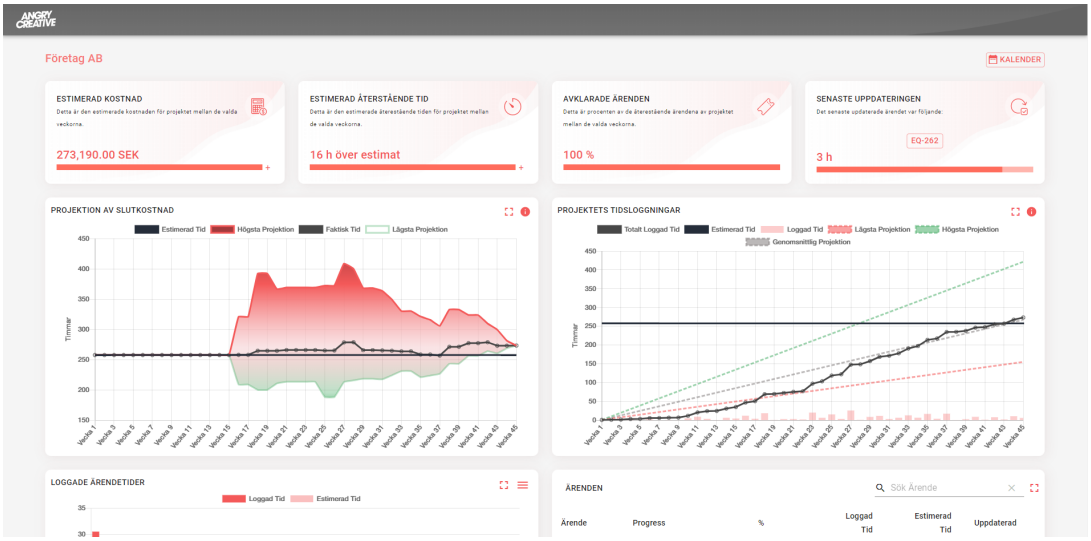


Figure 6.2: An overview of the finished dashboard.

Information boxes

The information boxes presented general information about the project. This made it easy for the user to get a quick overview of the current status of the project (see Figure 6.3).

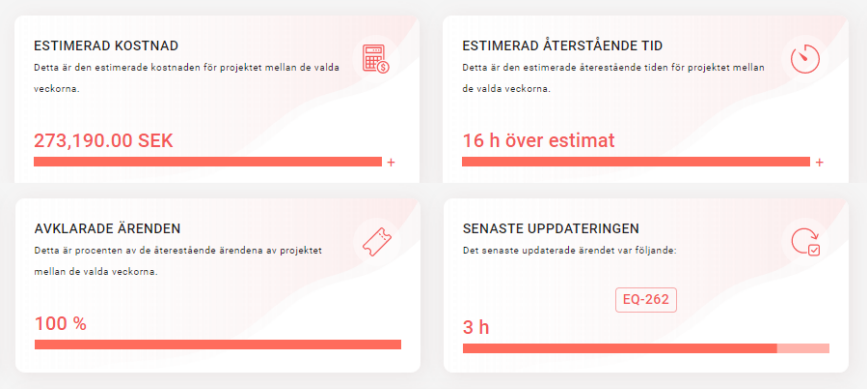


Figure 6.3: The completed information boxes.

Filtering calendar

To be able to see different time periods of the project and specific week's data, a filtering calendar was implemented. This calendar made it possible for the user to zoom in on the data in different time intervals to see it more clearly (see Figure 6.4).

Figure 6.4: The completed filter calendar.

Budget graph

The most important component on the dashboard was the budget graph. It gave the user a projection of the end cost with the help of estimations. The graph showed an area/interval of the weeks current estimated time interval and the interval became more accurate the more tickets got closed (see Figure 6.5).

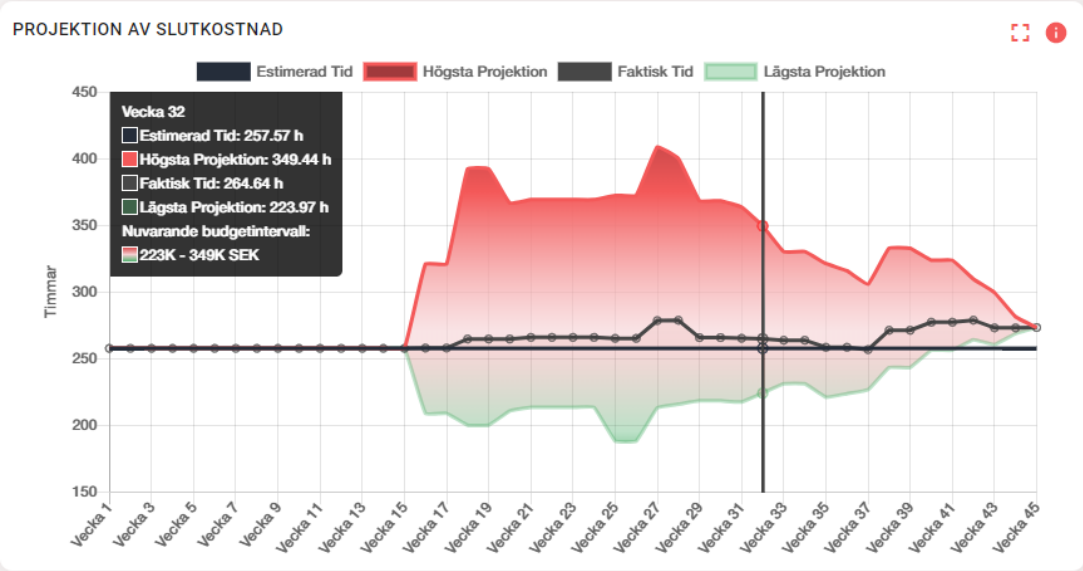


Figure 6.5: The projection graph showing the interval during week 32 of a project.

Progression graph

The progression graph gave the user a quick overview of the current state of project. It showed how much time had been logged each week and how it related to the estimated time for the project. It also showed projections on where the project could finish according to the project's prior progression (see Figure 6.6).

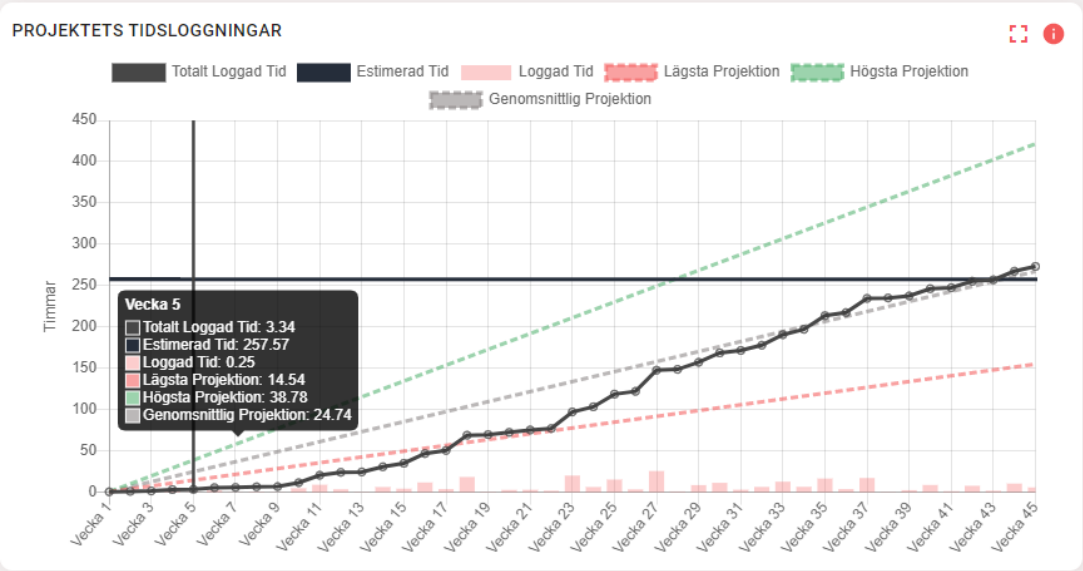


Figure 6.6: The progression graph showing the progress of a project over the weeks it were conducted.

Ticket graph

All of Angry Creative's projects are ticket based and the estimated time of a project is based on the tickets in a project. Due to the central part tickets have for Angry Creative's projects a ticket graph was created. The ticket graph showed the estimated and logged time of every ticket. The graph could also be sorted in various ways such as alphabetically, ascending, descending, by estimated time and by date. This made it possible for the user to get specific overviews of all tickets in the project (see Figure 6.7).

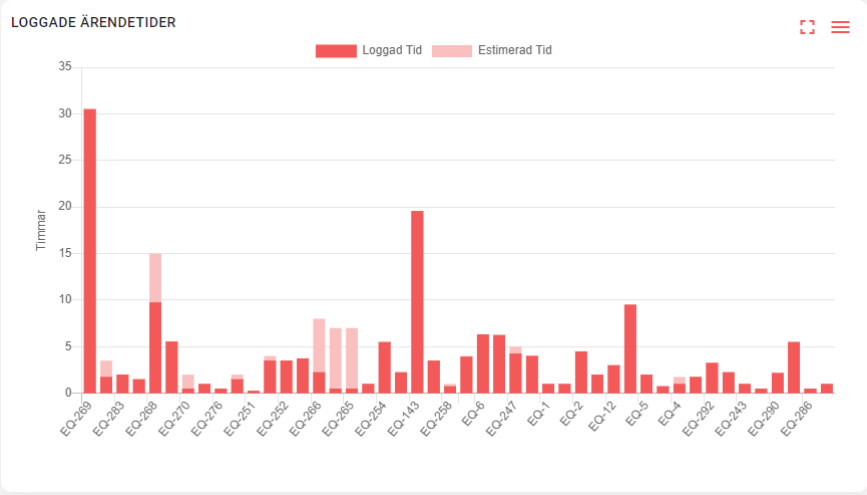


Figure 6.7: The ticket graph showing all tickets as bars.

Ticket table

The ticket graph gave a good overview of each ticket at a quick glance, but detailed information about each ticket is just as important, therefore a ticket table was created. The ticket table could also be filtered in a number of ways to give the user the perspective and information they were looking for. The user could also search for specific tickets (see Figure 6.8).

ÄRENDEN

Sök Ärende

Ärende	Progress	%	Loggad Tid	Estimerad Tid	Uppdaterad
EQ-269		100% +	30.50	16	2019-05-03
EQ-273		50%	1.75	3.5	2019-05-02
EQ-283		100%	2	2	2019-04-30
EQ-285		100%	1.50	1.5	2019-04-25
EQ-268		65%	9.75	15	2019-04-18
EQ-263		100% +	5.55	2	2019-05-13
EQ-270		25%	0.50	2	2019-04-11
EQ-278		100%	1	1	2019-04-10

8 rows 1-8 of 46

Figure 6.8: The ticket table of the completed dashboard displaying all tickets of the project.

Ticket view

The ticket table could not show all the information about the tickets, e.g. description, title and the progression for a ticket over the weeks it was worked on, therefore a specific ticket view was developed. This view showed all information about a ticket; when it was worked on, what the task was and how it related to it's estimation time (see Figure 6.9).

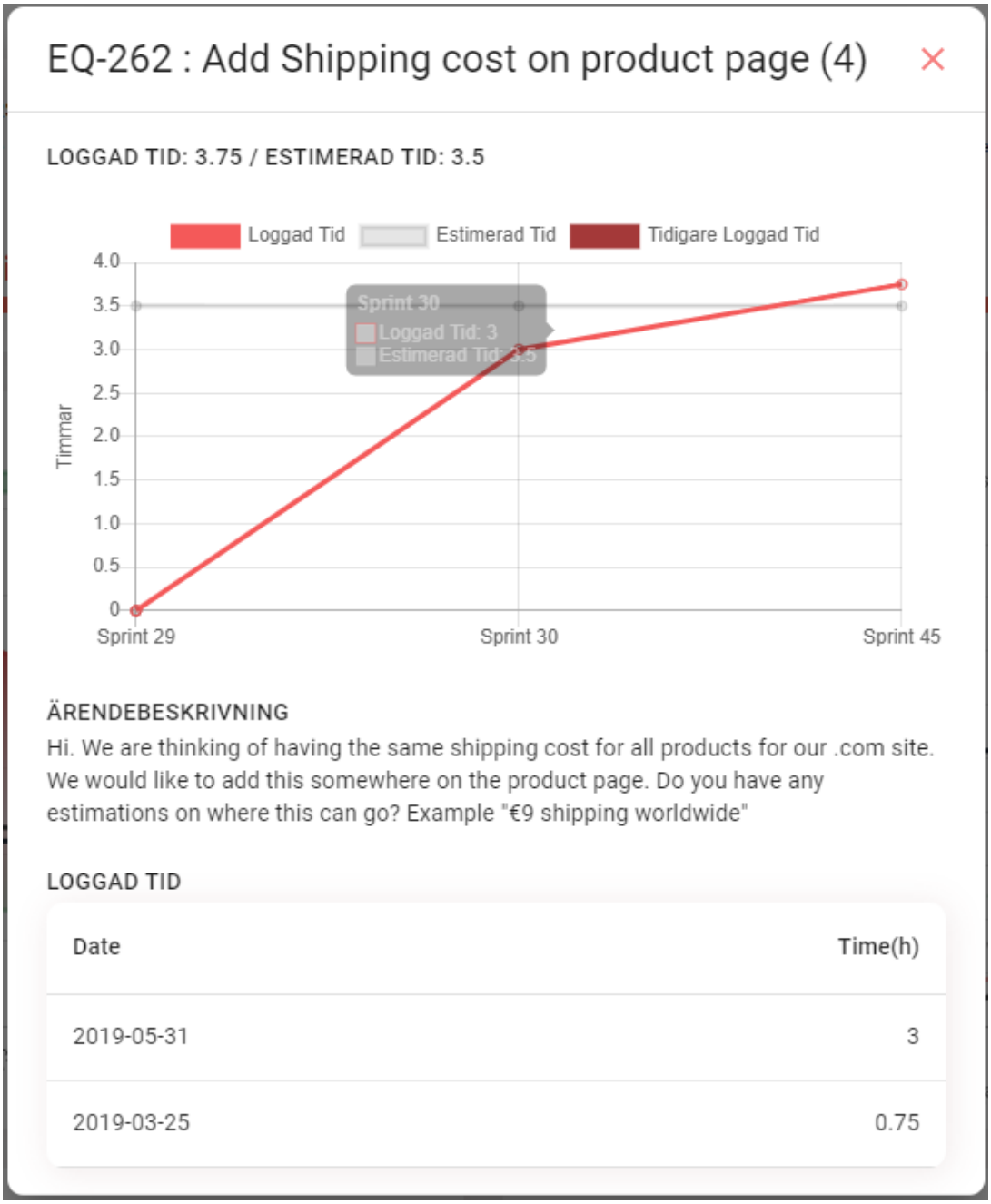


Figure 6.9: A view of a specific ticket on the web application.

6.3 User evaluation of the dashboard

The dashboard was evaluated with user tests in the form of a usability test. The user test was held remote for 2 Developers, 1 UX-designer, 4 Product Owners and an Agile Coach, all from Angry Creative. All the users had different knowledge about information visualization and experiences with dashboards. The tests gave insight into what the dashboard was missing, what part of the interaction was intuitive as well as suggestions for future implementations. A quantitative score of the usability of the dashboard was also obtained from the tests.

6.3.1 User tests

The results from the user tests were summarized from the notes that were written during each individual user test. These notes contained the user's thought process of the task they were doing, as well as suggestions for future work. The questions asked during the user test can be found in Appendix A.

A majority of the user tests showed a positive user experience. Most of the interactions and design choices that were made on the web application were understood by the users and there were only a few tasks during the test that the users had difficulty completing. Most of the users had no problems describing their thought process during their tests, but sometimes the interviewers had to intervene and remind the users to explain their thinking. This usually happened when the task was complicated and the focus was more on completing the task. However, these thoughts were often the most important ones to find flaws in the design and to get new insights of the system.

A positive tendency that was concluded from the user tests showed that everyone found most of the components on the dashboard to be intuitive and that it was easy to get a quick overview of a project's status. The design was complemented to be stylish and to follow the Angry Creative's graphical profile as well as having elements from Material Design (See 2.4.3).

Another trend that could be seen from the user tests was that the dashboard had minor issues with the coherence of what elements could be interacted with or not. These thoughts were often expressed by the users when they were asked to find another way of completing the same task that they just finished, but in another way. Some users also expressed minor issues with the color choices on the dashboard. These complaints were mainly about the colors of the lines in the graphs being too transparent or ambiguous. It was also expressed by some that the overall dashboard color was too red.

When all the user tests had been conducted and summarized, the comments and suggestions were shaped into tasks that were placed in a backlog. This backlog was a collection of all the things that were planned to implement before the master thesis was over and also the tasks that would be considered as future work.

6.3.2 System usability scale

Once all the user tests were done the result of the system usability test could be calculated. The results from the questionnaire is summarized in Table 6.1.

Table 6.1: The summarized answers from the system usability test's questionnaires. The numbers in the table correspond to the amount and percentage of the people who chose that answer.

	1 (Strongly disagree)	2	3	4	*5 (Strongly Agree)*
I think that I would like to use this system frequently.			**1 (12.5%)**	**5 (62.5%)**	**2 (25%)**
I found the system unnecessarily complex.	**3 (37.5%)**	**5 (62.5%)**			
I thought the system was easy to use.			**1 (12.5%)**	**5 (62.5%)**	**2 (25%)**
I think that i would need the support of a technical person to be able to use this system.	**7 (87.5%)**		**1 (12.5%)**		
I found the various functions in this system were well integrated.				**6 (75%)**	**2 (25%)**
I thought there was too much inconsistency in this system.	**4 (50%)**	**3 (37.5%)**	**1 (12.5%)**		
I would imagine that most people would learn to use this system very quickly.				**1 (12.5%)**	**7 (87.5%)**
I found the system very cumbersome to use.	**6 (75%)**	**2 (25%)**			
I felt very confident using the system.		**1 (12.5%)**		**3 (37.5%)**	**4 (50%)**
I needed to learn a lot of things before I could get going with this system.	**5 (62.5%)**	**3 (37.5%)**			

The final result for the usability was calculated to a value of **87** which is classed to be at the top rank (see Figure 2.5). This means that an intuitive web application had been created.

6.3.3 User tests' feedback

When evaluating the results from the user tests a few changes were decided to implement on the dashboard to improve the overall quality. The following subsections will explain what changes were done and the reason they were implemented.

Filter calendar

The filter calendar was supplemented with the dates of each week in a tool-tip. The interaction of the filter calendar was also changed to be as a pop-up when pressing a button, placed at the top of the dashboard. The reason for moving the calendar, was because the prior placement made some users confused that it only altered that row in the dashboard. With the new placement it was easier to understand that it affects the whole dashboard. The dates were added because a majority of the users wanted to know what dates each week in the calendar corresponded to.

Full Screen view

The graphs and the ticket table was complemented with a full screen view to be able to see more details of each graph. This was added because many users were mentioning that it was sometimes difficult to see the data in the graphs. This was also strengthen by B. Shneiderman [13] in the Visual Information Seeking Mantra that the user should be given the possibility to zoom.

Colors

The colors on the graphs were changed to be more discernible with the rest of the colors to be easier to read and understand. This change was one of the most commented things from the user tests. Many users were complaining that the colors were too similar and not distinguishable from each other. To be able to see a bigger difference in the Budget graph, the gradient was changed to include green in the lower end of the cost. The progression graph's estimation lines were also made more protruded with more opacity and saturation.

Ticket view

The ticket view window was enhanced with more feature to have a more coherent content with the rest of the dashboard. This was mainly by adding a feature to show prior logged time in the progression graph when a filtering had been done. This was added because some confusions were raised during the tests when the users were asked to filter the data and look up more information about the tickets.

Progression graph

The progression graph was supplemented with bars to give a representation of how much logged time there was each week. This was a point that many users mentioned because at the time of the user tests, the graph only displayed the total logged time at the current week. This change helped to give the graph a better representation of the current week's status.

Ticket bar chart

The ticket bar chart was made more coherent with the ticket table with an addition of two more sorting functions. These sorting functions helped the users to distinguish the tickets from each other even more. The functions added were sort by date and sort by estimated time. These were added because after the users had interacted with the ticket table, they commented that the way the sorting ought to be the same between both components.

Other changes

The last changes that were made from the user tests were minor but important. Some users commented that a few of the component's titles were misleading and not correct to what they referred to. This was that the labels in the budget graph was named *estimation* but in reality were a *projection* of the final cost. Another thing pointed out was that the data points that were displayed in the graphs tool-tips were missing suffixes, which was added in the last implementation.

7 Discussion

In this chapter the result and method of the project will be discussed. This chapter will also discuss the work in a wider context and discuss the choices that were made through out the project.

7.1 Results

The web application had all the features and the functionality that was proposed and that were included in the first prototype. All the visualization elements followed the The 7 interaction techniques explained by Yi. et al [10] and Schneirderman's Visual Information Seeking Mantra [13] written about in section 2.3.2. Despite following these guidelines, the result of this project needs to be analyzed. These sections will discuss how the results was compared to what was believed to be the outcome of this project.

7.1.1 Web application

The dashboard had all the functionality that was possible to create with the data, but due to lack of data it was limited. The different chart components on the dashboard were quite simple in their functionality, as they did not have a brush or a zoom function. The reason behind this was that it might would have complicated the information for the users. However, other interaction techniques were implemented in form of a filtering calendar and a full screen mode to complement for the loss of functionality. These interaction techniques are similar to the techniques used on websites the user might use in their day to day life. The system was intended to be easy and intuitive to use even for users with limited experience of visualizations and similar systems.

The resulting dashboard used the existing data to its full capacity, but in the beginning of the project more data was promised. Comments, ticket status and which developer worked on each ticket was data thought to be accessible. This data would have meant a great difference in the visual data representations both for predictions models and features. These features are things that will be implemented to the backend systems in the future. Even if it could not have been done in the scope of this master thesis, this will be something Angry Creative can implement on their own at a later point.

The performance of the system and code structure are things that could have been done better. When building a system that is nested in the form of its data handling, it is easy to both over complicate the process and excess use of unnecessary data. This could have been done better by handling the data parsing on the backend. This would have meant that many of the components would not need to re-parse the data multiple times and would have made the code both less complex and more understandable. If the data parsing had been done on the backend many smaller endpoints would have been created instead of the one large endpoint that fetched all the data which is the implementation at the moment. The smaller endpoints would mean that server calls would be done more often to get the resources needed for rendering the charts or when filtering the data. The advantages with smaller endpoints are that they create a less complex data flow and that they move the parsing from the frontend to the backend.

The budget graph was the most important part of the dashboard as it was this component the master thesis was based on. It showed what it was intended to visualize by projecting what the end cost would be at the end of a project. From the feedback of the user test it was given that the component was hard to understand at first glance. Another problematic part of the graph was that the end cost estimations were not accurate in the beginning of a project and showed numbers that could not be accepted by the customers of Angry Creative. Because of this inaccuracy it was of great importance that the data shown was visualized in an easy and intuitive way. The reason behind why many users found it hard to understand was because it was a new way of visualizing that kind of data. In order to help the users understand the charts better an information window was implemented which explained the data and how it was visualized. Most users understood the components better after reading this information and the general feedback was that when they understood what it was intended to show, the data was indeed visualized in an intuitive way.

A question that can be asked is if the information windows should be needed to understand a certain graph. In a best case scenario this should not be the case, but when using new ways of visualizing data it could become necessary in order for users to comprehend the data.

The test users had a hard time to understand the progression graph because of its many data lines. The users understood the basic concept of what the the graph was showing, i.e the progression of the project. However, they had a hard time interpreting what the estimation lines were intended to show. This meant that they had a hard time understanding what insights and conclusion they could draw with the help of them. An information window was also created for this graph to handle this feedback.

The estimation lines in the progression graph could also have been done better. In the resulting application they are simple in form of their estimations as they follow the most logged week, the average logged week and the least logged time week. These estimations were mostly created to give the user a view of when in time the project could be finished. To be able to create better estimations for this purpose, prior projects from Angry Creative and more data about each ticket would have been needed.

The design language of the application followed Google's Material Design and this was appreciated by the test users. The users found the design language recognisable and knew how to interact with most elements on the dashboard. Using a design language that the users already knew made the interactions on the dashboard easy to use as they felt familiar with the interface and it's interaction elements. This was also written about in section 2.4.3.

In summary many of desired features could not be implemented because of the lack of data and time. Time in the form of going through old projects and to use machine learning on them in order to get better projections. Lack of data in the form of the missing parameters e.g. comments and ticket status as previously mentioned. In regard to what the final result became it had all features that was proposed and followed the theory presented regarding information visualization. However, after the user tests it was realized that the data could

have been presented in an even more intuitive and clear way for users without any prior experience.

7.1.2 Evaluation

The results from the user tests were mostly positive and the comments given were mostly small things that were easily changed. Almost everyone found the dashboard easy to use and quickly understood the purpose of the dashboard, furthermore a great score from the SUS questionnaire was achieved. But it's important not to forget for whom this web application was made for. The tests were conducted for 8 people at Angry Creative, all with different but with a technical backgrounds or interest. All of the customers for Angry Creative does not come from a technical background which means that the results from the user test might not be fully accurate. Another thing that needs to be taken into account is how reliable the results from the user tests actually were. Both parties participated on the user tests were acquaintances and had have meetings with each other prior to the tests. Therefore the interviewees could have given bias answers to the questions and maybe not the harsher critics they had in mind.

The validity of the user tests is also important to discuss. Because this web application is developed for their customers, it would have been of preferable to have user tests with them. The user tests were planned in the beginning to be held with the customers of Angry Creative but because of the current pandemic situation caused by Covid-19, the plan had to be changed. It is therefore important to ask the question if the user tests are valid and if the result can be used. Usability and intuitiveness are two things that are not easily measured since they are perceived different from person to person. Even if there is a difference, a general line can be drawn for how usable and intuitive an application can be. The user tests were held on different people all with different backgrounds and education, but all of their answers were coherent and a top level SUS score was obtained. With this in mind, the obtained result would probably differ if the tests were held for Angry Creatives customers. However, we believed that with the extra information given on the dashboard about the components, anyone with any computer knowledge can understand and use the dashboard to get information about their project's progression at Angry Creative.

7.2 Method

Developing a dashboard with many different chart components and different ways of displaying the same data there were many choices to be made. In hindsight many things could have been done differently to receive a better result. A crucial thing that could have been done differently is the choice of visualization framework. The chosen Chart.js worked great with getting charts up and running fast, but the customizability was less then expected. When the dashboard got more complex the need for more custom settings for the chart components was needed.

Another thing that could have been done differently is that all the data parsing is done on the frontend and every time a component re-render. This is an inefficient solution and could cause problems the larger and the more data the project gets. A better solution would be to parse all the data on the backend and then fetch the results.

No user tests were conducted on the prototype section 3.3 that was developed. In hindsight this was seen as a good decision as it would have been hard to test interactions on a prototype without the functionality. The prototype was created to get feedback about the design choices of the dashboard from the supervisor and product owner of the dashboard.

User tests on web application was only conducted once and from that many insights, interaction and design propositions were obtained. Many of these propositions were later implemented, but never tested. Unfortunately there were not time to conduct a second user test so the new functionality could never be tested. A second user test would had made it

possible to see if the newly implemented functionality improved the dashboard in the way they were intended to do.

Another thing that might have given better result would have been if the user tests were held in person and not remote. Having the test remote was doable but it would have been preferable to do the tests physically at the office. This way it would be easier to communicate as well as better seeing what the test person was looking and interacting with.

7.2.1 Source criticism

The applied method used well known sources in the information visualization research field and published short articles in the central research fields of this thesis, (eg. Yi. et al [10] and Shneiderman [13]). A few none scientific resources were used in the user test field, but as these emanated from case studies they were seen as credible. When it came to the use of graphs and visualization techniques mostly dashboard related articles were used as they gave an overview of what was needed on a dashboard. No viable sources were found regarding how economic data should be visualized in this particular case. The scarcity of sources on this subject instead gave the possibility to test what was thought to be good ways of presenting the data and then let the user test's answers be the source of validity.

7.3 The work in a wider context

The web application that has been created for Angry Creative is a great foundation for their continued work to become more efficient and transparent towards their customers. As their customers start using the web application less time will be needed to update and inform them about the status of the project as they can see it for themselves. Intuitive and easy to use components also made it possible for user with no prior experience of dashboards and visualized data of this kind. The data was visualized in a multitude of ways and the same data was visualized in multiple different ways to give the user unique insights.

8 Conclusion

Angry Creative were satisfied with the project and the web application created. However, the application was more a proof of concept to see what was possible to create with their data. Many of the test users saw great potential in the product and it will most likely be something they continue developing on at Angry Creative.

This chapter will include the final conclusion of the project in form of answering the research questions that were stated in the beginning of the project and written in Section 1.4.

1. How can an end price be estimated for an agile project with regard to new events and time changes?

The end price of an agile project can be estimated with the help of the tickets that make up the estimated time for the project. All Angry Creative's projects are made up of tickets that have an estimated time they will take to complete. When a ticket is marked as done, the ticket can either become the exact estimated time, over- or underestimated. The difference between the estimated time and the closed time is the interesting factor as it is with the help of that difference an end cost can be estimated. First, the percentage of how much a ticket is over- or underestimated is calculated. Then calculating the average of all closed tickets on a certain week and the weeks prior to that week give an estimate of what the end cost can be.

In the budget graph (see Figure 6.5) these calculation are shown and it is also showing that the further the project progresses, the more accurate the estimation becomes.

2. How can interactions be implemented in an interface for showing agile project data?

Interactions should be implemented to support the data as good as possible. To be able to support the data, intuitive interactions with a coherent design language needs to be implemented. All data components should have interactions that makes it easier to understand the data and what it is presenting. These interactions are: hover and click functionality which are both important for understanding and helping the user to see details in the data.

3. How should economic data be visualized and with what techniques in order to make someone that is not familiar with the data able to understand the contents of it?

Economic data should be visualized in many different ways in order to give the user insights about the data (Section 2.3.2). The techniques that has shown to be best fitting to be used are line and bar charts as well as data tables. Line graphs are good for showing data over time, which works well with economy data as it spanned over a number of weeks and changes over time. Showing specific tickets requires other forms of visualization techniques and because of this a bar chart and table was used, where each bar/cell represented a ticket. A user that is not familiar with this kind of data needs graph components that are coherent and easy to understand in order to grasp and get insights from it. Support systems in the from of information boxes that explain the data help users understand what the different aspects of a visualization are intended to show.

4. How should an interface be designed so that a client can quick and easily get an overview of the progression of a project?

A customer can get a quick overview of the progression of a project with the help of a dashboard. A dashboard containing multiple components with visualized data and boxes that show general data give an overview of the progression of a project. Data that is shown in multiple ways make it possible for the user to locate the specific information as well a good overview of the general progression.

9 Future Work

This dashboard is a tool that will make Angry Creative's communication with customers much less time consuming and it will increase their transparency. The goal of this master thesis was to achieve these goals and they have in theory been achieved. However, the web application is not production ready for a number of reasons. These reasons will be discussed in this chapter.

Feedback from the user tests gave insights regarding the functionality and interaction design of the dashboard. From this feedback it was realized that the chosen chart framework was not enough for the interactions and customizations that was intended to be implemented. Therefore the future work that needs to be done is that the graphs needs to be redone in another data visualizing framework with more customization functionality. A framework that has this functionality is e.g. D3.js.

Another key feature that the test users would like to see in the production ready application was to be able to print reports from the dashboard. Today, Angry Creative sends out status reports to their customers at a regular basis. If this could be done by the customer through the dashboard this would free up time for the product owners as they would not need to create these reports for the customers.

User test were only held once during the development of the web application. Functionality and changes on the dashboard were made after the user tests and these would need to be tested in order to know if they met the feedback correctly and if they improved the dashboard in the desired way.

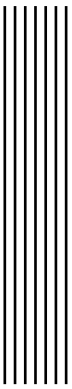
Another thing before the application should be released is that user tests should be held for the actual end user. The user tests as discussed in section 7.1.2 was held for Angry Creative's employees and not for their customers. In order to get more feedback and to see if the dashboard is production ready is to conduct a user test for them. This way, it can be seen if the dashboard's usability is the same as what was concluded in the user tests of this master thesis.

In order for the application to be production ready, estimations and predictions will also need to be more accurate. Making the predictions more exact can be done with the help of machine learning on data from older or other projects. The predictions need to be more accurate in a production environment as the percentage of the over estimation in the web application today is not reliable. It was not reliable in the sense that they projected the project's cost would be over 100 % than what was estimated in the beginning of the project. The cus-

tomers of Angry Creative have a budget for the projects they are purchasing and they will need to see projections that are closer to the real estimation, otherwise they will be worried about the outcome of the project.

A production ready dashboard would need a lot of support functionality that is out of the scope of this master thesis. Some of the features the dashboard would need is to be connected to the real backend system, as the one in use now uses an image of an old version of the real time managing system. A new additional backend system would need to be implemented which connects the customers to their correct projects and a support system that would make it possible for Angry Creative to monitor the user's traffic on the dashboard. This would allow for a wider range of use cases for the web application.

The system that was created during this master thesis is a good prototype to continue developing on. It has shown what is possible to visualize with the data and what the system the product owner initially had in mind could look like. The future work that could be done on this project are large enough to be their own thesis and hopefully Angry Creative will continue working on this project.



Bibliography

[1] M. Owais and R. Ramakishore. "Effort, duration and cost estimation in agile software development". In: *2016 Ninth International Conference on Contemporary Computing (IC3)*. 2016, pp. 1–5.

[2] P. Jain, L. Ahuja, and A. Sharma. "Current state of the research in agile quality development". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016, pp. 1177–1179.

[3] K. Beck M. Beedle A. van Bennekum A. Cockburn W. Cunningham M. Fowler J. Grenning J. Highsmith A. Hunt R. Jeffries J. Kern B. Marick R. C. Martin S. Mellor K. Schwaber J. Sutherland D. Thomas. "Manifesto for agile software development". In: URL: https://agilemanifesto.org/iso/en/manifesto.html.

[4] N. Nan and D. E. Harter. "Impact of Budget and Schedule Pressure on Software Development Cycle Time and Effort". In: *IEEE Transactions on Software Engineering* 35.5 (2009), pp. 624–637.

[5] Therese Macan. "Time Management: Test of a Process Model". In: *Journal of Applied Psychology* 79 (June 1994), pp. 381–391. DOI: 10.1037/0021-9010.79.3.381.

[6] S. Nakazawa and T. Tanaka. "Prototype of Kanban Tool and Preliminary Evaluation of Visualizing Method for Task Assignment". In: *2015 International Conference on Computer Application Technologies*. 2015, pp. 48–49.

[7] L. Pries-Heje and J. Pries-Heje. "Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India". In: *2011 Agile Conference*. 2011, pp. 20–28.

[8] Eliane Valiati, Carla Freitas, and Marcelo Pimenta. "Using multi-dimensional in-depth long-term case studies for information visualization evaluation". In: Jan. 2008, p. 9. DOI: 10.1145/1377966.1377978.

[9] Robert Spence. *Information Visualization: An Introduction*. 3rd. Springer Publishing Company, Incorporated, 2014, pp. 42–43. ISBN: 3319073400.

[10] J. S. Yi, Y. a. Kang, and J. Stasko. "Toward a Deeper Understanding of the Role of Interaction in Information Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007), pp. 1224–1231. ISSN: 2160-9306. DOI: 10.1109/TVCG.2007.70515.

[11] Robert L. Harris. *Information graphics : a comprehensive illustrated reference*. Oxford University Press, 1999, pp. 26–31. ISBN: 0195135326.

[12] George Dinwiddie. *Feel the Burn: Getting the Most Out of Burn Charts*. Sticky Minds, Better Software, 2009, pp. 26–31. ISBN: 9783030142735. URL: http://idiacomputing.com/pub/BetterSoftware-BurnCharts.pdf.

[13] B. Shneiderman. "The eyes have it: a task by data type taxonomy for information visualizations". In: *Proceedings 1996 IEEE Symposium on Visual Languages*. Sept. 1996, pp. 336–343. DOI: 10.1109/VL.1996.545307.

[14] Ogan M. Yigitbasioglu and Oana Velcu. "A review of dashboards in performance management: Implications for design and research". In: *International Journal of Accounting Information Systems* 13.1 (2012), pp. 41–59. ISSN: 1467-0895. DOI: https://doi.org/10.1016/j.accinf.2011.08.002. URL: http://www.sciencedirect.com/science/article/pii/S1467089511000443.

[15] A. A. Rahman, Y. B. Adamu, and P. Harun. "Review on dashboard application from managerial perspective". In: *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*. July 2017, pp. 1–5. DOI: 10.1109/ICRIIS.2017.8002461.

[16] W. Noonpakdee, T. Khunkornsiri, A. Phothichai, and K. Danaisawat. "A framework for analyzing and developing dashboard templates for small and medium enterprises". In: *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*. Apr. 2018, pp. 479–483. DOI: 10.1109/IEA.2018.8387148.

[17] Clayton Lewis and John Rieman. "Task-centered user interface design". In: (1993). DOI: http://hcibib.org/tcuid/tcuid.pdf.

[18] URL: https://material.io/. (accessed 2020-04-07).

[19] Mijo Balic Ingrid Ottersten. *Effektstyrning av IT - Nyttan uppstår i användningen*. 2nd. Liber AB, 2004, pp. 42–43. ISBN: 9789147074501.

[20] F. B. Zainuddin and S. Liu. "An Approach to Low-fidelity Prototyping Based on SOFL Informal Specification". In: *2012 19th Asia-Pacific Software Engineering Conference*. Vol. 1. 2012, pp. 654–663.

[21] William Albert and Thomas Tullis. *Measuring the User Experience : Collecting, Analyzing, and Presenting Usability Metrics*. 2nd. Elsevier Science Technology, 2013. ISBN: 9780124157811.

[22] Jakob Nielsen. *How Many Test Users in a Usability Study?* 2012. URL: https://www.nngroup.com/articles/how-many-test-users/ (visited on 03/18/2020).

[23] Kate Moran. *Quantitative User-Research Methodologies: An Overview*. 2018. URL: https://www.nngroup.com/articles/quantitative-user-research-methods/ (visited on 03/20/2020).

[24] S. B. Linek and K. Tochtermann. "Assessment of usability benchmarks: Combining standardized scales with specific questions". In: *2011 14th International Conference on Interactive Collaborative Learning*. 2011, pp. 67–75.

[25] John Brooke. *SUS - A quick and dirty usability scale*. URL: http://cui.unige.ch/isi/icle-wiki/_media/ipm:test-suschapt.pdf. (accessed 2020-03-20).

[26] Jeff Sauro. *Measuring Usability With The System Usability Scale (SUS)*. URL: https://www.userfocus.co.uk/articles/measuring-usability-with-the-SUS.html. (accessed 2020-03-19).

[27] Sandra G. Hart and Lowell E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". In: *Human Mental Workload*. Ed. by Peter A. Hancock and Najmedin Meshkati. Vol. 52. Advances in Psychology. North-Holland, 1988, pp. 139–183. DOI: https://doi.org/10.1016/S0166-4115(08)62386-9. URL: http://www.sciencedirect.com/science/article/pii/S0166411508623869.

[28] Luca Longo and Maria Chiara Leva. *Human mental workload : models and applications : second International Symposium, H-WORKLOAD 2018, Amsterdam, The Netherlands, September 20-21, 2018, Revised selected papers.* Communications in computer and information science: 1012. Springer, 2019, pp. 52–53. ISBN: 9783030142735. URL: https://link.springer.com/book/10.1007%5C%2F978-3-030-14273-5.

[29] URL: https://reactjs.org/. (accessed 2020-03-18).

[30] URL: https://angular.io/. (accessed 2020-03-18).

[31] URL: https://vuejs.org/. (accessed 2020-03-18).

[32] URL: https://redux.js.org/. (accessed 2020-03-18).

A User test tasks

The following tasks were the ones that were asked in the user tests. The tasks were asked by the interviewer to perform whilst their thought process were noted. Sometimes the questions were not stated exactly how they are written down below to get more thoughts and feedback from interviewee.

A.1 General tasks and questions about the dashboard

- Hur bär du dig åt för att ta reda på den estimerade kostnaden för projektet? Och, vad är den estimerade kostnaden för projektet?
- Vad kan du läsa ut från projektets tidsloggningar-grafen, och hur går tankarna för att komma fram till detta resultat?
- Vad kan du läsa ut från loggade ärendetider-grafen och hur går tankarna för att komma fram till detta resultat?
- Vad kan du läsa ut från projektion av slutkostnads-grafen och hur går tankarna för att komma fram till detta resultat?

A.2 Tasks about interactions with the tickets

- Klicka på en godtycklig ticket för att se vad som händer
- Hur mycket tid har loggats på denna ticketen?
- Under vilka "sprintar" har det loggats tid på denna ticketen?
- Går det att få upp ticket modalen på fler sätt?

A.3 Tasks about interactions with the Progression graph

- Hur bär du dig åt för att ta reda på hur mkt tid det loggas på en viss vecka?
- Nu vill vi att du tar bort alla alla linjer förutom "Högsta Estimat"-linjen. Om projektet hade fortskridit enligt denna linjen, till vilken sprint hade projektet varit klart då?

A.4 Tasks about interactions with the Ticket graph

- Vad betyder de olika färgerna på barsen?
- Vad kan du utläsa från ticket EQ-143?
- Kan du sortera grafen på något sätt?

A.5 Tasks about interactions with the Ticket table

- Hitta EQ-253 och säg hur mycket tid som loggats?
- Vad betyder "+" jämte progressbaren?
- Sortera tabbelen efter högst loggade ticket först. Vilken ticket är detta och hur ser den ut?
- Sök efter EQ-253. Vad ser du?

A.6 Tasks about interactions with the Budget graph

- Hur stort är budgetintervallet vecka 26?
- Berätta vad du tror de olika linjerna står för?
- Vad har hänt vecka 45? Är projektet underestimerat?
- Berätta om den generella utvecklingen av kurvan.

A.7 Tasks about interactions with the Filtering calender

- Välj två veckor. Hur ändras informationen?
- Vad händer med informationslådorna?
- Välj vecka 1-9. Vad händer med dashboarden? Varför ser det ut som det gör?
- Filtrera mellan vecka 1-37.
- Filtrera mellan vecka 25-28.
- Vad hände med bar charten nu? Vad kan du utläsa?
- Återställ filtreringen till normala värdena

A.8 Mixed tasks about interactions with the entire dashboard

- Om allt hade gått enligt den medelmåttet veckan, när hade projektet blivit klart då?
- Hur mycket tid har loggats på ärende EQ-240? Vad för mer information kan du se om EQ-240?
- Hur kan du på ett enkelt sätt se vilka tickets som är överestimerade? Underestimerade?
- Vilka delar av dashboarden kan du interagera med?
- Vilka veckor loggades det mindre tider?
- Även om detta är ett verktyg som främst utvecklats för kund, ser du som PO något värde i den? Hur skulle det kunna vidareutvecklas för att du skulle få ut något relevant av den?

B Sus and NASA TLX

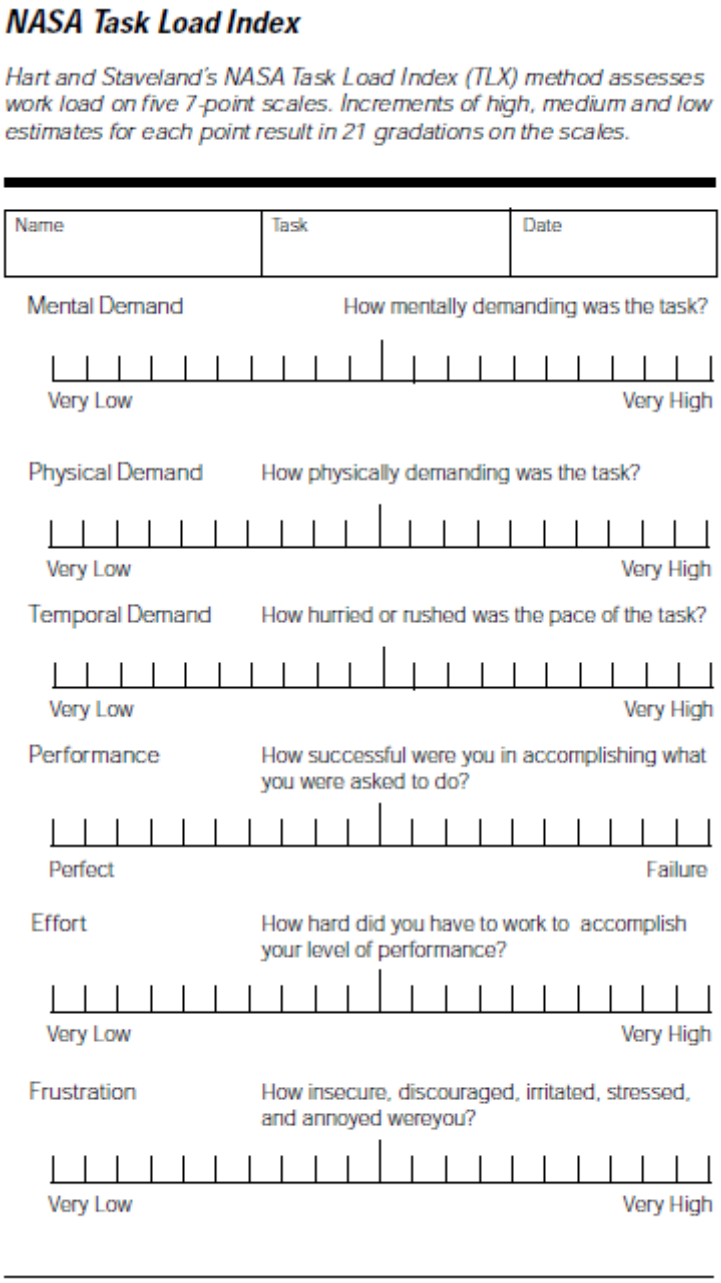
The following appendix includes the questionnaires for both SUS [25] and NASA TLX [28] that were used or referenced during this master thesis.

B.1 SUS - System usability scale

Table B.1: The original system usability test questionnaire.

	1 (Strongly disagree)	2	3	4	*5 (Strongly Agree)*
1. I think that I would like to use this system frequently.					
2. I found the system unnecessarily complex.					
3. I thought the sustem was easy to use.					
4. I think that i would need the support of a technical person to be able to use this system.					
5. I found the various functions in this system were well integrated.					
6. I thought there was too much inconsistency in this system.					
7. I would imagine that most people would learn to use this system very quickly.					
8. I found the system very cumbersome to use.					
9. I felt very confident using the system.					
10. I needed to learn a lot of things before I could get going with this system.					

B.2 NASA TLX



NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?

Very Low — Very High

Figure B.1: The original questionnaire for NASA TLX.