Project Approach Tool

Bachelor graduation assignment by Matyas Kone at Saxion University of Applied Sciences for developing a plan of an approach web tool.

Student: Matyas Kone (416062)
Company: Saxion University of Applied Science HBO-IT
Company supervisor: Danny Plass

School: Saxion University of Applied Science
Graduation supervisor: Etto Salomons
Study: Software Engineering

ABSTRACT

Matyas Kone, Project Approach Tool
(Under the supervision of Danny Plass)

The purpose of this document is to describe Matyas Kone graduation assignment, which is the last and final part of the HBO IT, Software Engineering study at Saxion University of Applied Sciences. The graduation assignment has been carried out at Saxion University HBO-IT lectorate in Enschede.

This research conducts the development of an online project management tool for the Saxion University of Applied Sciences to serve students and teachers better, innovatively and creatively, which, as a student, has always been my purpose. In this document, the reader shall find the detailed development process of the application. The primary programming language used for this development is JavaScript, and additional open-source frameworks as ‘React’, ‘Vue.js’ and ‘Angular’ have been taken into consideration. Based on the exploration, the ‘Angular’ library was most suitable for this project. The developed web tool can professionally manage any complex project and allows the users to personalise the tool for the project and their own needs. Users can choose methods they care to engage with, from pre-defined templates, and all the well-known research methods have been implemented, as well as pre-defined cards or any method that is used at the institution either by students or educators. The backend API stores the data and maintains communication with the webserver, to make the process smoother and more reliable. Throughout the development, the researcher has deepened his knowledge in web development and carried out a fully working application that will be used at the Saxion University of Applied Sciences.

Preface

In 2016 I entered the Software Engineering course at the Saxion University of Applied Sciences. I always had a passion for technology and programming. First, I started to develop my interest in mobile application and engage with Android development, which led me to my internship. In addition, I have realised that a programmer's job is to discover and experiment in many different fields, and in order to be successful, one shall have many different skills, professionally and personally equally. During my last years, I focused on Python and tracking systems and my graduation assignment creates an online web application for an institution that I highly value. I believe my years as a student paid off, and now I consider myself a great software developer with a high passion for the industry.

Before diving into this document, I would like to thank my company coach, Danny Plass for her support, feedback and involvement during this and previous projects as well. I would like to also thank the project study coach Etto Solomons for his support, and feedback during this project and remanding me for each important deadline well in advance. Furthermore, I would like to thank everyone else who was involved in the project as well as in my studies.

Table of Contents

1. Terminology

Term	Meaning
Card	A card represents the physical card in the application, but also can be a note, or question which can be assigned to the groups.
Group	A group is the main component of the application; the user can drag a group into the workplace and can assign cards into it.
MVP	Minimum viable product
MCA	multi-criteria analysis for researching
API/ backend	Application programming interface
Front end	Part of the application which the user interacts with
DOT framework	Development Oriented Triangulation
PAT	Project Approach Tool
MoSCoW	Must have, Should have, Could have, and Won't have
ORM	Object-relational mapping
NPM	Node package manager

VSode	Visual Studio Code
DOT	Development Oriented Triangulation
ICT	Information and communications technology
IDE	Integrated development environment

2. Introduction

During this graduation assignment, an online project approach tool has been created for the Saxion Unversity of Applied Sciences. As teachers and students have been struggling with finding the right tool for years, the development was necessary. The online platform's features serve the users better than any platform or tool for this matter, and the application has been built in a user-friendly way, considering all the necessary approaches, the client's needs, the pros and cons of other platforms, used research methods and evaluation methods in different professional areas. Different programming languages and libraries have been taken into account as well as other researches for similar tools. The application strives to shorten project management time without creating an overly complicated system that is smart enough to serve more technical minded people as well as creatives who like to have a simple, clean and trendy design. The application is modular enough to expand its full potential and potentially develop it further for a different purpose. With enough care and development, it can be a revolutionary tool in project management, but for now, the paper only shows the current steps of the development and leads the reader through the whole process, addresses questions and strives to answer them with supported research and visuals from the fully functioning prototype.
The main challenges and questions which will be answered during this project:

- Understand the framework itself
- Which framework suits best for this application?
- How to design an intuitive and easy to use application?
- How to create the application using the requirements and designs?
- How to store the boards offline?
- How to store the boards in "cloud"?

This document will describe the realization of the assignment from planning to actual product.
These are the goals of this document:

- Describe how the product developed from design to implementation
- Describe what choices were made
- Describe how the stakeholders and end-users were involved in the development process
- Describe technical workings behind the product

3. Project Background

When a student or student group receives a new project, the first step is always the planning; of course, this is true for any project in general. Creating a plan is a crucial step for every project since the plan can highly influence the project result. During this project, the development will be focused on the students within school projects.

At the HBO-IT department of Saxion, the teachers and students are using the Development Oriented Triangulation (DOT) framework, with multiple decks of cards, when starting a new project. The DOT framework main goal is to help project members make well-founded decisions on **what and how to research**. A general overview of the main framework components can be seen in Figure 3.1

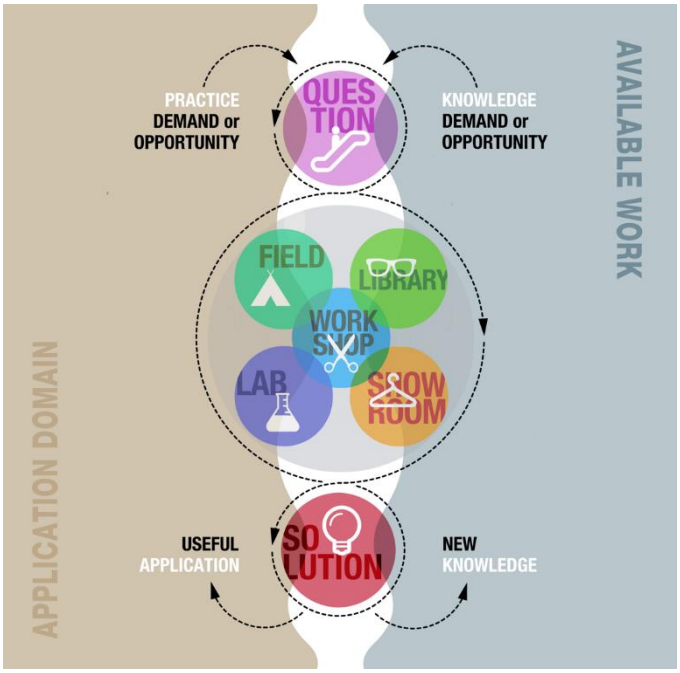


Figure 3.1 Overview of the DOT framework

The framework arranges the project activities and methods into five research strategies. These strategies help the users on **how** to do the research.

- Field
- Library
- Workshop
- Lab
- Showroom

If the methods cover all five strategies and are balanced according to the project needs, the research will more likely succeed. By using the DOT framework, the project members can get a great overview of the resources are used efficiently.
At Saxion, two decks of cards are used alongside the DOT framework, depending on the needs of the project:

- Steppingstone: This deck of cards is developed by Miriam Loose (one of the stakeholders) allows the users to think from the practical process and deliverables point of view
- The ICT research method cards: These cards provide additional ICT inspired methods and a way to think about what to use when. The cards organized by the five research strategy of the DOT framework.

The used decks are depending on the type of the project as well as on the field of the project. This way of approach project can be used in a different type of projects, and also for different parts of the projects.

Currently, when a new project starts and the project members choose to use this approach tool, they will receive multiple decks of cards. The decks are based on their experience and on project nature. Less experienced users usually receive an introduction from a teacher about how to use the cards together with the DOT framework. After receiving the cards, they

have to familiarize themselves with the cards, the number of available cards depends on how many decks they received. Once they familiarize themselves with the basics of the framework and cards, they can start creating a plan. During the planning, they can group the cards into phases, and the phases can represent a stepping stone, a deliverable, a research question. These groups then can be connected to each other. This part of the planning is very versatile. There are no exact rules on how to do the planning; therefore, the outcome is based on the project. However, there are few templates which can be used in order to help with the process.

This process works well but has a few drawbacks. One of the main drawbacks is that it is tough to reuse the created plan at the next meeting or in a different location. If they want to reuse the plan, they have to take a picture and re-do it based on the picture. For beginner users, it can take up a lot of time to familiarize and search through the cards and find what they are looking for. The card has limited space to show information.

Since at the moment, this is only available as a physical tool, the teachers thought that it would be useful to create a **web application** for this purpose. The main goal of the web application is **not to replace the physical** way but provide the same base functionality with **useful additional features and** make it appealing to use for wither audience who prefers the digital way of doing things.

3.1 Stakeholders

Matyas Kone
I'm the developer, how is developing the desired system. I'm devoted to creating a product which will be accepted by all the stakeholders and will be used after the development.

Saxion (HBO-ICT)
Saxion as a company is interested in the project, and they expect a product which the teachers can suggest to students to use during projects.

Danny Plass
Danny is the product owner of this project as well as the company coach. She is the representative of the national HBO-i Open OiO project to develop open educational resources for research in HBO-ICT education. She uses the DOT framework and card decks on a daily basis.

Etto Salomons
The project study coach from Saxion and representative of HBO-ICT 3s. But also a potential end-user of the Project Approach Tool.

Miriam Losse
The developer of the steppingstone card set, as well as a user for the end product and representative of Saxion Research in Education.

Koen van Turnhout
(co)developer and (co)author of the DOT framework and ICT research methods set and representative of HBO-i Research in Education.

Kasper Kamperman
CMGT representative and possible end-user for the final application.

4. The project processes

In order to deliver a product which is accepted by the stakeholders and has added value to the end-users, a structured process needed to be designed. These processes are organized into phases. In this chapter, I describe these phases and processes in detail.

4.1 Phases

The following diagram (Figure 4.1.1) shows the created phases and the result of each phase.

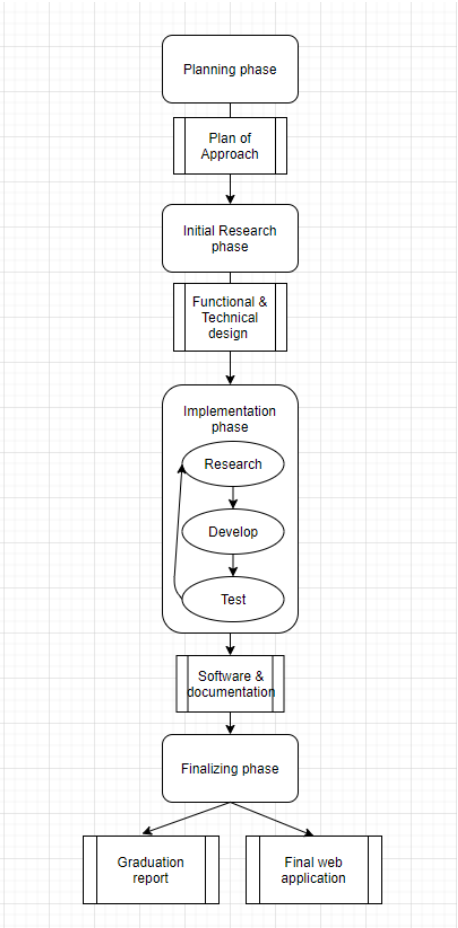


Figure 4.1.1 Phases flow chart

4.2 Planning phase

The first step of the graduation assignment was to create a plan on how to tackle the primary goal of this project. In my previous projects at Saxion, I never used the DOT framework or any of these card sets to plan a project. Therefore, the planning was my first introduction of what is the goal of the project.

Figure 3.1 Result of the PAT project planning

The planning was done together with Danny Plass since she is familiar with approaching a project with the DOT framework and with the cards. She is also the product owner in this project. In this process, we defined sub deliverables and assigned steppingstones and cards from the ICT deck. The steppingstones are defining what needs to be done in order to finish those sub deliverables, and ICT cards are defining the required activities.

4.2.1 Gantt Chart

In order to have a great visualization of the project planning and keep everyone who is involved on the same page, a Gantt chart was created. This chart shows the main project activities, the planned time for each task, and major deadlines.

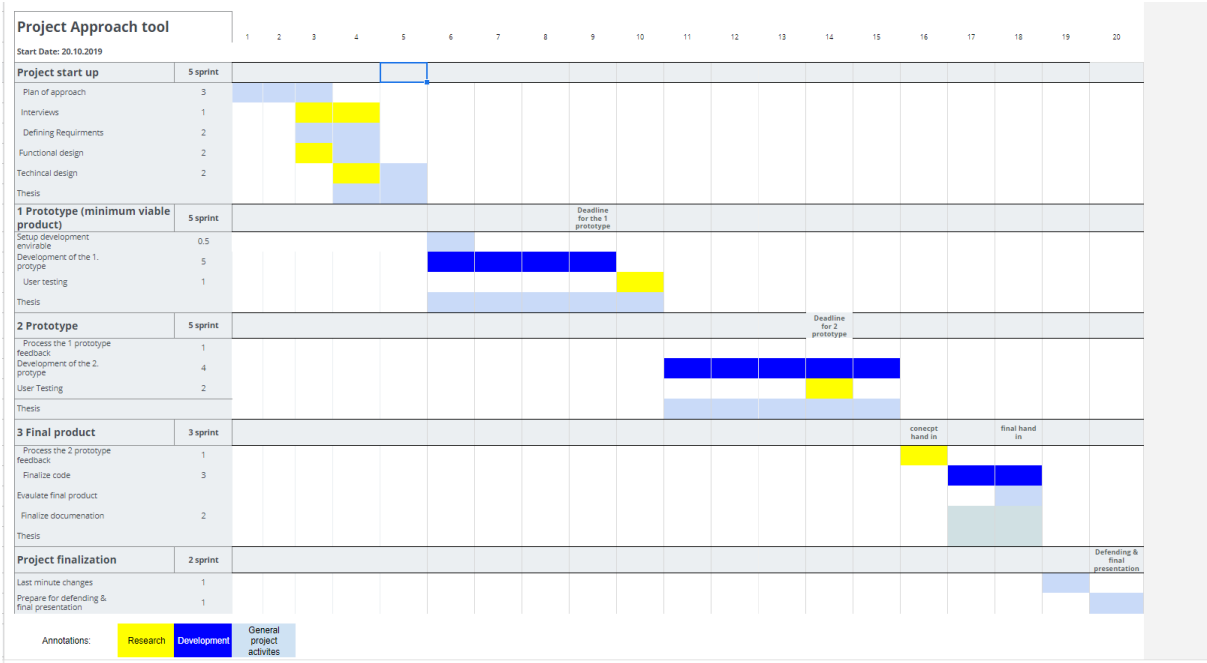


Figure 3.2 Gantt Chart

The project starts with a start-up period, which includes the planning, collecting the requirements, and based on that creating the functional and technical designs.
During the development period, two prototypes were created. These prototypes were handed out for the stakeholders and for student groups to test the application and provide feedback on it. The received feedback was used to create the next iteration of the software.

The result of the **planning phase** is the **plan of approach**.

4.3 Initial research phase

After finishing with the planning and a plan of approach, the next step was to create functional and technical design documents. The aim of the research phase was to have a **good understanding** of the final product and **make technical** choices for the realization.
In this phase, the following points needed to be clear:

- The main goal of the application
- Requirements of the application
- Define user stories based on the initial requirements
- Create mock-ups based on the requirements
- Choose a suitable technology stack for the application based on requirements and mock-ups

4.4 Implementation phase

The implementation phase was the most crucial phase during this project since in these phases, the final product was realized.

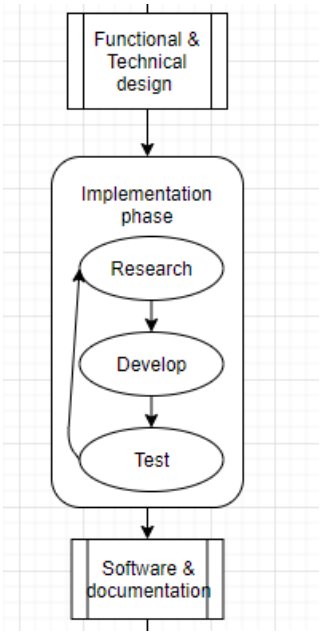


Figure 4.4.1 Implementation phase flow chart

In order to start with this phase, the functional and technical design documents needed to be finished and accepted by the stakeholders. Since the implementation phase is the most complex part of the project, it was divided into smaller phases: research, development, test as it can be seen in Figure 4.4.1. These phases were repeated during this implementation period. During the development phase, SCRUM was used, with one-week sprints. Therefore one development cycle was one week. This choice was necessary because, at the end of every sprint, there was a progress meeting with the product owner and based on the previous week results, **the requirements and their prioritization were adjusted.** This approach was vital for the success of the project since in the beginning none of the stakeholders had a clear picture of how the final application should look like and function.

4.4.1 Research

In the initial research phase, the fundamentals of the application were laid out. However, during the development, a lot of **small choices needed to be made**. In some sprint, the research phase was just a few google searches, but in others, some experimenting was required as well.

4.4.2 Development

During the development phase, new user stories were picked from the Scrum board and developed. These user stories could be already existing parts of the application which needed to be adjusted or re-written based on the product owner feedback, or entirely new parts. The development phase also included the code documentation, which can be comments, tutorials, or documentation based on the nature developed parts.

4.4.3 Test & review

After each user story was done from the backlog, it moved into the test phase, where the new functionality was tested, and the whole application as well, in order to make sure that the new part of the application does not break the already existing one. At the end of each sprint, there was a sprint retrospective with the product owner, where all the new functionality was shown, and based on the result, the next sprint was planned.

4.5 Finalizing phase

The finalization phase will be entered two weeks before the final deadline for this project. At this point, all the requirements regarding the Minimum Viable Product must be done, and most of the user stories should be in the done list. This, however, does not necessarily mean that the product is final. This goal of this last phase is to review and test the product with the stakeholders and to think about conclusions, future changes and recommendations. If, some significant issues are coming up during the final review, in this phase is the last opportunity to fix those. Finally, there will be time to wrap up and finish all documentation, coding and prepare the final graduation presentation.

4.6 Programming Strategy

In order to make the most out of this project, the **Scrum framework** was chosen as development management. The reason why Scrum fits better than Kanban is that Kanban is designed for a continuous development cycle without specific deadlines; this would not fit the planning since this project has **two deadlines** for each **prototype**. In theory, Kanban could be more efficient in projects where the requirements can change, but in this case, the requirements can only change between sprints, during the weekly meeting, therefore we also chose to make one sprint one week. This one-week sprint period allowed us to respond quickly if there was a misunderstanding between the developer and product owner.
On the other hand, Scrum is designed for teams; therefore, I applied some changes in order to fit this project better.
The planning poker was to be done by myself. However, since I have to do all the work, it is not as crucial as in team projects but provided excellent feedback for the product owner on what she can expect from the upcoming sprint. The code and product reviews will be done together with the product.
For the implementation phase, choosing a good IDE/editor was an important decision. **Visual Studio Code** (Visual Studio Code) has been chosen because it is intuitive and has plugin great plugin support for Angular and NodeJS as well. These plugins help to speed up the development processes with IntelliSense (Autocomplete, and framework-specific error checking), boilerplate code generation, and formatting according to the framework standards.
To manage Scrum and the user stories, Trello (Trello) was used.

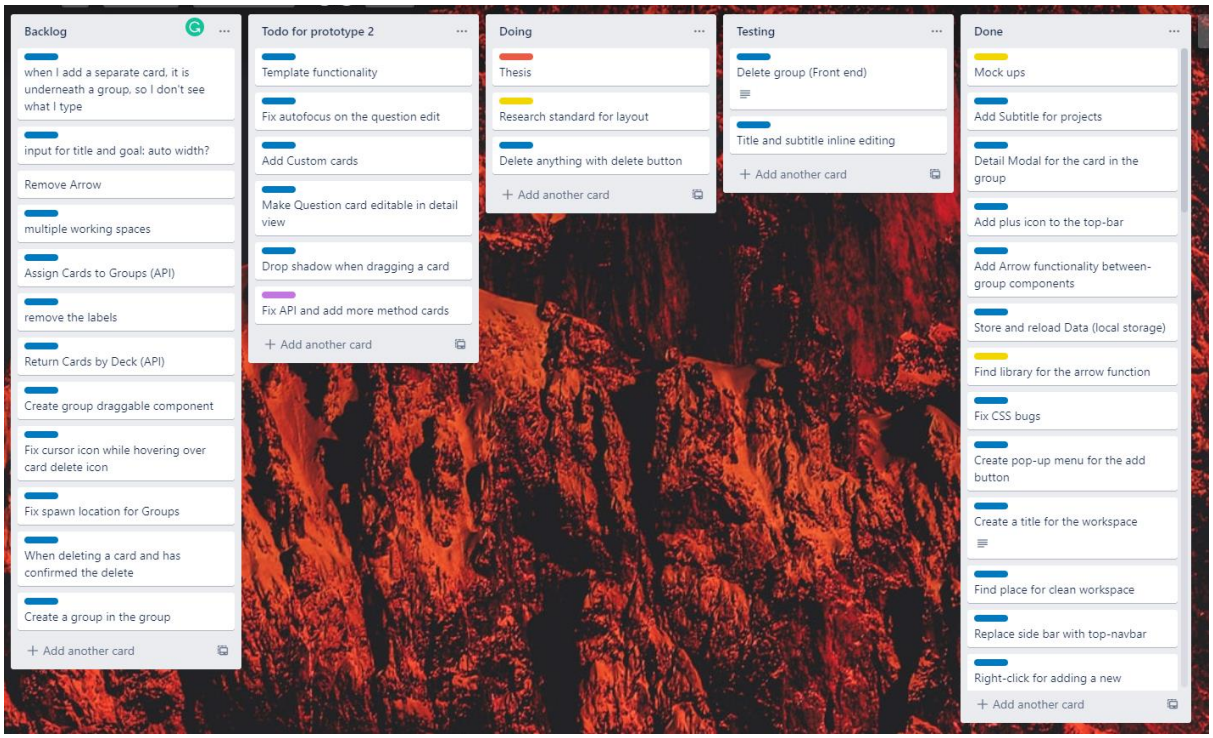


Figure 4.6.1 Trello board snapshot

- The user stories were prioritized with the **MoSCoW** method by the product owner
- In the Trello, board prioritization is represented by the order of the user stories top to bottom, from Must to Could
- Usually, for one project a new Trello board is created for each sprint or prototype but, since in the project I worked alone I decided to create one general board for the whole project, with two backlogs
- The following lists were created for the Trello board:
    - **Backlog**: All the Must, Should, and Could story, Won't have user stories were not included in this backlog.
    - **TODO**: This list consists of the user stories which were planned for the upcoming sprint or prototype
    - **DOING**: Represent the user stories which are worked on at the moment, the thesis was there the whole duration of the project
    - **TESTING**: Represent the user stories which are already done, from the user story could be moved to Done or back to TODO list if the tests failed or were not accepted.
    - **DONE**: The user stories can be moved here, once they meet with Definition is Done
- **Definition of done is for user story when the code is written, refactored, commented, documented and tested and accepted by the product owner.**

- During the project, there will be twelve sprints, where each sprint is one week.
- At the end of each sprint, there will be a Sprint Retrospective with the product owner in order to review the last week process and plan the upcoming one.
- At each prototype, there will be a meeting with all the stakeholders where the application is shown, and feedback is collected for future development.
- Every fourth or fifth week, there will be a meeting with the school supervisor in order to discuss the progress of the project.

4.6.1 Weekly progress meeting

Weekly progress had multiple purposes in this project. The meeting started with a quick overview of the previous sprint and proceeded into a sprint review. At each meeting, the new functionalities or fixes were shown to the product owner, as well as the entire application in order to prove that the updates do not break the already existing ones. The interesting solution and codes were explained. After a quick sprint retrospective is held. At this point, the focus moves into the upcoming sprint. Based on the previous sprint result, we planned the next one. The user stories and requirements were added or removed from the backlog. And finally, the user stories were prioritized by using the MoSCoW method

5. Product - Design

In this chapter, the aim is to describe and explain how the project involved from an **idea to a design** which can be used to develop a professional product for the client.

The main goal of the project: *Create a project approach tool in the form of a web application, which incorporates the DOT framework, with multiple decks of cards, and can be used in multidisciplinary projects.*

5.1 Current situation

The first step that needed to be done was to get a good overview and understanding of the current situation.

During ITC projects, there are always some challenges which need to be solved. In order to solve these challenges, the right questions need to be formulated and answered. These questions are usually more complex than a Google search. They are small research. These researchers are required to create a valuable and usable product at the end of the project. There are a few frameworks around there, but in Saxion and other Universities, they mostly use the DOT framework, which was mainly developed for ITC related projects. The framework helps the project members to structure and organize the research activities in order to proceed from a problem to a solution.

Currently, when a new project starts and the project members choose to use this approach tool, they will receive a deck of ‘real’ cards. The cards can be multiple decks for multiple purposes. Once they have the cards, they must create (Figure 5.1.1) the main questions or main steps and organize them in a way that makes sense for the project.

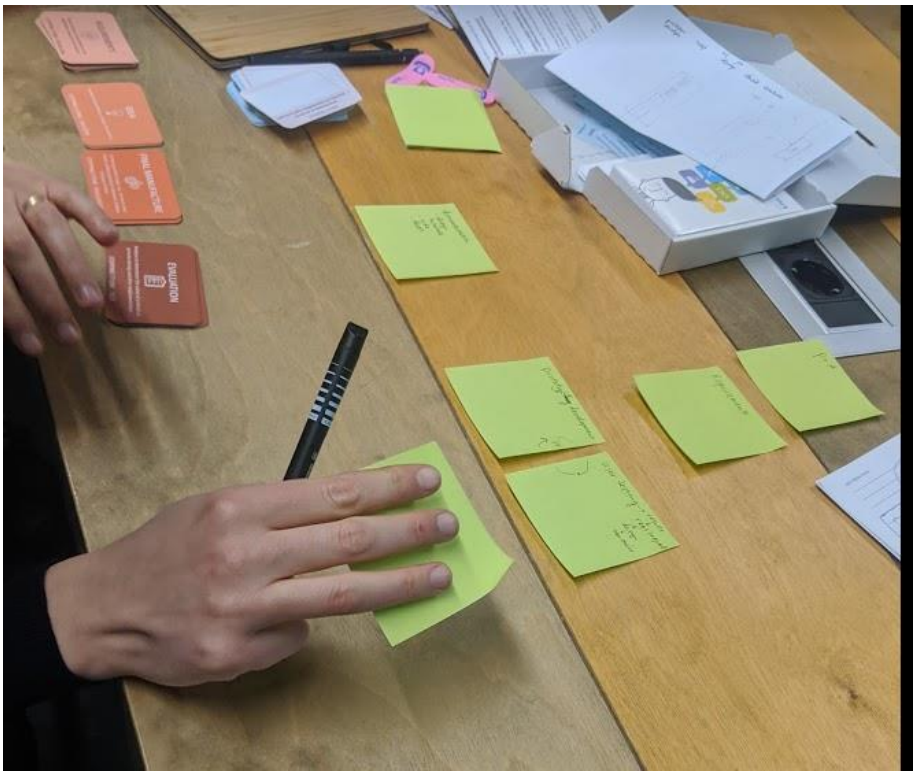


Figure 5.1.1

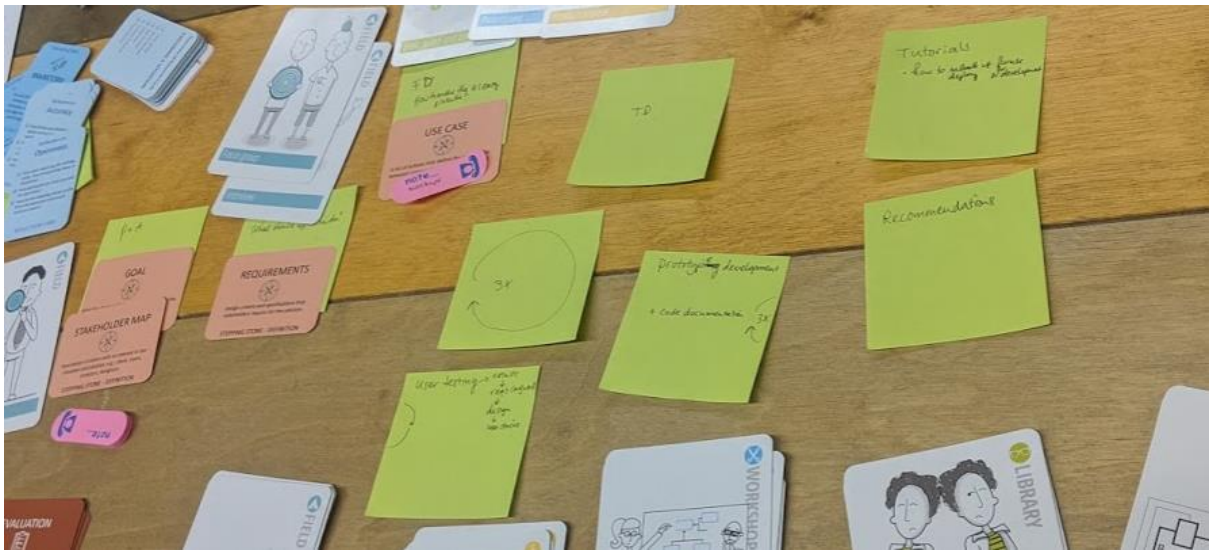


Figure 5.1.2

As you can see in Figure 5.1.2, the steps and questions are organized in a meaningful way. In the middle, you can see that those steps will be repeated three times. When the layout of the project is finished, the team members can fill them up with the cards.

Figure 5.1.3

In Figure 5.1.3, I already started to assign cards to the steps. In this case, I am using the ICT research, and steppingstones deck with some custom cards as well. To find the cards which you want to use can be problematic because you have to search through the whole decks, or you have to separate them in the beginning into groups, which is time-consuming, but on the other hand, it provides a better overview for the less experienced users. Currently, a card can only be used once in one layout. When the meeting is done the project, members have to take a picture, and the next time, if they want to use it again, they have to use the picture to set it up. This is a considerable limitation of the framework which can be solved.

5.2 The interviews

The interviews helped to understand what the stakeholders expect from this project and helped with formulating the requirements.

Danny Plass, Product owner
Since Danny is the product owner and company coach, we had multiple meetings about what the product should be. She explained to me, what is the main problem which needs to

be solved (described in the current situation section above). Since I never approached the project with this way nor used a Dot framework, my first challenge was to understand how they use it. In order to help this process, in one of our meetings, we used this framework to plan my project together. The result of this planning can be seen in the pictures of the previous segment. This experimenting helped me to understand how the project approach "tool" is working. Besides that, we also discussed the general requirements and use cases of the application. Between interviews, I drew some mock-ups of the user interface, and we reviewed them together.

Miriam Losse, stakeholder

The first interview with stakeholder was with Miriam. She is the author of the steppingstones, which will be incorporated into the software. During the interview, Danny was present, and Miriam joined via the internet. Miriam explained to me how she approaches a project currently, how she uses the DOT framework and cards. The general idea was the same as Danny described me, but Miriam has more experience with research projects, and Danny has more experience with IT product-related projects. Miriam uses her steppingstones to make sure that during research, the project members are aware that they also must deliver and end product. The steppingstones help them identify the sub delivered and steps which required to succeed with the project. Then I asked Miriam what extra added value she would expect from the tool compared to the current use of the framework. She pointed out that the product **should not replace** the current physical way of using it, rather it should be an **addition to it**. Therefore, one of her primary requirements was that the system could **provide more** useful **information** about each card. When choosing a card, the system should **ask questions** from the user, in order to make sure that the **right card** is used. She also would like to see an **archiving functionality**, with the ability to see a history of how the planning was changed. She also mentioned that the application would be used in **multidisciplinary projects**. Therefore the application **should be easy to use** and understandable for a different type of professionals, for example: in one project, a doctor and software engineer could work together. She also mentioned that it would be nice if there will reporting **function,** so the users can include that in their reports.
In general, the interview was useful as I could understand the project from a different point of view.

5.3 Desired situation

The stakeholder's desire an online software. Which can provide a solution for the use case described in the current situation chapter and resolve the current system weak points. And provide some additional functionality which is not possible with the physical cards. The desired situation is based on the interviews with the stakeholders and product owner; however, the final requirements will be decided by the product owner.

5.4 Requirements

Based on the stakeholder's expectation and interviews, a general requirement list was created. The list was presented to the product owner to prioritize the requirements. The prioritization was done by using the MoSCoW method. The **must-have methods** are the requirements for the minimum viable product, which was the **goal for the first prototype.** During the project, the requirements kept changing as the project involved. These new requirements were created based on feedback from the weekly progress meetings and based on the stakeholder's feedback. (The rest of the requirements can be found in Functional design document TODO or appendix)

Functional requirements

Must have requirements(For the Minimum Viable Product)

#	Requirement	Notes
1	As a user, I want to be able to create groups so I can assign cards to it.	
2	As a user, I want to able to drag and drop the groups, so I can organize them in order to fit the project	
3	As a user, I want to be able to edit and delete groups.	
4	As a user, I want to able to assign title into the groups, so I can quickly identify them.	
5	As a user, I want to able to assign cards into the groups.	
6	As a user, I want to be able to click on the card so I can reveal more information about them.	This applies to the card which is already assigned to a group, so it is visible on the workspace
7	As a user, I want to able to move cards between groups, so I can organize them.	
8	As a user, I want to able to remove cards from the groups.	
9	As a user, I want to be able to browse the cards, so I can choose or view more information about them	

10	As a user, I want to be able to create a custom card so I can personalize the tool for the project	
11	As a user, I want to be able to choose a deck of cards, so I can look through them more comfortable.	

5.3 Global Design

In order to satisfy the requirements, a front-end is required. However, the **decision** was made to **create a** supporting **backend** as well in the form of an API. By having an API, the data can be served from the main point instead of having to hard code it into the front end application. This saves time in the development process, and new functionality can be easier to add on longer term. However, the main focus kept on the front-end application since this is the most important part of the whole project. Without a properly working frontend, the application can provide the required functionality for the end-users.

5.3.1 Front-end

After the requirements were agreed on the next step was to visualize the application Based on interviews and requirements, I created mock-ups. This chapter describes the process of how the design of the application involved with every iteration of the product.

It is essential to define and understand the **two main components** of the application. These components will be referred to throughout the design and realization phase:

Group (Phase):

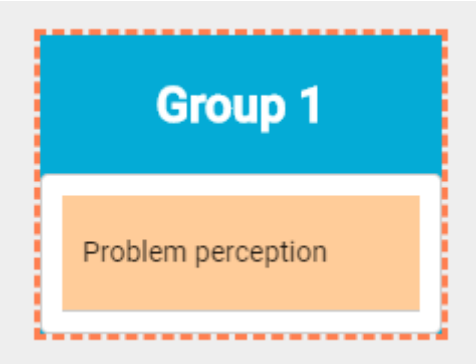


Figure 5.3.1.1 The group component

A group represent a list, with a header. The header contains the title of the group. The group can be dragged around in the workspace

Card (Question):

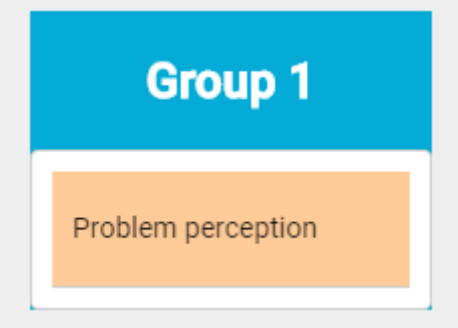


Figure 5.3.1.2 A card component

A card component represents the box, with the title of the box, this component can be dragged between groups. A question is also card, but the content of it defined by the user.

5.3.1.1 Initial Mockups

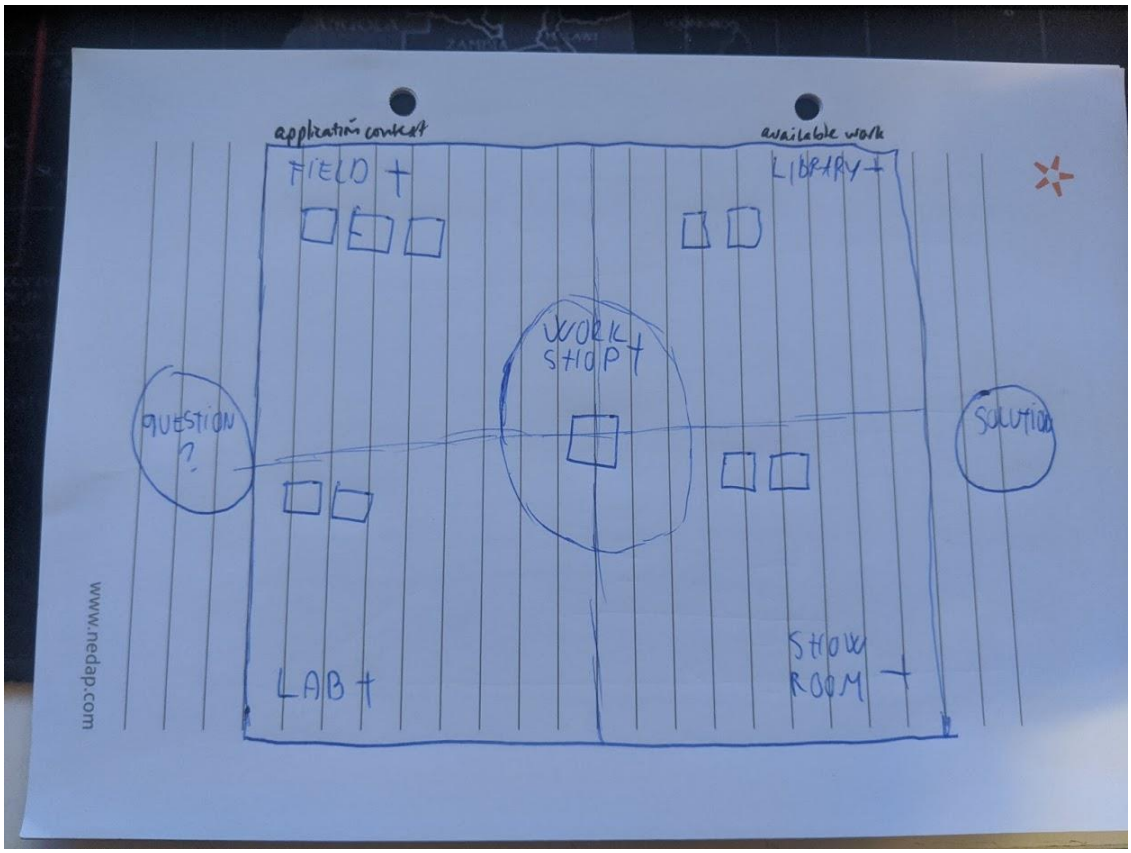


Figure 5.3.1.1 The first mock-up

As it can be seen in Figure 5.3.1.1, the initial idea was to fully incorporate the DOT framework and build the whole user interface around that so the user can assign cards into every five fields of the DOT framework. After I showed the mock-ups to the product owner, it turned out to be wrong, because the interface should be more flexible, and less focused on DOT framework overview layout. At this point, I also understand that the application should be a **standalone approach tool**, which **includes** the **DOT framework** principles and not

built only around the DOT framework. However, this design still can be used for an overview, or for a report feature in a later stage. Based on this feedback, new mock-ups were created.

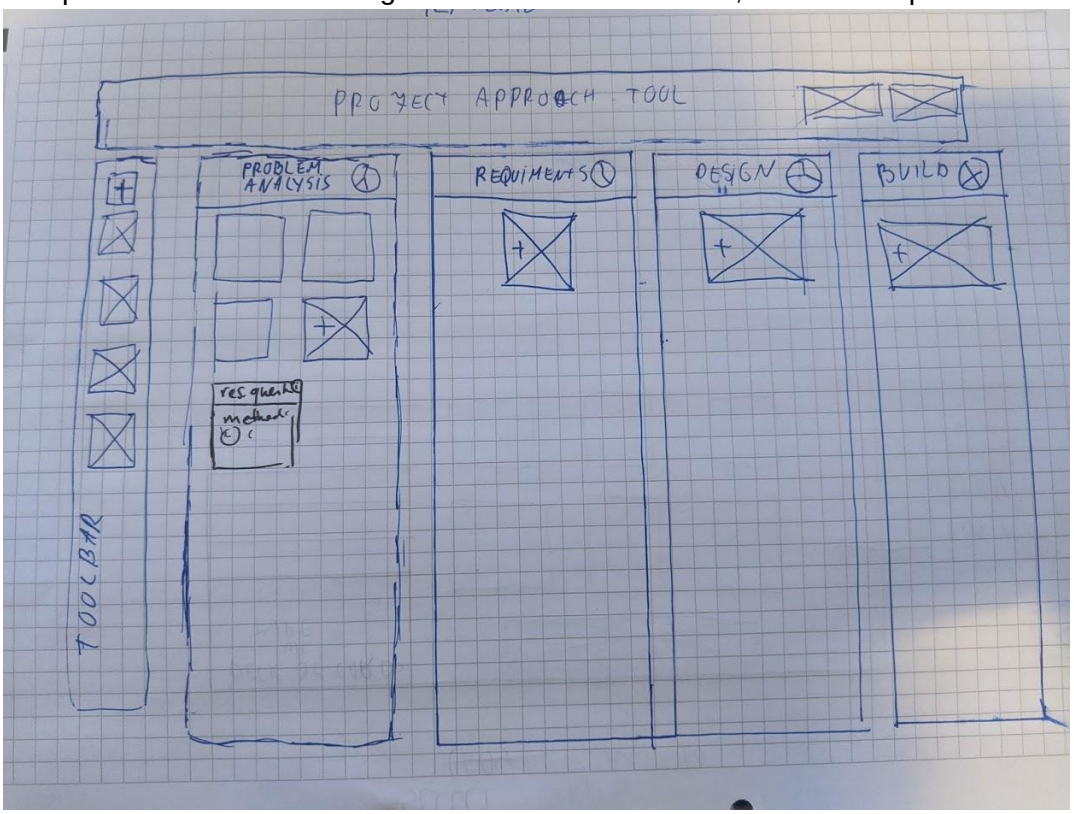


Figure 5.3.1.2 Second mock-up based on the feedback

Figure *5.3.1.2* shows the second try. This version already shows a more versatile interface. On the left, there is a toolbar, which will hold all the usable components of the application. In the middle, there is the “workspace”, in this specific muck up there is a predefined template. However, this part of the application can also be customised by the user, with the available components from the toolbar. The more significant components are the **groups,** and they can be dragged around the **cards** and can be moved between groups. In the top, there is a navbar with a few menu items and the project name which can be modified as well.

5.3.1.1 Proof of Concept

*At this point, the decision was made not to **create** more **mock-ups** nor paper prototypes but create the first iteration of the application.*
The main reason why proof of concept was preferred over mock-ups and paper prototypes:

- With mock-ups, it is hard to show the drag and drop capabilities
- Without knowing the limitations of the chosen to drag and drop library, some functionality can be planned which takes way more time to develop, and possibly leaving less time for other features
- Easy to share it with stakeholders who are not available at Saxion

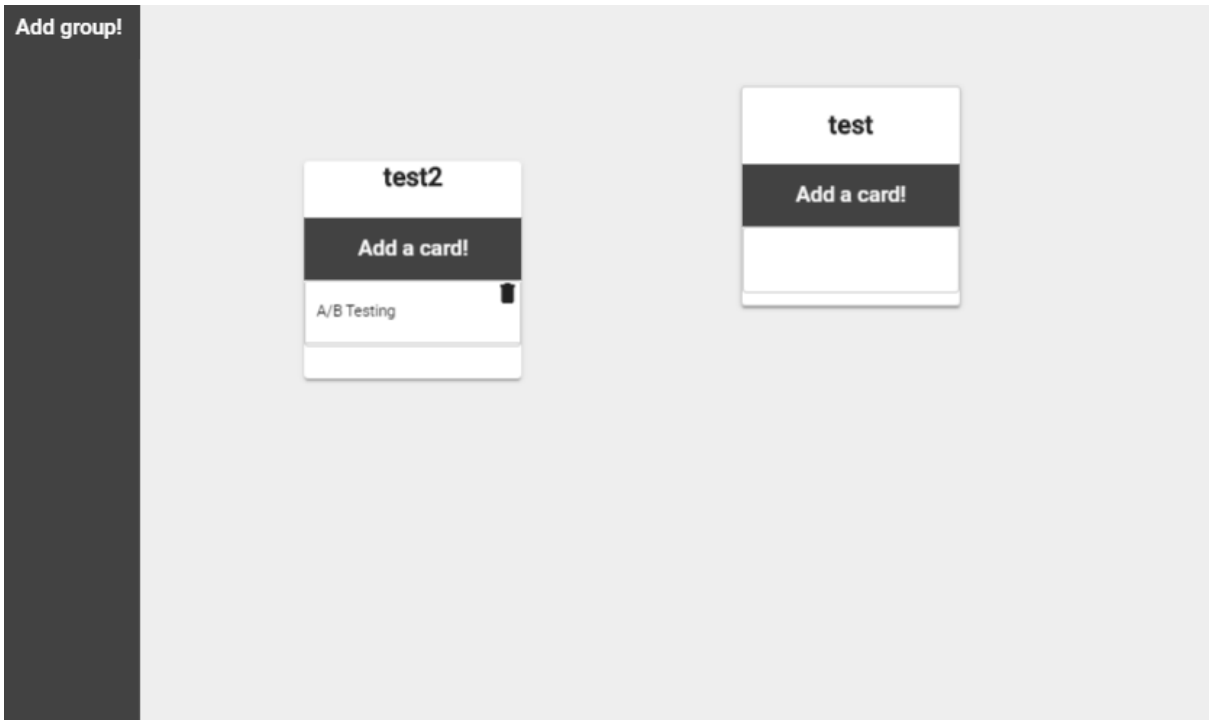


Figure 5.3.1.1.1 The proof of concept

Figure 5.3.1.1.1 shows **proof of concept** application. This is totally different from the mock-ups. On the left side, there is a side panel with an add group button. By clicking this button, a group will spawn on the workspace. After the group spawned on the workspace, the title of the group can be edited, and the group can be dragged around. With the add card button, a pop up is showed and card can be chosen from the list, which will be added to the group, and can be dragged between the groups. This proof of concept was a **huge success**. It proved that the chosen framework was the right decision for this application.

5.3.1.2 The first prototype

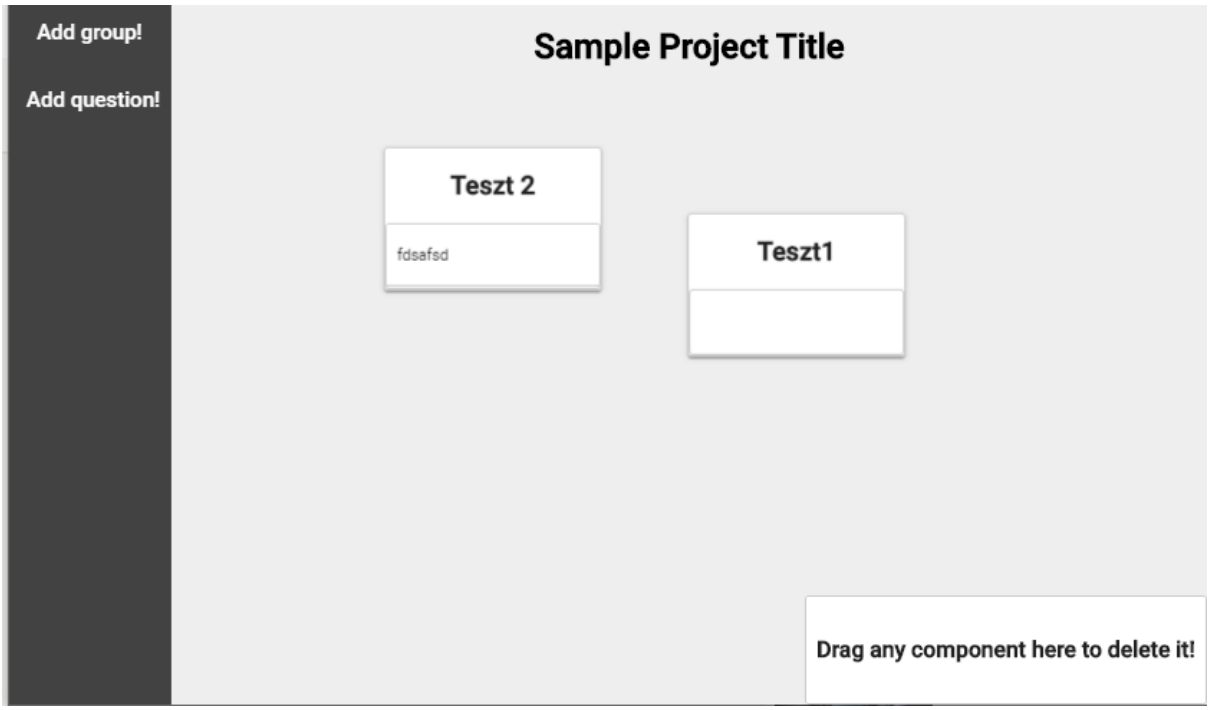


Figure 5.3.1.2.1 First prototype design

For the first prototype, the following design (Figure 5.3.1.2.1) was created. In addition to the previous version, the Add group button was removed from the Group component, and a new card can be added with Right-click, this was done to achieve a cleaner interface which can be presented in reports. An editable project title was added in the top centre of the application. The add question button was added to the side panel; with this button, the user can create a question card. This prototype was presented to the stakeholders during the first stakeholder meeting. In this phase, the stakeholders have accepted the design. Their input was used to create the next iteration of the design.

5.3.1.3 Paper Prototyping

After the first prototype, we went back to paper prototyping. At his point, I had a good understanding of the drag and drop library capabilities, which helped to decide what can be done. The paper prototyping was done together with the product owner during an extended weekly meeting. At this point, we already started to implement the material design guidelines (https://material.io/design/), in order to create a user-friendly interface which meets the current standards.

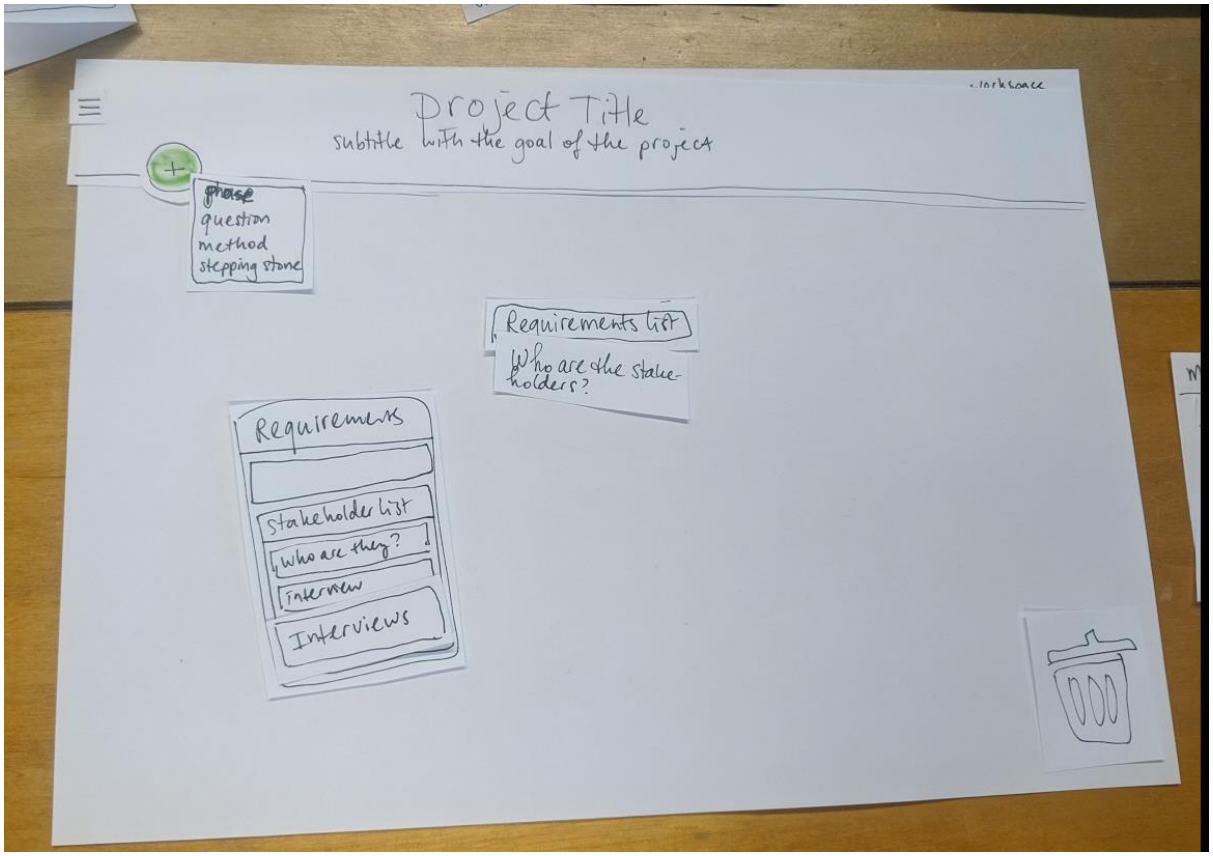


Figure 5.3.1.3.1 Workspace re-design

As it can be seen on figure 5.3.1.3.1, the whole user interface was redesigned while keeping the two main components of the application the same. A top bar was added on top of the workspace. This contains the project title, project goal and two interaction: side menu button and a plus button. Clicking on the plus icon opens a menu. In this menu, every component which can be added to the application will be shown. When clicking on an item, the required interaction will happen. The menu icon toggles the side panel. The delete text was replaced with bin icon, making the interface look cleaner.

5.3.1.4 The second prototype

In the second prototype, the result of the paper prototyping was incorporated with the addition of arrows. The arrows can be drawn between the groups, and if the groups are dragged, the connected arrows move with the group. The groups can be selected and deselected, when they are selected and delete button pressed and the group will be deleted and connected arrows as well.

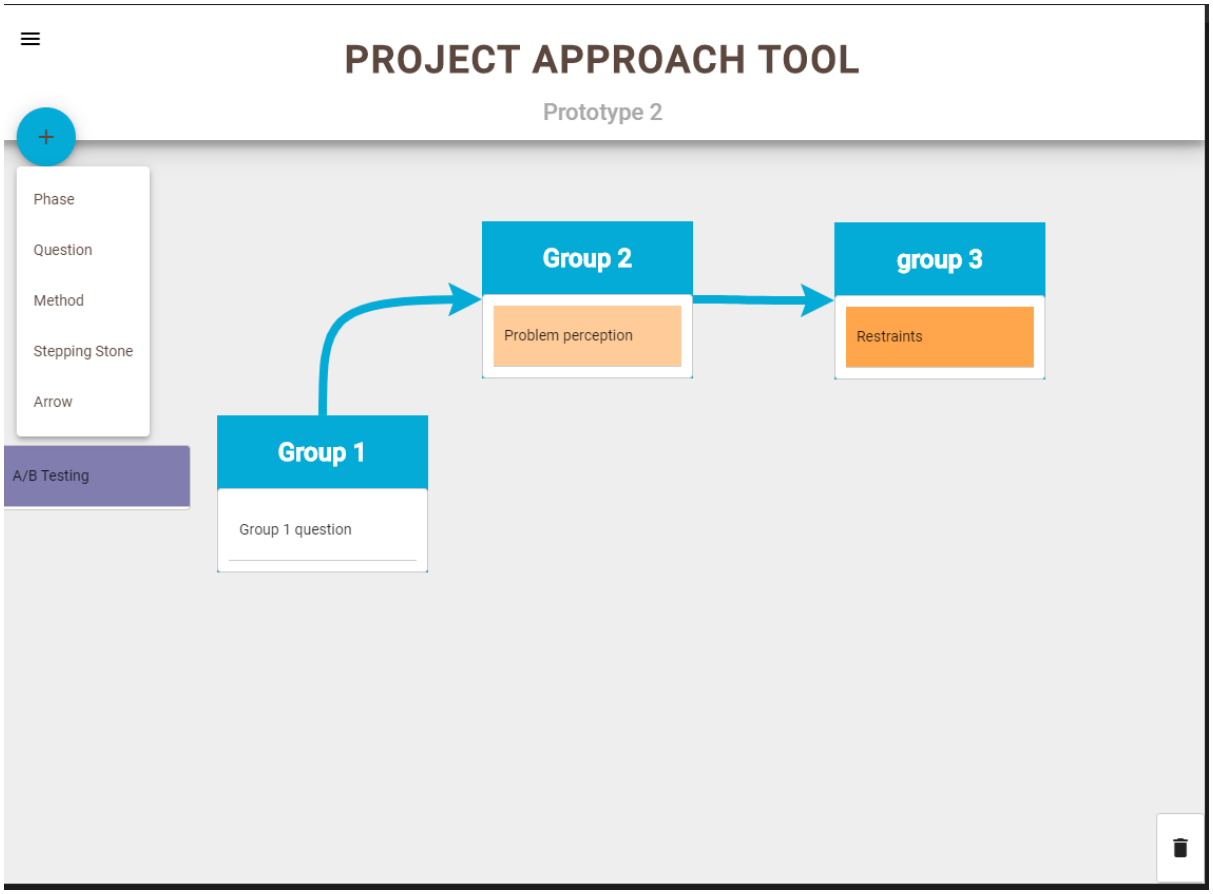


Figure 5.3.1.4 Prototype 2

The title and subtitle stay the same during editing and viewing to maintain consistency for the user. With the introduction of the add menu, adding cards is not possible anymore by interacting with the groups. When a card is chosen from the popup modal, it will spawn in the left side centre in a “spawn group”. This group is just temporarily holder for the cards and questions from here the user can drag the card. When the “spawn group” is empty, it is hidden, once a card is moved out and dropped somewhere the group hides back. The side panel can be opened with the hamburger icon and in this prototype, the only action from here to clear the working space. During the entire project, this second prototype had the most significant user interface update, as this was the main priority.

5.3.1.4 The final product design

5.3.2 Backend

A decision was made in the beginning to create a backend as a first step. The reason for this was that the method cards could be supplied from the backend, instead of hardcoding them into the front-end application. Furthermore, the backend can be extended with additional functionalities. This meant that the back end should be a REST-API.

The initial requirements for the backend:

- Setup a basic NodeJS rest API.

- Create the data structure for the card object
- Create the routing for creating, updating, and getting the cards
- Create the data structure for the workspace object
- Create the routing for creating, and getting the templates (The templates uses the workspace object as well)

The data structure for the Card:

- ID: number
- Title: string
- Type: ID => reference to the type
- Short description: string
- Long description: string
- Notes: string
- Picture: string(URL)
- Questions: string(array)

The data structure for the Workspace:

- ID: number
- Groups: [Groups]
- StoredLines: [Lines]
- Title: string
- Goal: string

The data structure for the Group:

- ID: number
- Title: string
- Location: { x: number, y: number}
- Cards: [Card]
- Selected: boolean

5.4 Framework research

This chapter describes the process of how the two main frameworks were chosen. After all, the requirements were defined, and mock-ups were created, the next step was to research the fundamentals of the front and backend application. In order to be able to compare the different frameworks, **Multicriteria Analysis** was used. The Multicriteria Analysis allowed to compare the essential features of each possible framework and make the final choice by weighting the criteria importance regarding the requirements of this project.

5.4.1 Front end framework

The **most crucial choice** during this project was to choose **the main framework**. The main framework will provide the necessary functions: such as data binding, component creating handling, the basic structure for the application, and can help to achieve better overall performance. The success of this project can depend on the framework choice; therefore, the choice must be made carefully. The three frameworks are **Angular, Vue.js and React**. These three frameworks were chosen because they are the most popular Web application frameworks at this moment. All of these frameworks are open-source and free to use. These three frameworks are the most advanced and used frameworks currently for web applications, and generally, they all can fulfil the basic requirements of the projects such as performance, basic HTML DOM manipulation.

	Angular	**React**	**Vue**
Initial release	2010	2013	2014
Developer	Google	Facebook	Evan You
Official site	angular.io	reactjs.org	vuejs.org
Current version	8.2.14 (13.11.2019)	16.12.0 (14.11.2019)	2.6.10 (20.04.2019)
Language	Typescript	JavaScript	JavaScript
Size	500 kb	100 kb	80 kb
Github Stars	56.1k	142k	155k
Contributors	1k	1.3k	285

5.4.1.1 Defining the criteria with the representative scores.

The four chosen categories, which are relevant for this project are: popularity/community, Security, drag and drop functionality, documentation. While looking at popularity, the primary focus was the active community around the framework. A more popular framework will more likely to get updates and have a bigger chance to find a solution for problems which can occur during development. Since the application will be used by Saxion and hopefully other institutions as well, security is essential. Therefore the security was rated from one to five, and the grading is based on who is behind the framework: how often, and how quickly it gets general and security updates. The next criteria is the most important one, the drag and drop functionality since the whole application will be based around that. The last criteria are the documentation because if the framework has proper documentation, it can really speed up the development process.

	Angular	**React**	**Vue**
Popularity/Community	**1000**	**1300**	**285**
Security (1-5)	**5**	**4**	**3**
Drag & drop functionality (1-5)	**4**	**3**	**2**
Documentation (1-5)	**5**	**5**	**3**

5.4.1.2 Translate the scores into comparable values

In order to be able to compare the previous table data, the scores were transformed into comparable values. From zero to one, where zero is the worst and one highest fit. These values were calculated by dividing the values per ten, where it was possible, if not, then the values were calculated by the percentages.

	Angular	**React**	**Vue**
Popularity/Community	**0.8**	**1**	**0.2**
Security (1-5)	**1**	**0.8**	**0.6**
Drag & drop functionality (1-5)	**0.8**	**0.6**	**0.4**
Documentation (1-5)	**1**	**1**	**0.6**

5.4.1.3 Scores with weight

In this step, the weight was assigned to the criteria based on their relevance to the project. The most important criteria are the drag and drop functionality since this is a core feature of the application.
Weights:

1. 0.5
2. 1
3. 1.5
4. 1

The values with added weight:

#		Angular	React	Vue
1.	**Popularity/Community**	**0.4**	**0.5**	**0.16**
2.	**Security (1-5)**	**1**	**0.8**	**0.6**
3.	**Drag & drop functionality (1-5)**	**1.2**	**0.9**	**0.6**
4.	**Documentation (1-5)**	**1**	**1**	**0.6**
	Total Scores:	**3.6**	**3.2**	**1.96**
	Ranking:	**1**	**2**	**3**

As can be seen in the table, the highest-ranking framework is **Angular**. Therefore, Angular is the most suitable choice and will be the **final choice**.

5.5.1 Back end framework

Based on the functionality of the application, it makes sense to have a supporting backend. Mainly because the cards can be stored in the global database instead of hardcoding them into the frontend application. This means it is easier to add, edit or remove components from the system. Besides this main functionality, it can provide a user management system, with cloud workspace synchronization functionality and many more possibilities for future development. The main backend functionality will be data handling between the database and application which means it is REST-API. Therefore, most of the current frameworks can fulfil the requirements quickly. Three frameworks were chosen, and the Multi-criteria analysis tool will be used to find the best fit for this application. The three frameworks are **Django, Golang and ExpressJS**(Nodejs).

	Django Rest Framework	**Golang**	**ExpressJS**
Initial release	2014 (3.0.0)	2009	2010

Developer	Django Software Foundation	The Go Authors(Google)	TJ Holowaychuk, StrongLoop
Official site	https://www.django-rest-framework.org/	https://golang.org/	https://expressjs.com/
Current version	3.10.3(29th July 2019)	1.13.4 (October 31, 2019)	4.17.1 (May 25, 2019)
Language	Python	Go	JavaScript
Github Stars	16.2K	67.9k	46.9k
Contributors	0.9k	1.4k	230

5.5.1.1 Defining the criteria with the representative scores.

Four criteria were chosen for the comparison, which is relevant to this project. The first criteria are the programming language. The programming language and the ease of use categories are important because it should be a well known easy to use language, so there is no time needed to learn a new language. Popularity was chosen because the bigger community means more tutorials, more questions already answered, which helps to speed up the development. Performance for an API is always a key since the user experience is heavily influenced by loading times.

	Django	Golang	ExpressJS
Programming language	Python	Go	JavaScript
Popularity/Community	900	1400	230
Ease of use	3	2	5
Performance	4	5	4
Documentation	4	3	5

5.5.1.2 Translate the scores into comparable values

In order to be able to compare the previous table data, I convert those scores into values from zero to one, where zero is the worst and one highest fit. These values were calculated by dividing the values per ten, where it was possible, if not, then the values were calculated by the percentages.

	Django	Golang	ExpressJS
Programming language	0.5	0	1

Popularity/Community	0.8	0.7	1
Ease of use	0.6	0.4	1
Performance	0.8	1	0.8
Documentation	0.8	0.6	1

5.5.1.3 Scores with weight

The weights were assigned based on those categories which help with quick and easy development process but also keeping in mind the future development.
Weights:

1. 1
2. 0.8
3. 1.5
4. 0.5
5. 1

The values with added weight:

#		Django	Golang	ExpressJS
1	Programming language	0.5	0	1
2	Popularity/Community	0.64	0.56	0.8
3	Ease of use	0.9	0.6	1.5
4	Performance	0.4	0.5	0.4
5	Documentation	0.8	0.6	1
	Total Scores:	3.22	2.26	4.7
	Ranking	2	3	1

Based on the research, the highest-ranking framework is **ExpressJS** by scoring high on the ease of use criteria**.** Therefore this framework will be used to develop the application.

6. Product Realisation

Based on the requirements, and result of the research, the product was realized. In this chapter, the realization of the product is explained, including implementation choices with reasoning, implementation challenges and the final implementation. This chapter is divided into two parts, explaining the front and back-end implementation. As it was mentioned in the planning chapter, the main focus was on to create a Minimum Viable Product. Therefore the initial backlog was filled with the user stories regarding the MVP. The explanation will be in the same order as the development was done: back-end first and front end secondly,

therefore it is easier for the reader to understand why and how a certain decision was made. Starting with backend resulted in a quicker and easier front end implementation because there was no need to hard code the data in the front end application.

Both the front and back-end application exist in GitHub repository provided by Saxion, including readme files on deployment as well as on development instructions.

6.1 Back-end Application (API)

This chapter consists of the description of the realization of the back-end application, including implementation choices and final implementation.

6.1.1 Implementation choices

Besides the initials research, a few smaller choices were required before the implementation.

As mentioned before the main framework for the backend will be **ExpressJS**(ExpressJS), there NodeJS needs to be used with the NPM package manager.
As a database choice, **MongoDB** (MongoDB) was chosen. First of all, this is a NoSQL (non-relational, schemaless) database, which is commonly used for real-time web applications. With a NoSQL database, it is easier to manage dynamic and unstructured data, which could be handy in this project. MongoDB is also one of the most used database solution used together with Express.js. Therefore, a lot of information and tutorial is available online.
To connect MongoDB and ExpressJS, a database ORM (Object-Relational-Mapper) is required, which helps to write easy to understandable queries using object-oriented paradigms. For this purpose, **Moongose** (Moongose) was chosen. Mongoose is explicitly written for using it with NodeJS and MongoDB; it is very well documented and has excellent performance, but most importantly, it can provide the required functionality with quick development time.
In addition, a few other smaller libraries were required:

- Body-parser: A middleware to handle HTTP request bodies.
- Morgan: A middleware for logging HTTP requests into the console.
- Forever: Required to run the backend application on the server, while the SSH session is closed.
- Nodemon: A developer tool which automatically restarts the node application when a code change is detected. (only required during development)
- express-oas-generator: Generate swagger API documentation.

The back end was designed in a way to keep the development and maintenance time to the absolute minimum, leaving more time for the front-end application.

6.1.2 Implementation

The first step was to create the fundamentals of the application. During the development, the ExpressJS guidelines were followed to set up the server, including the routing and folder structures.

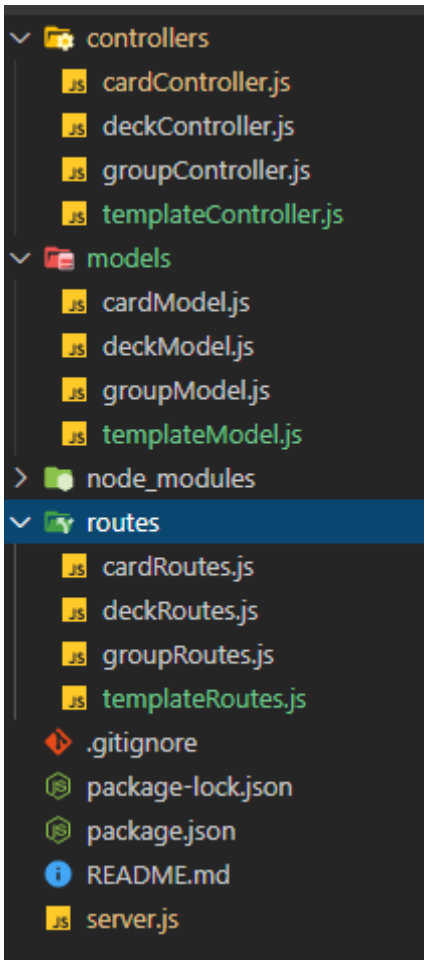


Figure 6.1.2.1 API directory structure

The final directory structure can be seen in Figure 6.1.2.1. Each component of the application has it is own subfolder; in this way, it easy to maintain, add or remove features. The entry point of the application is the server.js. This file contains the actual HTTP server, the connection to the Database via Moongose, and takes care of registering the routes for the application. The controllers are responsible for the data handling between the API and database. The models are representing the data structure mentioned earlier. Furthermore, the route files are managing the routes for each controller.

Figure 6.1.2.2 shows the implemented API calls

Figure 6.1.2.2 shows all the implemented and used API calls. More detailed information about API can be found on API address/api-docs/#/ (http://145.76.18.86:13788/api-docs/#/.

6.1.2 Testing

The API was tested with the Insomnia (Insomnia) rest client, and each new functionality was tested during development as well as the whole application after each commit.

6.2 Front-end Application

This chapter consists of the description of the realization of the front-end application, including implementation choices, final implementation, challenges and exciting choices. *This chapter, organized in chronological order, represents the actual development process.*

The fundamentals of the application were generated with the Angular CLI. During the whole project, the Angular standards were used for naming conventions, folder structure, component handling. For creating the user interface components (for example buttons, lists, cards, icon), **Angular Material** (Angular Material) library was used. This library is the official Material design library for Angular, developed by Google. *Standard Angular (such as data binding, directives) features will not be explained in this document, because the purpose of this report is to focus on the project-specific compelling solutions and implementations. However, where necessary, a short introduction will be presented. Detailed information can be found on the official Angular website*

6.2.1 Component Creation

The first step was to create the two fundamental components of the application: the group and a card. For each of this element, two Angular components were created. In this way, the logic and design of these elements can be separated.

The card

```
<div class="card-container" (click)="openDetail()" *ngIf="!canEdit" >
    {{card.Title}}
</div>
<mat-form-field *ngIf="canEdit">
    <input #question matInput placeholder="Question" [(ngModel)]="card.Title" (keyup.enter)="saveTitle()" autofocus />
</mat-form-field>
```

Figure 6.2.1.1 Card component

The card component has only an editable title (this functionality will be explained later). The title of the card is shown with one-way data binding using double curly braces, whereas the input value is binding with two-way data binding using ngModel. The ngIf directives use a boolean value from the component typescript code, in order to show or hide the required element. The input field is used for question cards.

The group

```
<div class="group-card">
  <div class="group-header">
    <div id="title" *ngIf="!canEdit">
      <h2 (mousedown)="check($event)" (click)="editTitle($event)">{{ group.title }}</h2>
    </div>

    <mat-form-field class="example-full-width" *ngIf="canEdit">
      <input #name matInput placeholder="Title" [(ngModel)]="group.title" (keyup.enter)="saveTitle()"
        (focusout)="saveTitle()" appFocusOnShow />
    </mat-form-field>
  </div>
  <div class="group-content">
    <div cdkDropList id="{{ group.title }}" [cdkDropListData]="group.cards" class="card-list"
      (cdkDropListDropped)="drop($event)">
      <app-card cdkDrag *ngFor="let card of group.cards" class="card-listitem"
        [ngStyle]="{ 'background-color': card.Color }" [card]="card" (save)="dropped.emit()"></app-card>
    </div>
  </div>
</div>
```

Figure 6.2.1.2 Group component

The group component is a bit more complicated than the card. It has two main parts: the header of the card which holds the title, which can be edited. The second parts hold a list of the card components. The functions of this group will be explained later in this paragraph.

6.2.2 Drag and Drop

The drag and drop functionality was the first significant step in creating the components since this is the core functionality of the application. In order to make the components draggable, the Angular Material drag and drop functionality (Angular Material Drag and Drop) was used. As it was stated in the research paragraph, the mean reason why Angular was chosen because it has the most advance and easy to use drag and drop capabilities which do not require any third-party library. This is important because this library will always be updated as long as Angular is updated, and will always be compatible with the latest version of Angular. This drag and drop feature is relatively easy to use once the developer understands the main elements of it. These elements are explained from the 'smallest to the biggest' hierarchy, for more natural understanding.

The Card

The smallest draggable element is the card, component, these cards are just draggable HTML elements, and they can be dragged in between groups.

```
<app-card cdkDrag *ngFor="let card of group.cards" class="card-listitem"
  [ngStyle]="{ 'background-color': card.Color }" [card]="card" (save)="dropped.emit()"></app-card>
```

Figure 6.2.2.1 Card element with cdkDrag

Figure 6.2.2.1 shows the card object, which is represented in a group. To indicate the card is a draggable element, the cdkDrag directive is added to every generated card object in every group.

The Group List

```
<div cdkDropList id="{{ group.title }}" [cdkDropListData]="group.cards" class="card-list"
  (cdkDropListDropped)="drop($event)">
  <app-card cdkDrag *ngFor="let card of group.cards" class="card-listitem"
    [ngStyle]="{ 'background-color': card.Color }" [card]="card" (save)="dropped.emit()"></app-card>
</div>
```

Figure 6.2.2.2 Group list element with cdkDropList

The group list holds each card element within a group. Between these groups, the cards can be moved, and cards can be reorganized in between a group as well as creating a linear hierarchy. A list item is indicated with cdkDropList directive for Angular, and it requires an attached list which supplies the data for the list. This data must be supplied with the [cdkDropListData] directive.

The Group

The group elements represent the title of the group and group list. The groups can be dragged around the workspace freely. The group element is the most complex part of the application, as it can be seen in Figure 6.2.2.3.

```
<app-group class="group-item" *ngFor="let group of workspace.groups" [id]="group.id" cdkDrag [group]="group"
  [cdkDragFreeDragPosition]="group.location" (cdkDragMoved)="drag(group)" (cdkDragEnded)="onDragEnded($event,group)"
  (mousedown)="check($event)"
  (click)="drawing ? addArrow(group) : select(group, $event)" (dropped)="updateLocalStorage()" [class.selected]="group.selected" >
</app-group>
```

Figure 6.2.2.3 the Group element

The group element is also a stand-alone component, and cdkDrag directive makes it draggable. However, the group can be dropped anywhere on the workspace because it does not have a parent cdkDropList, unlike the Card element. (Rest of the directives and listener will be explained later with the feature for which it required.)

The workspace

The most significant component is the workspace container, which holds all the draggable elements of this project.

```
<mat-sidenav-content class="workspace-container" cdkDropListGroup>
  <mat-toolbar>···
  </mat-toolbar>

  <div id="spawn-list" *ngIf="spawnList.cards.length > 0">···
  </div>

  <app-group class="group-item" *ngFor="let group of workspace.groups" [id]="group.id" cdkDrag [group]="group"···
  </app-group>

  <div id="trashcan" class="ths">···
  </div>
```

Figure 6.2.2.3

In order to connect all of the different group lists and the trashcan as well as the spawn list, we have to create a cdkDroptListGroup. All the components which are children elements of this side-nav-content and have cdkDropList directive added to it will be connected into this drag group, that is why the cards can be moved between the groups.

```
drop(event: CdkDragDrop<string[]>) {
  if (event.previousContainer === event.container) {
    moveItemInArray(
      event.container.data,
      event.previousIndex,
      event.currentIndex
    );
  } else {
    transferArrayItem(
      event.previousContainer.data,
      event.container.data,
      event.previousIndex,
      event.currentIndex
    );
  }
```

Figure 6.2.2.4

Figure 6.2.2.4 shows the logic which moves the card between groups, and this is triggered when the card is dropped (the user released the card).

Once I got a good understanding of these essential components and arranged them in a proper way, the drag and drop functionality started to work. It is still required a fair amount of time to get it right. However, far less, then we planned for which prove that the initial research regarding the main framework was a success.

6.2.3 The modals

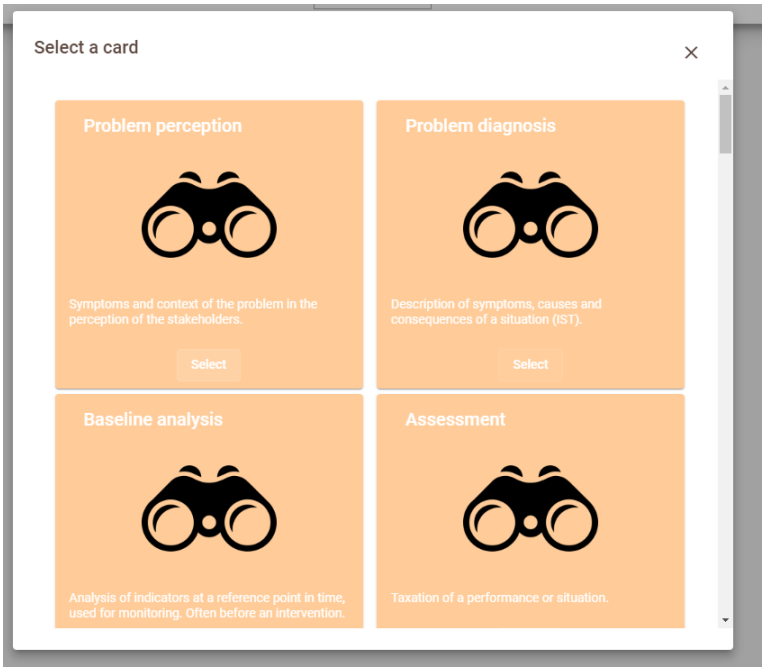


Figure 6.2.3.1 The card selector modal

The application requires three modals for selecting a card, selecting a template, and showing detailed information about the card. The modals are regular angular custom components, but they are opened dynamically from the typescript code.

```
addMethod() {
  const dialogRef = this.dialog.open(CardselectorComponent, {
    width: '800px',
    data: { steppingStone: false }
  });
  // Wait for the user select card in the dialog
  dialogRef.afterClosed().subscribe(result => {
    if (result) {
      this.spawnList.cards.push(result);
    }
  });
}
```

Figure 6.2.3.2 open a dialogue/modal

Opening a dialogue is very simple, as it can be seen in Figure 6.2.3.2. In this specif version, the width of the dialogue is defined, and essential data is shared with the modal. While the dialogue is open, a subscription is set to listen when the modal is closed. The modal can be closed, without a selection, and with selection. If nothing is selected and modal is closed, nothing will happen. When a user selected a card or template, a required functionality will be called.
In the card selector modal, the cards are separate components than the modal, and the select button is part of the card component. Therefore the app component has to notify the parent modal component about the selection and can return the selected card to the workspace. This functionality is achieved by using Eventtransmitter, where the parent component listens to child component action.

6.2.4 Inline edit text fields

In this application, there quite a few places where a text needs to be edited. These texts which are editable are part of the application, such as title, the goal of the project, group title. Stakeholders required that the editable text should look the same as the final text, so the user can see how it will look like after the editing is finished.

Figure 6.2.4.1 Title and goal while editing

Figure 6.2.4.2 Title and goal after editing

Figure 6.2.4.1 show the text field during editing, and Figure 6.2.4.2 shows the text when it is not in edit mode. Of course, the user can always edit the title by clicking on the text.

```
<span id="title-container">
  <input *ngIf="canEditTitle" class="title" #name [(ngModel)]="workspace.title"
    (keyup.enter)="moveFocusToSubtitle()"  appFocusOnShow AutoSizeInput [extraWidth]="80" placeholder="Title" >
  <p class="title" #title *ngIf="!canEditTitle" (click)="canEditTitle = true">{{ workspace.title }}</p>
</span>
```

Figure 6.2.4.3 Inline edit HTML code

Figure 6.2.4.3 shows the implementation of the title HTML element. It consists of two child elements in one parent span element (the span element is required by the angular material toolbar). The child uses the ngIf directive in order to show and hide the required element based on user interaction. When the user clicks on the title, or in case of the new workspace, the title is editable, and when the user presses the enter, the title is saved.

6.2.4.1 Autosize input

Unfortunately, the input HTML element is not a responsive element, which means it has a predefined width and does not shrink or append based on the content. In order to solve this problem, a secondary <p> element was created and which has the same attributes as the input field, and the same content. When the user enters the text, the secondary <p> element is measured and applied to the input element. This solution is sort of worked and introduced much ugly code. Therefore a little bit more research was required, which resulted in an already existing Angular directive (ngx-autosize-input). This directive takes care of the described issue while keeping the code clean and easy to understand.

6.2.4.2 Autofocus

For each input text, it was required to get the focus when they are shown, which meant for each of these elements, I had to add the same line of code. However, in order to keep the code clean, I create a custom Angular directive, which can be added to any HTML element, and when the element is rendered, it will set the focus automatically, and the user can start to type directly. This can be added to an element by calling the appFocusOnShow directive in the HTML template.

Creating the inline feature required almost the same amount of time as a drag and drop feature, which is interesting since the drag and drop is a more crucial part of the final product.

6.2.5 The arrows

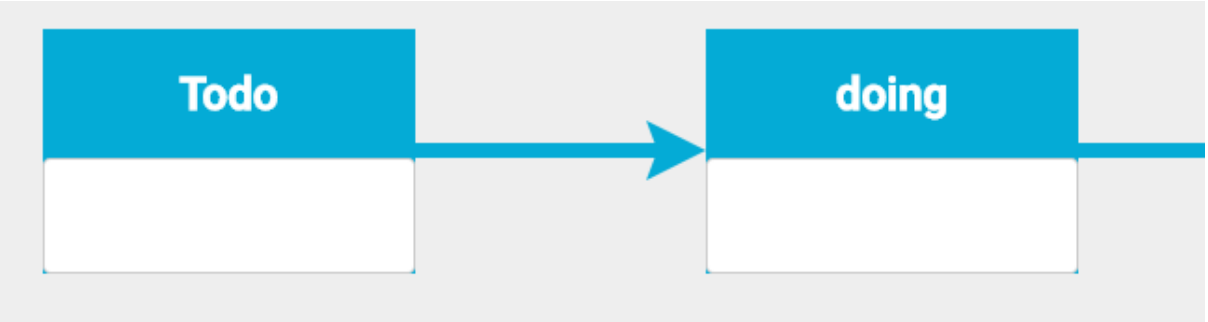


Figure 6.2.5.1 Arrow functionality

Without the arrows, the application would almost make no sense. Therefore it was essential to create the best possible arrow functionality. Before the development could begin, research was needed. In order to find a proper solution, a few research criteria were defined.

- The library should not use Jquery, only pure javascript since Angular not support Jquery by default
- The library should be as small as possible, preferably only contain the arrow functionality
- The arrows should position themself automatically, around the groups
- The arrows should be able to move, while the groups are dragged
- The arrow parameters should be easily changeable

During the research, three possible solutions ticked all the requirements.

- Konva JS (https://konvajs.org/)
- Leader Line (https://github.com/anseki/leader-line)
- Create the functionality from scratch with HTML5 svg.

For each solution, a git branch was created, so the changes do not net to be reverted, and the final solution can be merged back into the master branch. First, I started with the KonvaJS, as I had previous experience with this library. After adding this library to the project, I realized that this library only could work with a canvas, and to that canvas, only Konva objects can be easily added. This meant, to continue with this library, the whole drag and drop system should be moved into the Konva ecosystem, which will result in a time-consuming process.
After realizing that Konva is not visible, the next step was the Leader Line library. This library only provides the required arrow functionality without other unnecessary features. In order for this library work, an HTML canvas is not required, so the already built components do not need any modification. Creating a line is pretty easy with this library, as shown in Figure 6.2.5.2.

```
// tslint:disable-next-line: no-unused-expression
line = new LeaderLine(
  this.document.getElementById(this.connect[0].id),
  this.document.getElementById(this.connect[1].id),
  this.leaderLineOptions
);
this.lines.push(line);
```

Figure 6.2.5.2 Creating a new arrow

The constructor requires two HTML elements and optional leader line options. After adding the line into two hardcoded sample group, it worked. At this point, the decision was made to not try any more solution but proceed with the LeaderLine library.

The first challenge was to draw a line between the two selected groups. When the user selects the arrow function form the menu, a “drawing mode” is entered. The user has to select to groups, and the arrow will be drawn in between. The arrow will point from the first selected group to the second selected group. The created line object will be stored in a list, in the context of the workspace component. This is necessary to interact with the line on user interaction (for example, moving or deleting a group). The line also has to be stored in

the workspace, so on reload, the arrows can be redrawn. About each line, the id generated by the LeaderLine constructor is stored alongside the two groups which it connects.

The second challenge was to move the lines when the groups are moving. Unfortunately, this feature is not implemented in the library. The drag and drop API provides a call when the user is dragging the element, cdkDragMoved. This method is called by every pixel of the movement, which means the line can be updated in real-time.

```
(cdkDragMoved)="drag(group)"
```

Figure 6.2.5.3 cdkDragMoved call

This call can be attached in the HTML template, and a group object was attached to call as an argument.

```
// Update line as group is being dragged in real time
drag(group: Group) {
  this.workspace.storedLines.forEach(line => {
    if (line.start === group.id || line.end === group.id) {
      this.lines.forEach(ln => {
        if (ln._id === line.id) {
          ln.position();
        }
      });
    }
  });
}
```

Figure 6.2.5.4 The arrow update method

To update the arrows, two for loop is required. First, the application should find all the connected lines for the moving group in the array of the workspace. If the line is found, it has to loop through the context line group, which holds the Leader line object, that is the reason why the workspace holds the LeaderLine id. When the two ids are matching the line object can be updated. This can be done by calling the ln.position() method, which takes care of the updating (this method is from the LeaderLine library). This works flawlessly. However, there was concern regarding performance issues; when there are multiple lines connected to the group, maybe it can become laggy. This was tested on multiple low-end devices, with a high amount of lines. However, no performance issue was found.

6.2.6 Save and restore workspace status

If the workspace is cleared on every refresh of the page, there is no added value to the end-users; thus, it was important to solve this issue. At this point, a decision was made to store workspace in the browser local storage but design the system in such a way that it can be easily refactored to work with the API. The reason behind this decision was that in order to store the workspace in the cloud, a user system would need to be developed as well, and meet all the Saxion, and GDPR requirements. Creating such a system would take up a huge

amount of time, and the product owner decided not to develop a login system. The local storage store the data with no expiration date. The data will not be deleted when the browser is closed and will be available the next day, week, or year. Before the data is saved into the local storage, it is converted into a JSON file, when reading it back it converted back to a javascript object. After each user action, the local storage is updated. This can be done fairly easy.

```
updateLocalStorage() {
  localStorage.setItem('workspace', JSON.stringify(this.workspace));
}
```

Figure 6.2.6.1 Save to local storage

When re-opening the web page, the application checks if there is workspace stored in the database, this done in the ngOnInit Angular lifecycle hook.

```
ngOnInit(): void {
  if (localStorage.getItem('workspace') === null) {
    this.openTemplateModal();
    // open modal for choose a template
    this.workspace = new Workspace();
    this.workspace.groups = [];
    this.workspace.storedLines = [];
    this.lines = [];

  } else {
    this.workspace = JSON.parse(localStorage.getItem('workspace'));
    this.canEditTitle = false;
    this.canEditGoal = false;

    setTimeout(() => this.drawArrows());
  }

  this.mouseTracker = this.document.getElementById('mouseTracker');

}
```

Figure 6.2.6.2 NgOnInit with reload functionality

The arrows have to be drawn back, but first, all the HTML element must be added to the HTML dom by Angular. Otherwise, the arrows can not receive the start and end elements. In order to fix this issue, a setTimout functionality needs to be called, which will fire the drawArrows method once the HTML is ready.

6.2.7 Testing the front-end application

During the development process, three types of testing were done.
Functional testing was done after each functionality were added, to the system by the developer as well as each progress meeting together with the product owner. Functional testing is a type of software testing that validates the software against the functional

requirements/specifications. The functional testing focuses more on the user interface, API, client-server communication rather than the source, hence it was chosen for this project as the main goal is to create a user-friendly application.
Stakeholder testing was done twice during the stakeholder meetings. The user testing with real student group was done by the product owner because I did not have a chance to be there in person. However, the product owner collected the feedback from the student, so in the next progress meeting, it was discussed and formulated into requirements.

7. Conclusion and future recommendation

The goal of this graduation assignment was to create an **application** which can provide a project approach tool for projects. Requirements were set up, a process was defined, designs were created, a research was done, and the application was implemented. The final product proves that the initial research regarding the frameworks was successful. The realized Angular web application together with the created API, help project groups to start their project with a good plan.

When comparing the requirements with the final product, it can be concluded that almost **all of the requirements have been implemented**. During the development phase of the project, the requirements were changed and reprioritized in order to have a usable product in the end. The result of this approach worked well for this assignment since the final product has a great added value for the project groups. When comparing the first mockups to the final realized interface, the differences can be clearly seen. The differences show how the application involved based on the feedback from the stakeholders.

The final conclusion of this project is that the main goal of the assignment has been achieved and the stakeholders have accepted the final product.

However, the created product is not a feature-complete application yet. Therefore I would like to address my suggestion for future development.

One of the most important features is the **user login system**. This would add the possibility to save the workspace in the cloud so the user can access it on multiple devices, and can share it to other users. This functionality, however, would require the application to meet all the GDPR rules, as well as Saxion policies for storing user data.
A user can create custom cards; however, it would be useful if those custom card decks can be shared. This would require a review system, where teachers can review those decks before adding it to the application.
When a user using a certain card combination, the system could ask questions which makes sure that the user using the cards in their intended way. This could be extended with an AI providing even more help for the users.

8. Version

Date	Note
12.02.2020	Initial draft version
22.02.2020	Second draft version
23.03.2020	Final draft version

9. References

Angular (03-2020) Retrieved from https://angular.io/

Vue.js (03-2020) Retrieved from https://vuejs.org/

React (03-2020) Retrieved from https://reactjs.org/

DOT framework (03-2020) Retrieved from
https://maken.wikiwijs.nl/129804/DOT_framework__EN_

Express (03-2020) Retrieved from https://expressjs.com/

Django (03-2020) Retrieved from https://www.djangoproject.com/

Go (03-2020) Retrieved from https://golang.org/

Leader Line (03-2020) Retrieved from https://www.npmjs.com/package/leader-line

Konva (03-2020) Retrieved from https://konvajs.org/

Material Design Guidelines (03-2020) Retrieved from https://material.io/design/

Angular Material (03-2020) Retrieved from https://material.angular.io/

Visual Studio Code (03-2020) Retrieved from https://code.visualstudio.com/

10. Appendix

10.1 Initial requirement list

Nonfunctional requirements

#	Requirement	Explanation
1	The application must be a web application	

2	The application must have a backend (API)	
3	The application must be easily usable for multidisciplinary users	
4	The application must be supported Chrome	
5	The application should support Firefox, Safari and Microsoft Edge	
6	The application must support desktop and laptop size screens	
7	The application should support tablet size screens	
8		

Functional requirements
Must have requirements(For the Minimum Viable Product)

#	Requirement	Notes
1	As a user, I want to be able to create groups so I can assign cards to it.	
2	As a user, I want to able to drag and drop the groups, so I can organize them in order to fit the project	
3	As a user, I want to be able to edit and delete groups.	
4	As a user, I want to able to assign title into the groups, so I can easily identify them.	
5	As a user, I want to able to assign cards into the groups.	
6	As a user, I want to be able to click on the card so I can reveal more information about them.	This applies to the card which is already assigned to a group, so it is visible on the workspace
7	As a user, I want to able to move cards between groups, so I can organize them.	
8	As a user, I want to able to remove cards from the groups.	
9	As a user, I want to be able to browse the cards, so I can choose or view more information about them	

10	As a user, I want to be able to create a custom card so I can personalize the tool for the project	
11	As a user, I want to be able to choose a deck of cards, so I can look through them easier.	
12	As a user, I want to able to see the deck colour of the card in the group list, so I can get a quick overview easily	
13	As a user, I want to able to define the title for the project, so the project is recognizable	
14	As a user, I want to able to assign Research Question to the groups, so I can connect the cards to it.	

Should have requirements

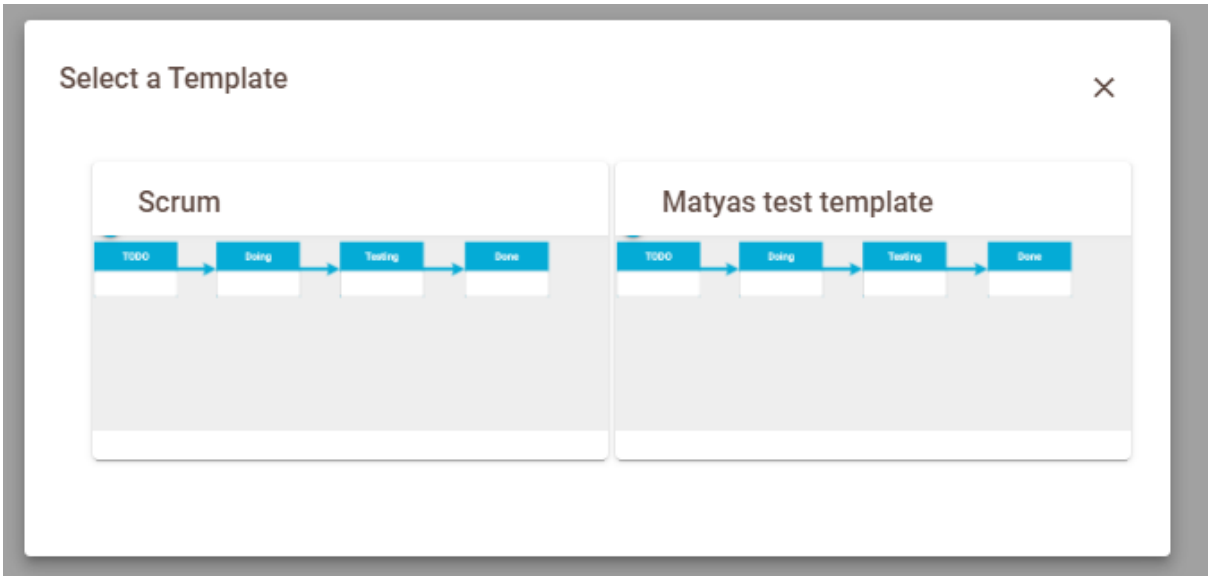
#	Requirement	Note
1	As a user, I want to able to receive question when I select a card.	The user should be noticed if the card has a question assigned to it.
2	As a user, I want to be able to differentiate between the cards by the deck colours	Colour Schema of the DOT framework
3	As a user, I should be able to select a template, so I can start planning easier and quicker.	
4	As a user, I should be able to connect groups to each other with arrows	
5	As a user, I should be able to log in and save the current working space to my account.	
6	As a user, I should be able to continue working on the project after I closed the browser tab.	The workspace state is saved in localstorage
7	As a user, I should be able to connect groups with arrows, so I can indicate the connection between them	
8	As a user, I should be able to choose from predefined templates, so I can start planning quicker	
9	As a user, I should be able to assign deadlines to the groups.	

Could have requirements

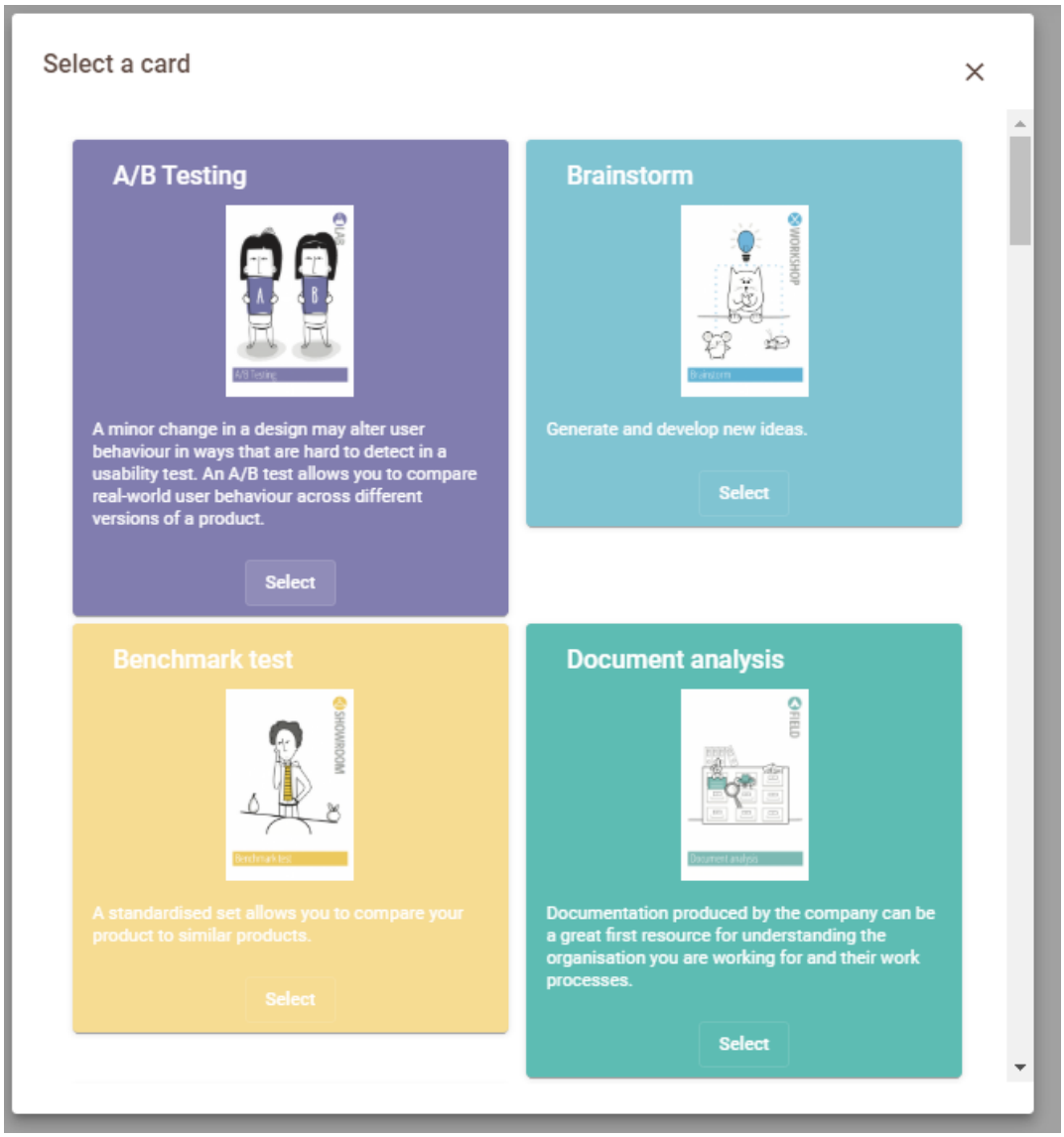
#	Requirement	Note
1	As a user, I want to able to share my custom cards (decks) to the application userbase, so others can use it as well.	
2	As a user, I want to be able to log in and access my workspace from a different computer	
3	As a user, I want to able to create multiple workspaces, so I can have multiple projects saved.	
4	As a user, I want to be able to export and import workspace state, so I can transfer it to different browsers	
5	As a user, I would like to able to use report functionality so I can include the workspace in my documentation	Clean print screenable look.
6	As a user, I would like to assign boost to the cards	
7	As a user, I want to able to follow an overlay tutorial when I launch the application for the first time, so I can learn the features of the application.	

10.2 Application Designs

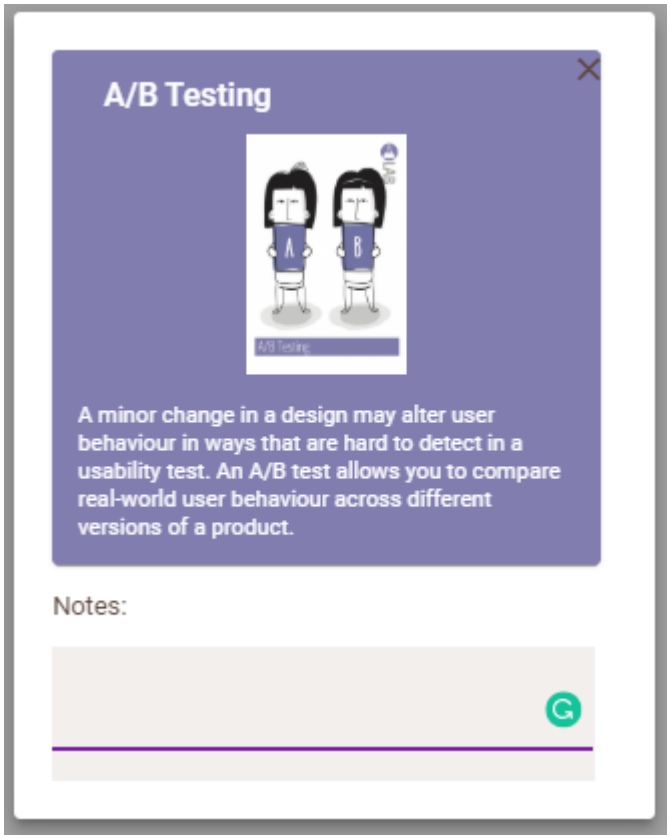
Template selector



Card Selector



Card detail view with notes



Dropdown menu with material design