VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

FACT - A Micro Learning Management Tool

Callum Gill

Supervisor: Alex Potanin

Submitted in partial fulfilment of the requirements for
Bachelor of Engineering with Honours.

Abstract

A short description of the project goes here.

Acknowledgments

Any acknowledgments should go in here, between the title page and the table of contents. The acknowledgments do not form a proper chapter, and so don't get a number or appear in the table of contents.

Contents

Figures

Chapter 1

Introduction

This project aims to develop a product that solves the problem of presenting essential information to employees in a concise, easy to read and measurable way. This project was suggested as further development of a proof of concept that HAUNT, a small web development company suggested. HAUNT acted as an industry supervisor for this project development as well as product owner. The product aims to be a micro learning management system that reduces a lot of the overhead of HR around keeping employees informed about procedures and policies within a company. This report will cover the motivations behind this product, the development and implementation of the product, engineering choices involved and the evaluation of the product with expert feedback. As this product will be developed in a start-up environment this report will also cover the challenges that software development faces in these environments.

1.1 Haunt

HAUNT is a start up company that develops and maintains various websites. As they are a new company they have recently gone through the human resource management of hiring new employees and informing them of company policies, health and safety and information on the company and relevant technology. They identified the potential need for a system to easily convey and manage this information quickly and efficiently.

1.2 Start-ups

Many start ups fail or find it difficult to develop their own products efficiently this can be effected by multiple factors, they potentially lack experience in small product development or within the target market, the market isnt large enough to sustain the product, they have no users to establish feedback and/or validate ideas. This project will attempt to implement a process model that is effective when working in a start up environment without an established user base.

1.2.1 Difficulties within start-ups

As stated before start up companies often fail within their infancy. They can fail due to failures in execution of sales, marketing and delivery [3] of their products. We will focus on process for product development and delivery that would be effective in a start up environment. The problematic issues of start ups and small companies when it comes to product deliveries include:

- Developers are inexperienced
- Product over fitting (product may be highly customized or targeted towards a certain user)
- Small/non-existent user base

These issues can be mitigated by proper planning and a well versed development process model. We will explore why some of these become issues.

1.2.2 Developers are inexperienced

This may or may not be the case in start ups as many developers come from or have some experience in product delivery management. However when dealing with smaller companies and their own products they often fall into methods and processes that harm the development of the product. This can be easily solved by ensuring the head developer of a product is highly experienced and wont neglect important non-coding issues such as architecture, design, testing and documentation. Having a technical lead whom is experienced helps the potentially less experienced colleagues keep on track when it comes to the non-coding standards.

1.2.3 Product over fitting

When a start up company is developing a small product they have a tendency to over fit their product to a certain user. This is because they have such a small or non existent user base they only have a few sources of feedback and if they over value this feedback they may end up with a product that isnt really suited to other users. To solve this, developers and project managers need to take any feedback that they gather from their users or themselves if they are the only test cases and make sure that it aligns with the goals of the product. Often less, better designed features are more effective than a product full of many half completed features.

1.2.4 Small/non-existent user base

Start up companies often have a limited user base or none at all. The problem with this is it increases the difficulty of gathering requirements from users and the feedback that you do receive is often from the same source. Therefore it can contribute to the over fitting problem. This is the type of environment that this project will be focusing on, the development of a product in an environment with little to no established user base.

1.2.5 Summary

To try and mitigate these issues we needed to implement a development model that helps reduce these potential problems. We needed to ensure that we have a proper authority structure, we have a process for requirement gathering and that we focus on the products goals to try and reduce over fitting to a certain user, including Haunt.

1.3 Problem and Motivation

The problem identified by HAUNT was the lack of a lightweight system that could be easily updated, inform users of their relevant information and present the required information

in a non intrusive and intuitive manner that portrayed the important information without being overbearing. While there are some current solutions for information management systems and learning management systems, neither solve the issue of being too information dense and hard to update appropriately for a small HR team.

The main problem can be split up into two issues, the display, organization and management of the information and the notifications to the user along with displaying user relevant information. These issues is what this product tries to solve.

1.4 Available Solutions

As stated previously, there are two potential types of product that partially solve the issues identified. These are learning management tools (LMS) and knowledge management systems (KMS). However these solutions are only partial solutions to the problem and are aimed at solving different but similar issues. This report explores these current solutions and identifies the components that work well and the areas that they are lacking.

1.5 Solution

The suggested solution is a micro learning management system which address the need for a more lightweight knowledge management system that is easy to update and organize. It solves some of the issues that the current solutions involve such as user targeted information, a more lightweight display of information and an ease of updating information and informing users.

Chapter 2

Background

This chapter explores current solutions that a company could use for managing company information and policies and the informing of staff. The main aspects that we will be exploring are the organization of information, the presentation and navigation of this information for users and the ease of updating this information and informing users of these changes that they may need to be made aware of. We will explore two main solution types, knowledge management systems and learning management systems.

2.1 Knowledge Management Systems

Knowledge management systems main goal is to provide effective and efficient knowledge management. This includes the need to archive information and provide an interface for easily retrieving and display this information. They often take the form of collaboratively driven systems known as Wikis. Wikis provide an environment for conversational knowledge to be created and iteratively improved in an ad-hoc manor and 'combine the best elements of earlier conversational knowledge management technologies, while avoiding many of their disadvantages' [6, 5].

2.1.1 Wikis

A Wiki refers to is a set of linked web pages that are created by incremental changes applied by a group of collaborating users and the software that is used to manage the pages and information. The key characteristics of a wiki are:

- It enables web documents to be authored collectively.
- It uses a simple markup scheme (usually a simplified version of HTML, although HTML is frequently permitted).
- Wiki content is not reviewed by any editor or coordinating body prior to its publication.
- New web pages are created when users create a hyperlink that points nowhere.

From these characteristics you can see that Wikis are a collaborative system that relies on group contribution to remain up to date and the creation of new content. This creates a large knowledge base that can be updated easily and frequently with an active community. However because of the lack of moderation outside of community amendments, some of the published knowledge can be incorrect. This is contrary to what is required within a

company for documents such as policies and other content that requires an approval process.When Wikis have been used as company KMS they often exhibit multiple issues these have been identified as [4]:

- Lack of a clear purpose for the wiki. Wikis are often created by individuals or small team and are then adopted by other teams or individuals.
- Usability. These issues range from poor hosting infrastructure, user knowledge of syntax, duplicated data and validity of data.
- Integrating the wiki into established work practices.
- Role of management. Usage of Wikis is gated by management involvement, encouragement, training and rewarding wiki users all contribute to a more successful wiki.
- Social issues. A critical mass of users is required for a successful wiki.
- Organizational Culture. Users are required to be open to sharing knowledge.

2.2 Learning Management Systems

LMS serve three main purposes, to manage users (both learners and facilitators), track online learning and progress and keep records of activity. LMS are often split into two sub categories, corporate or education, each focusing on a different learning environment. These share many characteristics and there is often overlap between the categories. Common features of a LMS include [7]:

- Online Courses
- Classroom management
- Communication and collaboration tools
- Content management and development tools
- Assessment and testing
- Reporting

Each LMS offers its own suite of functionality, because of this they can fit a multitude of roles in numerous different environments. But the end goal of most them is to create and manage a collaborative online or blended learning environment. One of the most popular LMS's that many others are based on or extend is Moodle.

2.2.1 Moodle

Moodle is an open-source free LMS that is distributed under the GNU public license. Moodle is overall designed to promote community-oriented learning methodology, involving collaborative activities and active participation from its users. This is aimed at creating an online learning community. Moodle includes a multitude of community features such as, forums, blogs, announcements and wikis. It also includes learning management and tracking features such as tests and quizzes, assignments, report, student profiles and activity tracking [7, 8]. However because of the multitude of features, like many LMS it requires training to use and manage effectively. Also for some of it applications it is over engineered and many

of its features go unused or used improperly/poorly. This is a factor with broad LMS systems, because they target a broad range of applications, they lack the clarity of how to best use their features or they some of the features are adapted in incorrect or inefficient ways.

2.3 Summary

There are many KMS and LMS systems that can be used for managing and teaching information for employees. However, neither of them have them as their primary use. KMS are used to store and retrieve this information but have no designated tracking of learning or directed learning environment. While LMS are over engineered towards learning, are focused on a collaborative learning experience and the testing or validation of the users learning rather than a source for easy to navigate information that is targeted towards the user automatically.

Chapter 3

FACT

3.1 Goal

The aim of FACT is to be a solution for provide a system for use within a corporate setting that presents required information and company policies to the employees in small, easy to digest chunks. This is to solve the problem of informing employees of information and being able to check if employees have actually read the information as well as providing a platform for reference material or training. What Haunt found when they were going through the early stages of their company development process was that there was a lack of effective and efficient applications that are effective at informing employees of essential information in a corporate environment, such as health and safety and other company policies. They established that there was a market for a solution that addresses this need for a lightweight application that can be used as an on boarding education tool as well as continued education of company policies. The solution would also notify employees of changes to policies that are relevant to them and allow a manager to ensure that their staff were up to date with company policies. The product should also be scalable to contain enough information that is easily navigable that it could replace or be used in conjunction with other knowledge management systems such as company wikis.

3.2 FACT - The Product

FACT is a micro-learning management tool that offers the ability to manage information on company policies, procedure and other employee on boarding necessities. The final product will also keep track of what users have read, notify them of any changes to the information that needs to be read and allows an administrator to monitor user participation. This information will be organized into decks that would be similar to courses in a more fully fledged learning management system. These decks will be the topics that a user could have assigned to them as required for their role. Decks will be filled with chunks of information called cards and/or facts. These will have a word limit on them to promote smaller chunks of information so they users are more likely to read them.

3.3 Features

3.3.1 Users and Organisations

Organisations will be the account that a user will be registered under. Users under the same organisations will be able to access the same resources such as decks, cards and tags. The

users will be rather simple entities that can be linked to decks through tags and able to be assigned a role for their access rights. A user will need to keep track of the completed cards and when/what version of each card was completed.

3.3.2 Decks, Cards and Tags

Cards would only contain snippets of information that could be easily digestible and mark off as understood. Cards should present the information in a way that the user can easily view all the information and tick them off once they are understood. Cards could also have the ability to show media such as images or video. The editing and restrictions for the cards should promote the idea of this bite sized information. The aim of a card or fact is to convey information that is a self sufficient piece of information, further detailed information could be linked to provide a more thorough overview (See **??**).

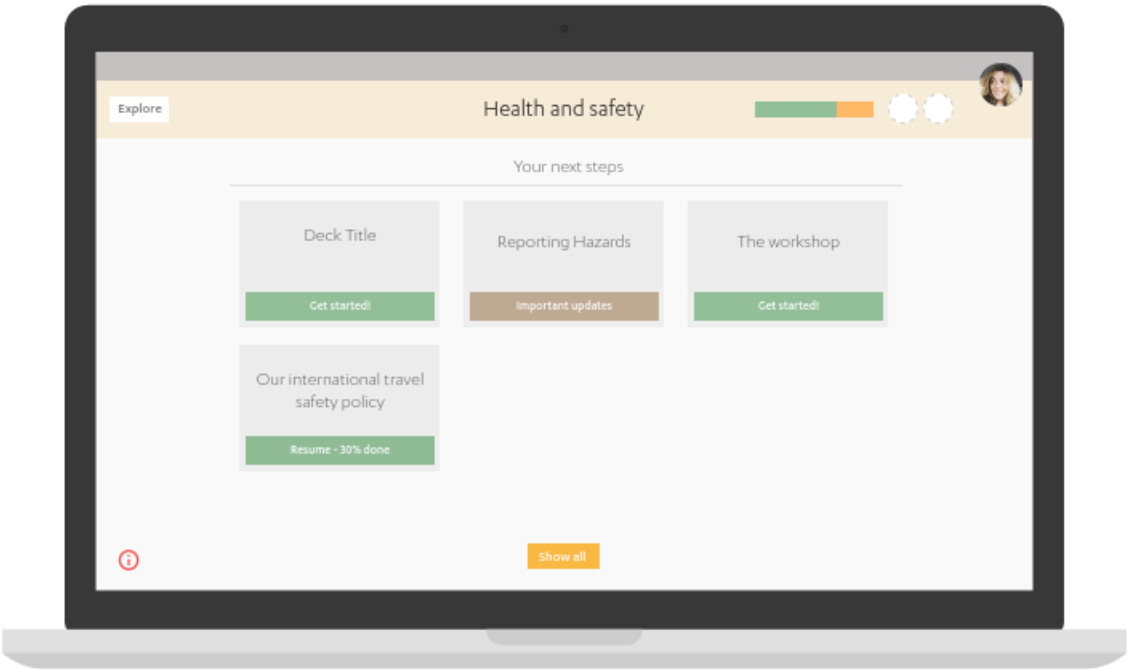


Figure 3.1: The initial pitch concept for the display of decks

The way that the cards will be organized is with decks. Decks will be the underlying structure that represents topics and the cards within them would be related to those topics. For example a deck could be labeled Emergency plan and the cards within could note the steps that need to be taken within different emergencies and other helpful tips and information. Decks would also have the ability to be tagged. Tags would represent the over arching topics that the decks touches. Such as Heath and Safety or PHP. Tags associated with a decks allow for much more interaction with from the users and the decks. It would enable a user to subscribe to certain tags, or a manager to decide what tagged decks are mandatory or recommended for an employee to complete and other similar actions.

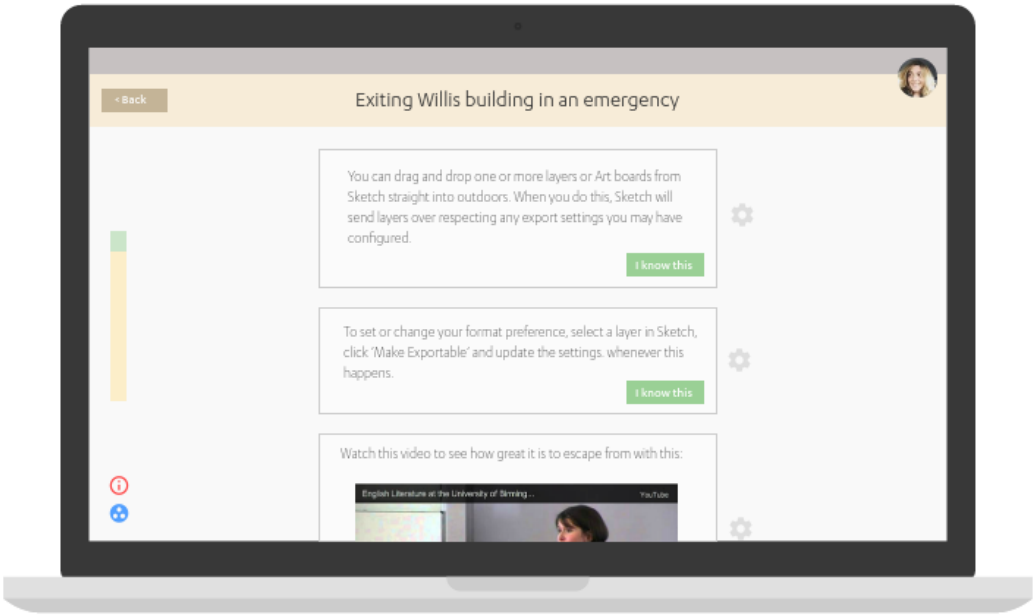


Figure 3.2: The initial pitch concept for the display of cards/facts

Chapter 4

Technology and Engineering Choices

Before starting development on the product I needed to decide with HAUNT the best technologies and architecture to develop our product in. This includes the API architecture, any major JavaScript libraries that would have major implications in the development and any external services that we may want to implement such as external authentication. This chapter will explore some of these options and explain any choices made and challenges that they may hold.

4.1 API Architecture

The first need is to provide an API service to access and modify database as required. The major decision that needed to be made was the choice between RESTful vs SOAP. Both are come with their pros and cons and are useful in different situations.

4.1.1 SOAP

SOAP is a web access protocol has long been the standard protocol when it comes to web services for its versatility. SOAP provides a way for application to communicate over a HTTP network. It is based on XML and is used to make requests and receive responses between an application and a service. ADD SOME STUFF HERE SOAP can quickly become complex and its architecture is more suited to operations and calls to action than simple database operations however it can do both. SOAP also supports multiple expansions to enhance its capabilities the major one being WS-Security which greatly increases security options and allows the use of various security token formats such as SAML (Security Assertion Markup Language). Because of this SOAP excels in many different environments but requires more customization and research into the required extensions. SOAP also has built in retry logic and error handling.

4.1.2 REST - Representational State Transfer

RESTful web services are a way of providing an application to to communicate with a service via a set of predefined stateless operations. RESTful services in general boast less bandwidth usage and faster response times [9]

TODO

4.2 React - UI JavaScript Library

React is a JavaScript library that is used to create interactive user interfaces easily. It was originally developed by engineers within Facebook when working on their own complex user interfaces. React brings a new concepts to web development, it shifts the generally accepted workflow of web development. React solves the problem of large scale user interfaces with data changes consistently [2].

React allows you to 'design simple view for each state in your application' and it will efficiently handle rendering the right components and automatically re-render the correct information and components when data changes[1]. Traditionally the major problem when designing and developing a user interface is keeping it in sync with the business logic and state of the application and data [10].

TODO EXPANSION REQUIRED

4.3 Summary

Chapter 5

Agile and Lean Methodologies

Software start up companies are required to be efficient with their development time and product analysis. There are two main methodologies that we will explore as possible solutions to implement during this development analysis at Haunt, Agile methodology and Lean methodology. Along with the overall methodology there are many tools and processes that you can utilize to implement the principles of your methodologies.

5.1 Agile Methodology

The Agile methodology was first created in February 2001 and is defined by the Agile Manifesto. It was designed specifically with software engineering in mind with the focus of the principles being customer satisfaction through early and continuous delivery of valuable software [1]. Agile methodology was developed for software development as the previous more rigid processes had expensive iteration cycles and did not fit the quickly evolving ideas and environment of software product development.

The principles in the Agile development are designed to produce a potentially shippable product after each iteration, around 2-4 weeks depending on the implementation of which process model. The principles from the Agile Manifesto:

- Satisfy the customer through early and continuous delivery of valuable software.
- Welcome changes in requirements. Agile processes should be flexible.
- Daily cooperation between business people and developers
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of success
- Promote a sustainable development process and environment.
- Continuous attention to technical excellence and good design enhances agility
- Simplicity
- Self-organizing teams
- Regular reflection on how to become more effective and tune process appropriately.

Agile methodology is great for flexible development of the product that can adapt to changing requirements and market environment.

5.2 Lean Methodology

Lean methodology is taken from the lean manufacturing process principles and translated into a software development environment. It originated in Toyota manufacturing around 1950 as Just-In-Time [4]. The main goals of lean methodology is to eliminate waste and work smarter not harder. Waste within a manufacturing context is easier to identify. Lean manufacturing identified 7 main wastes. In a software development context these translated into:

Toyota Manufacturing Wastes	Lean Software Development Wastes
Inventory	Partially Done Work
Extra Processing	Relearning
Overproduction	Extra Features
Transportation of Goods	Handoffs
Waiting	Delays
Motion (of people)	Task Switching
Defects	Defects

Table 5.1: Software Development Waste[]

To first eliminate these wastes you need to first identify some of their causes and follow a process that tries to minimise or eliminate them. These causes identified in Software Development Waste are very relevant in a startup company. Building the wrong feature or product, creating features that no one needs or products that have little market are obviously not good investments of time and lead to waste (Extra Features). Mismanagement of the backlog, focusing on features or products that offer worse value to the customer is obviously a less efficient use of development time and are often resulted in unfinished work when developers are needed to finish more important features (Partially done work). Rework, if your software has a lot of defects/bugs or is not very robust or secure you will need to rework and refactor parts of it resulting in wasted time (Defects). Delays, in Lean Software Development delays refer to waiting for people to be available to provide required information. Relearning refers to rediscovering something we knew [5]. And is either caused by forgotten decisions that have already been made or failing to communicate information effectively and therefor someone else rediscovering something that we already know.

5.3 General Lean Process

The generic process for Lean Software Development in a startup follows a Learn, Build, Measure process. Where you first gather information, gather required features and ideas, build a functional product that can be tested, test said product and gather data and then learn from that data and start the process again. Lean methodology and Agile methodology are closely related and Agile could be considered a subset of lean methodology that focuses on wasted time only. The key differences between agile and lean methodologies are that Lean focuses on reducing many types of waste to produce the product with the most value to the customer with little waste while agile more focused on constant integration and feedback to produce frequent iterations of the product that are always improving.

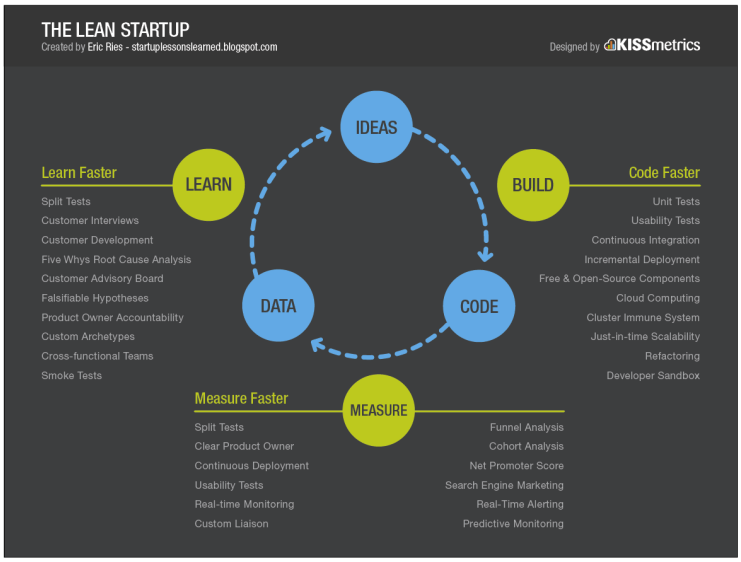


Figure 5.1: Lean process model

5.4 Process Models

Both methodologies can utilize multiple process models that fit or can be adapted to adhere to their principles. Very common models are SCRUM for agile and KANBAN for lean.

5.4.1 SCRUM

SCRUM is an agile process model that focuses on having a potentially shippable product after each iteration called sprints. Each sprint normally lasts for around two weeks. The features or tasks that are worked on are taken from the backlog. The backlog is populated by user stories that have been collected from the client or product owner. They are use cases for potential features for the product

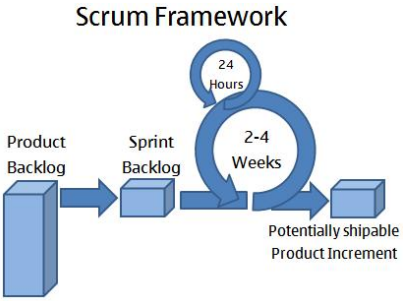


Figure 5.2: SCRUM process model

The SCRUM workflow begins with a planning stage for the sprint. The goal of the planning stage to to select and delegate the user stories to be completed this sprint. During this planning stage you select and evaluate the difficulty of user stories to be completed during the coming sprint. Then the daily scrum is used to say what each team member will work in the next 24 hours to achieve this goal. This is repeated until the end of the sprint. After a sprint you would have a potentially shippable product and there is a review and reflection process as well.

5.4.2 Extreme Programming: XP

Extreme programming (XP) was one of the first agile methodologies. It was designed to solve the problems of heavier, more formal methodologies and be more flexible and efficient as explained by Kent Beck, XP is a lightweight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly changing requirements. [2]. XP is similar to scrum in the fact that it emphasizes frequent releases and shorter development cycles. Along with advocating pair programming, code review and unit testing on all code. The basic principles of XP are [2]:

- Rapid feedback
- Assume simplicity
- Incremental change
- Embracing change
- Quality work

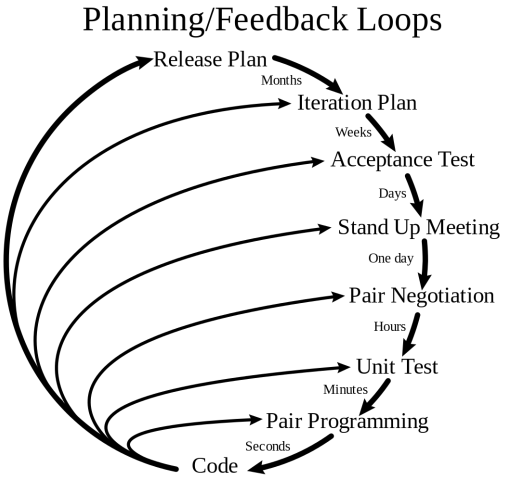


Figure 5.3: XP process model

XP is highly flexible and can easily manage in an environment of rapidly changing requirements. It is simple and easily understood and can be implemented with any team that is familiar with agile methods and manifesto. It relies heavily on pair programming and automated unit testing.

Chapter 6

Implementation

For the implementation of this project we decided to implement a agile methodology approach similar to other projects run at Haunt. The main process model that we will be taking much of the development process from are aspects of SCRUM. The development was split into multiple sprints, with each containing a requirements gathering stage, development and reflection stage. Each sprint will start with a client meeting with Haunt where we will populate the backlog and develop user stories as requirement gathering. Later on in the development we planned to include meetings with potential users and experts for feedback on the implemented solutions and gather more insight for more efficient user stories. However due to unforeseen circumstances we were only able to gather expert feedback within the final sprint. One of the key features of lean and agile methodology is saving on waste. To try and implement this we will focus on building the core product/prototype first and then gain feedback on the product via user testing to gather more specific features and requirements for the product.

6.1 Initial Product

At the start of the project Haunt had a preliminary proof of concept that needed to be further developed. This initial product contained an API that could add cards, decks and users. The front end consisted a simple view that could display the contents of the decks and cards related to the user. The API was developed in Ruby and followed the RESTful architecture, with the front end developed in JavaScript with Redux js for front end state control. All authorization was done through Auth0 as a temporary solution.

6.2 Pre-Development

Before initial development started we first needed to establish what aspect of the product we would be focusing on. Initially the plan was to develop a potential dashboard for the product. However because development on the main product prototype was no longer in progress via Haunt, this project changed focus to general development of the product. During the first meetings with the product owner, Rob McGrail, Technical Director at Haunt, we established that some of the underlying technologies on both the front end and api side were either excessive or not long term solutions and needed to be removed. These were identified as state control via redux and the authentication system through Auth0, the first sprint was focused on removing these required technologies to reduce future development time. The later sprints would be focused on refining, adding and fleshing out features in the product.

6.3 Sprint 1

As stated above the first sprint was focused on reducing the overhead of future development by removing unnecessary and excessive factors of the initial proof of concept, namely the authorization system Auth0 and the front end state management control redux js. The product owner and I deemed this an important step to reduce waste at later stages of development and not build up technical overhead.

6.3.1 Auth0

Auth0 is a third party single sign on and token based authentication system that the initial proof of concept was using for user authentication. While this was easy to setup and manage, it was not a suitable long term solution due to extra costs incurred through licensing and extra latency of relying on a third party service instead of having the authentication handled by the same service as the Restful API. Auth0 would potentially be able to provide better security of data and more complex After some consideration we decided it was best to replace the authentication with a JSON web token (JWT) system as it is industry standard for 'URL-safe means of representing claims to be transferred between two parties'[3].

6.3.2 JWT Tokens

JSON web tokens consist of an encrypted token that contains a payload consisting of a JSON object and is encrypted via a JSON web signature. Using these JWT tokens we are able to securely authenticate users and produce a token for them that tells the API their user role, organisation, id and creation date. This allows them to stay logged in securely and helps govern access to the API based on their role.

6.3.3 Redux JS

Redux is a state container for JavaScript applications, it was used to keep the current state of the front end under control to help lower load times and preserve data and scope. Redux is a great tool for managing your app state in a way that allows states to persist through the same session or even through sessions if saved locally. However it comes with some trade offs and requires you to structure your apps data in certain ways:

- State needs to be described as plain JSON objects and arrays.
- Describe changes in the system as plain objects
- Any logic for handling changes to the state needs to be as pure functions

These requirements for how you structure logic, state and changes within the system make sense and are often good practices to follow if you wish for your code to be decouple what happened, for example a button press or data finish loading with how this changes things, for example when the data is loaded change the values on screen. However because of the relatively simple states required for this web application and no foreseeable need for a persistent state. It was decided that maintaining the state store through Redux was not required and the app.

So for this sprint the main focus was removing these unnecessary libraries and replace them with current solutions. The Auth0 authentication service was replaced with our own JWT focused authentication using and redux state management was replaced with logic within the components.

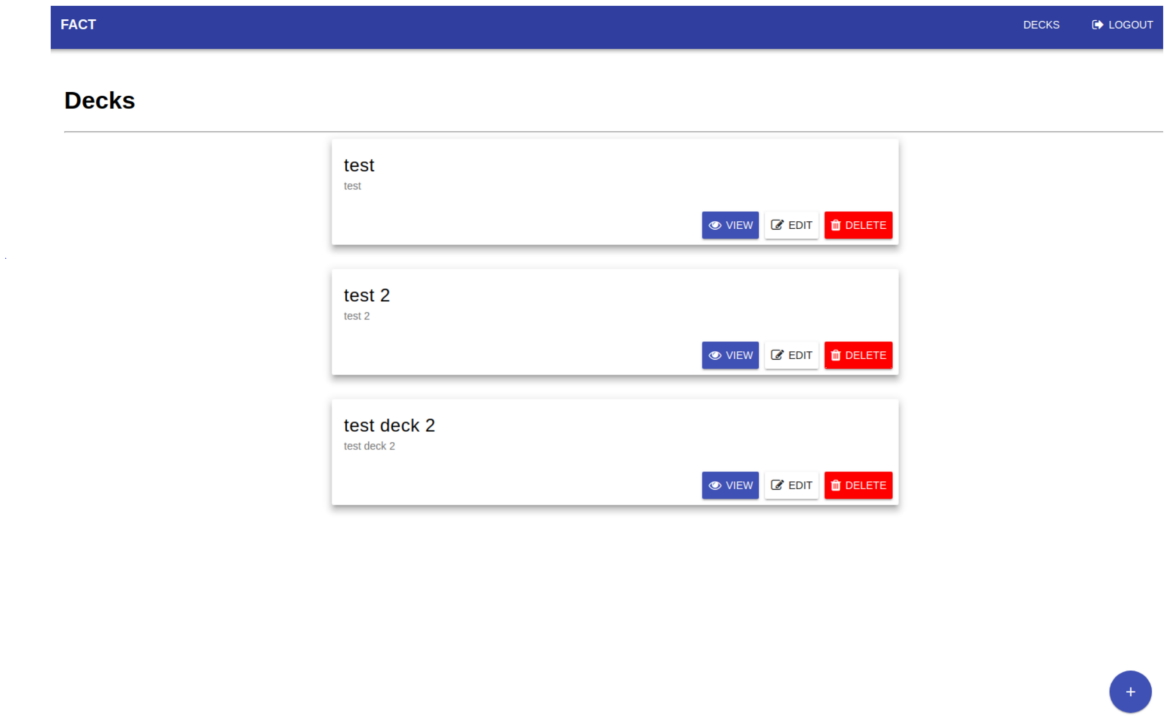


Figure 6.1: Display of a fact deck after development in sprint 1

6.3.4 Summary

With this sprint focused on removing some of the system that were no longer required a lot of this sprint went into refactoring the code base and implementing the new authentication system. The refactoring of removing redux from the system included moving the action logic for components into the components themselves. This reduced the indirection within the code but more tightly couples the components with the logic and actions of the system. This means that it is harder to implement another component with very similar features that should act the same way but with a different look or display. This was considered a reasonable trade off as it simplifies the workflow of each action and makes the code easier to follow, because the products logic flow should be reasonably simple with no overly complex actions or components.

6.4 Sprint 2

This sprint was targeted towards getting some of the key features functional and in a state where the app could be used functionally. The initial proof of concept had some of the features like deck and card creation, user and organisation creation and the ability to view decks/cards linked to your organisation. The next major step was getting editing for all of the assets and tidying up the interfaces for managing the decks and cards.

6.4.1 Improving Workflow of Cards and Decks

One of the major problems identified with the work flow of the proof of concept was the disconnect between the list view and the creation of cards and decks. Initially creating a new card would bring up a separate screen with a separate url. This disconnects the content

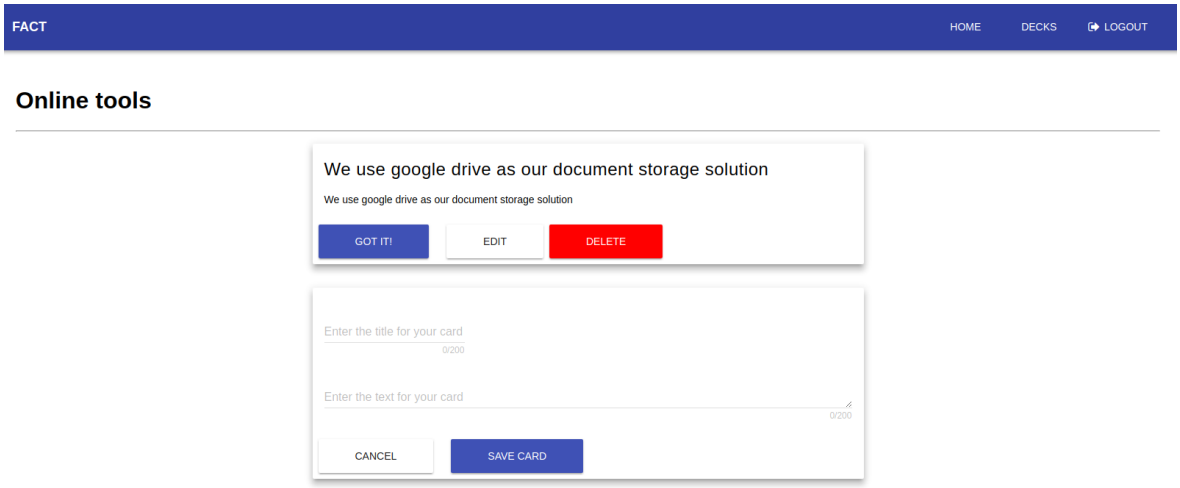


Figure 6.2: Card creation after sprint 2

from the context of the deck or organisation. To improve the user experience when creating cards and decks, I implemented an inline feature for adding cards. To do this I changed the creation form to be displayed as a card the same as the current cards within the deck. This made it so when you start adding a card it will display it in the list and allow you to fill out the details within the card without changing changing page or view.

This change streamlined the user experience and made it more clear what the creation of cards and deck was doing and the general structure of the information. This in display creation and editing can be seen in many other web applications that use a card based display system such as Trello and Basecamp (see appendix: A).

6.4.2 Cards and Deck Editing

During this sprint I added editing to the cards for both decks and cards this used the same system as the card creation just with different actions. This allowed me to reuse the same react component with different actions passed to it. This highlights the uses for react and the flexibility it brings when it comes to code reuse.

6.4.3 Summary

At the end of this sprint we went through the basic features that we needed to implement for a minimum viable product and identified some of the major ones that the product was still missing. It was also discussed about changing the component library we were currently using as it had limited design space and the overall default styling was not close enough to the final goal for the minimum viable product.

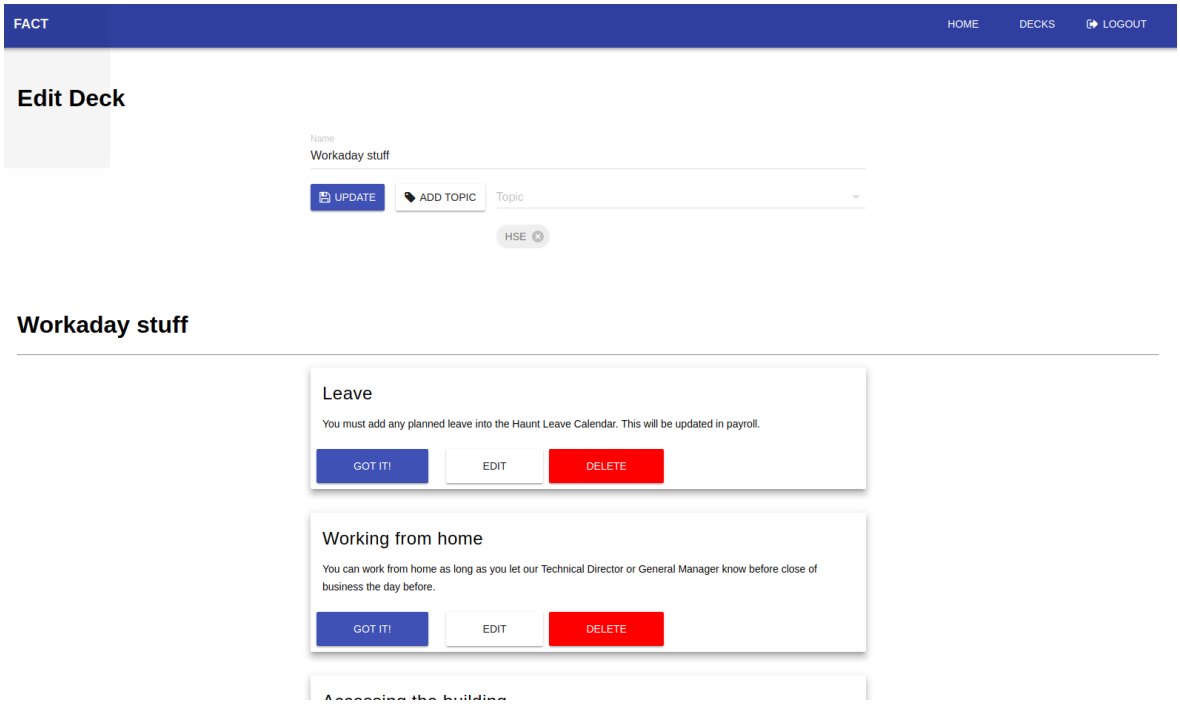


Figure 6.3: The inclusion of tags in sprint 3

6.5 Sprint 3

- Added tags to decks
- Finished api database shema allowing for more info in cards, including title and limit characters and card completions

6.6 Sprint 4

This sprint was the final sprint

- Added
- completion bar
- new layout
- create buttons
- drop down actions
- links in cards

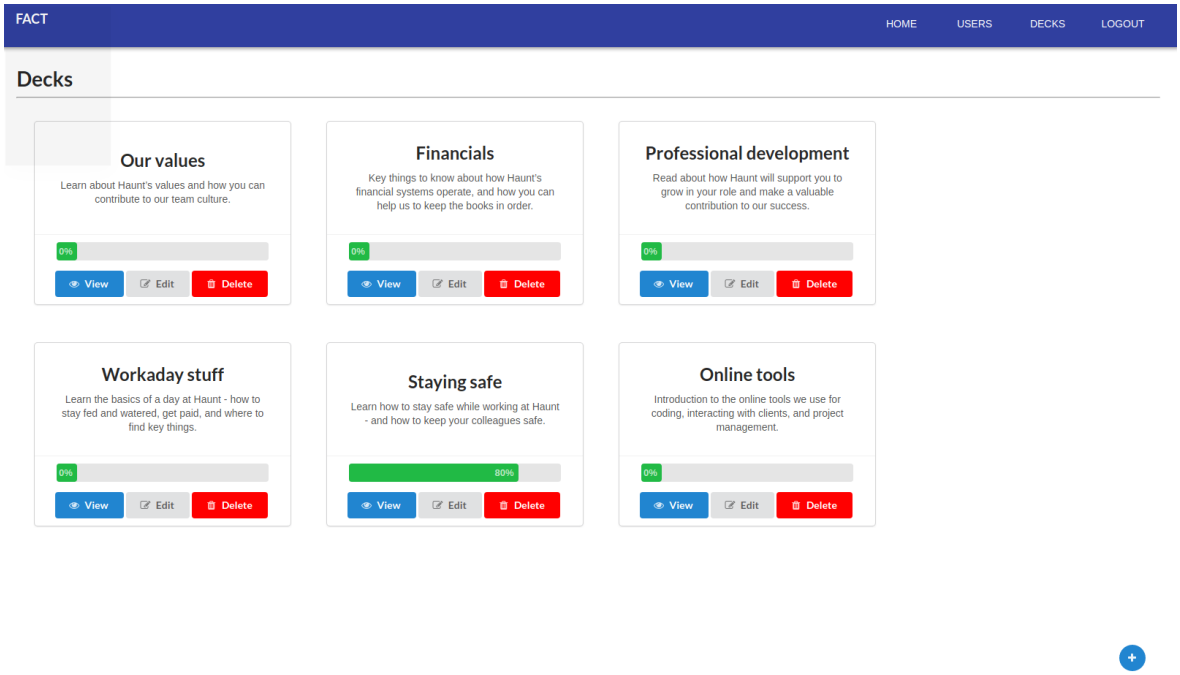


Figure 6.4: Deck display after final sprint

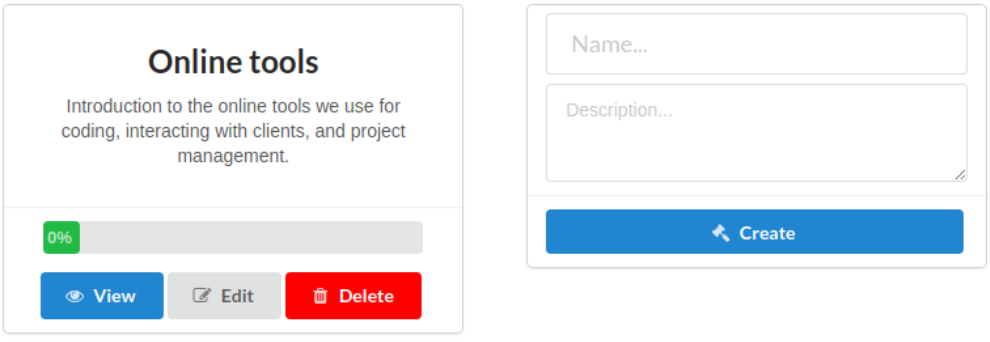


Figure 6.5: Deck creation interface after final sprint

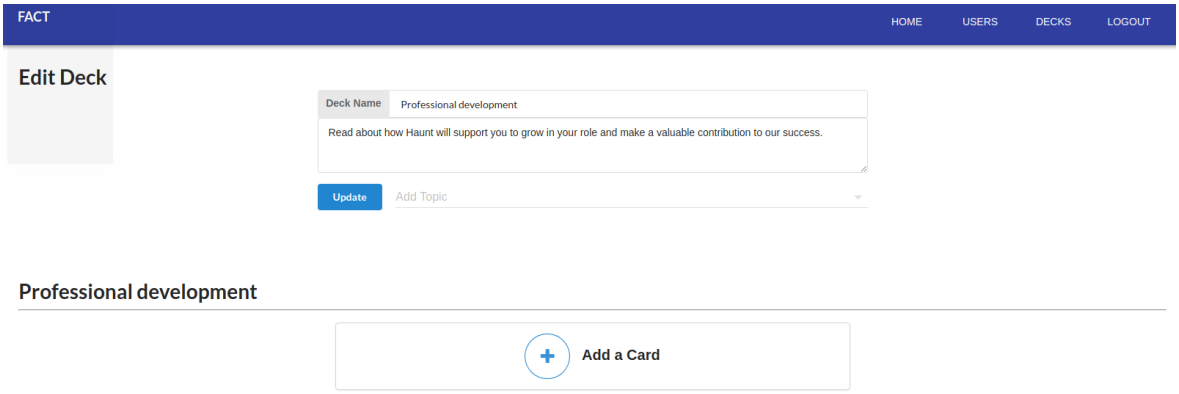


Figure 6.6: Deck edit interface after final sprint

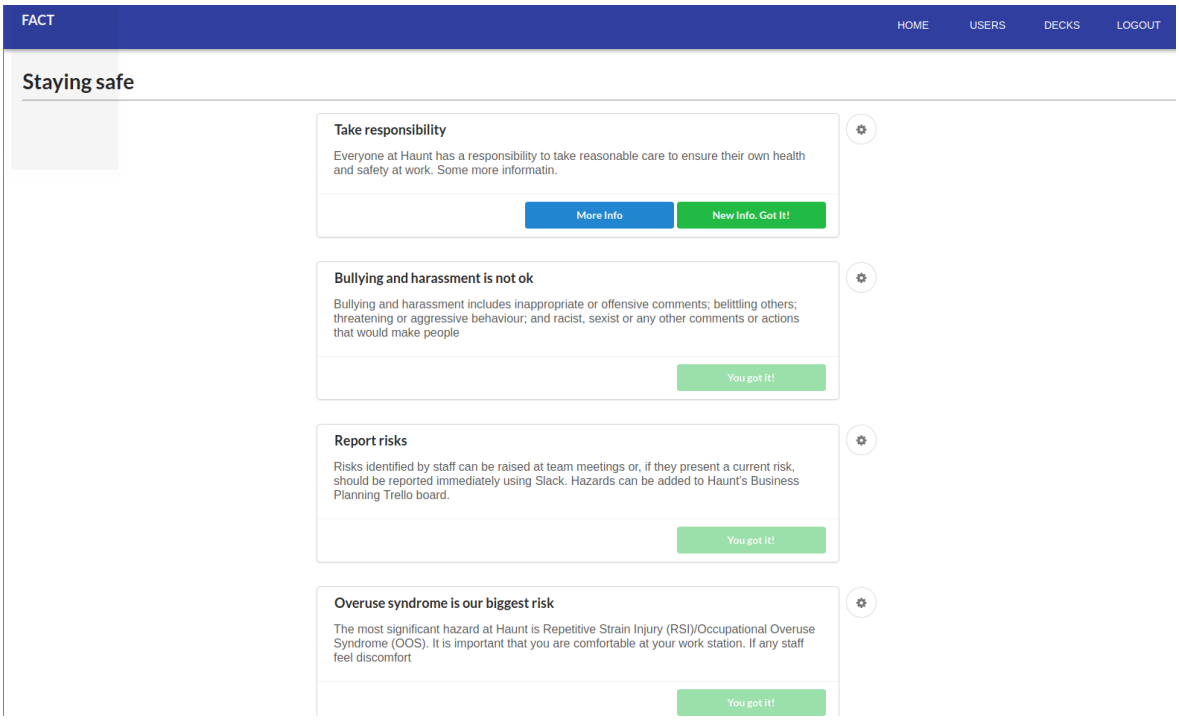


Figure 6.7: Cards display after final sprint after final sprint

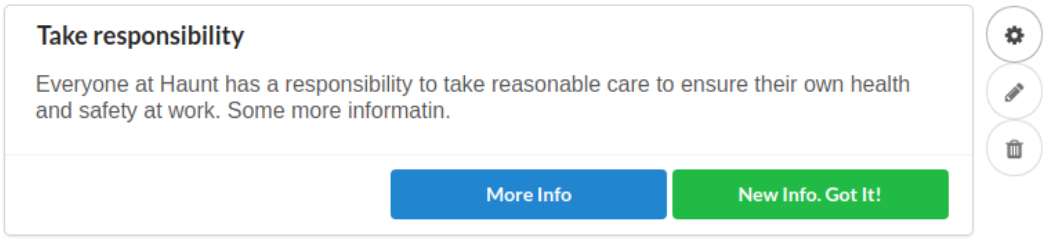


Figure 6.8: Card action dropdown after final sprint

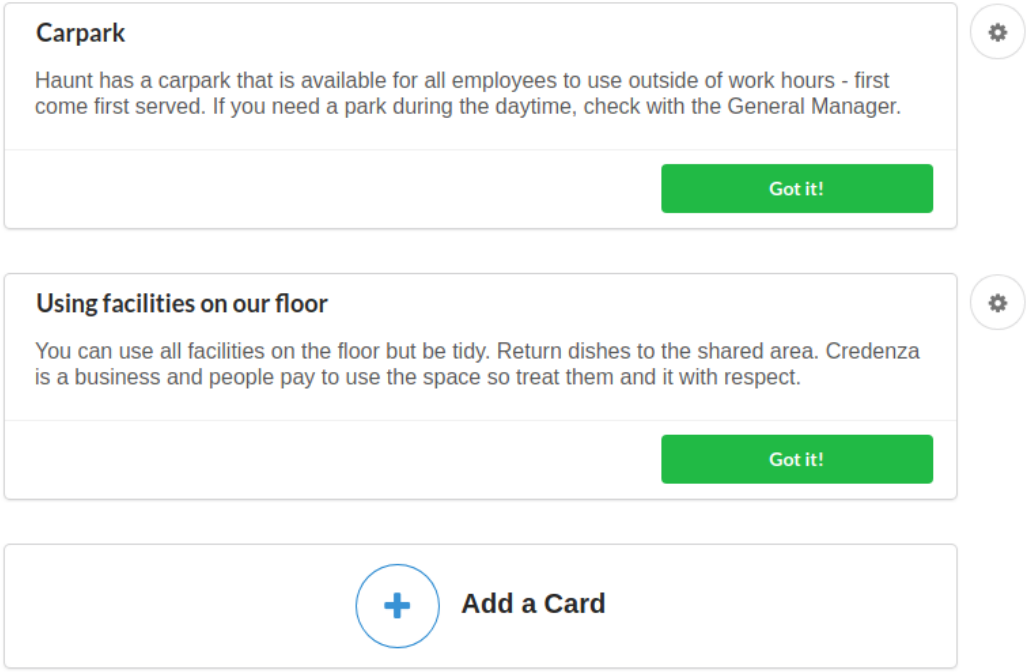


Figure 6.9: Card adding interface after final sprint

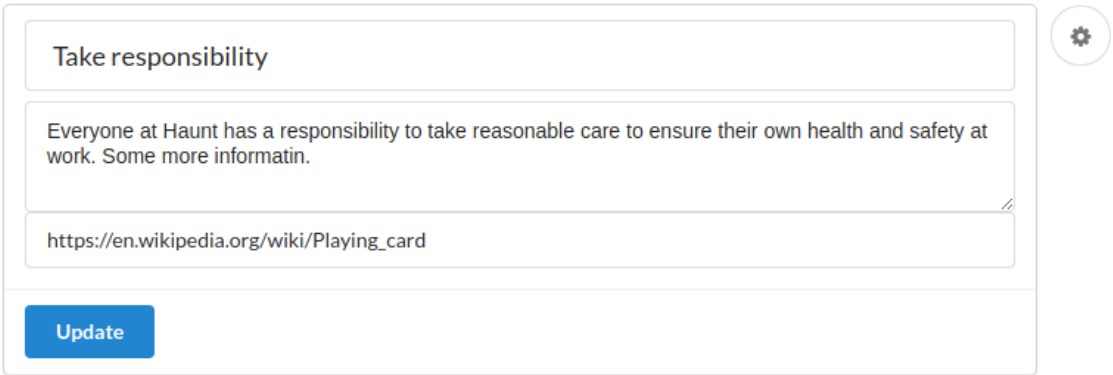


Figure 6.10: Card adding interface after final sprint

Chapter 7

Industry and Expert Feedback

Chapter 8

Comparrison to current solutions

Chapter 9

Conclusions

The conclusions are presented in this Chapter.

Bibliography

[1] React: A javascript library for building user interfaces.

[2] GACKENHEIMER, C. *Introduction to React*. Apress, 2015.

[3] JONES, M., BRADLEY, J., AND SAKIMURA, N. Rfc 7519: Json web token (jwt). *IETF, May* (2015).

[4] KINITI, S., AND STANDING, C. Wikis as knowledge management systems: issues and challenges. *Journal of Systems and Information Technology 15*, 2 (2013), 189–201.

[5] LEUF, B., AND CUNNINGHAM, W. The wiki way: collaboration and sharing on the internet.

[6] MAIER, R., AND HÄDRICH, T. Knowledge management systems., 2011.

[7] MCINTOSH, D. Vendors of learning management and e-learning products. *Learning Management Vendors 2014* (2014), 88–96.

[8] MUHSEN, Z. F., MAAITA, A., ODAH, A., AND NSOUR, A. Moodle and e-learning tools. *International Journal of Modern Education and Computer Science 5*, 6 (2013), 1.

[9] MUMBAIKAR, S., PADIYA, P., ET AL. Web services based on soap and rest principles. *International Journal of Scientific and Research Publications 3*, 5 (2013), 1–4.

[10] STAFF, C. React: Facebook's functional turn on writing javascript. *Communications of the ACM 59*, 12 (2016), 56–62.

Appendices

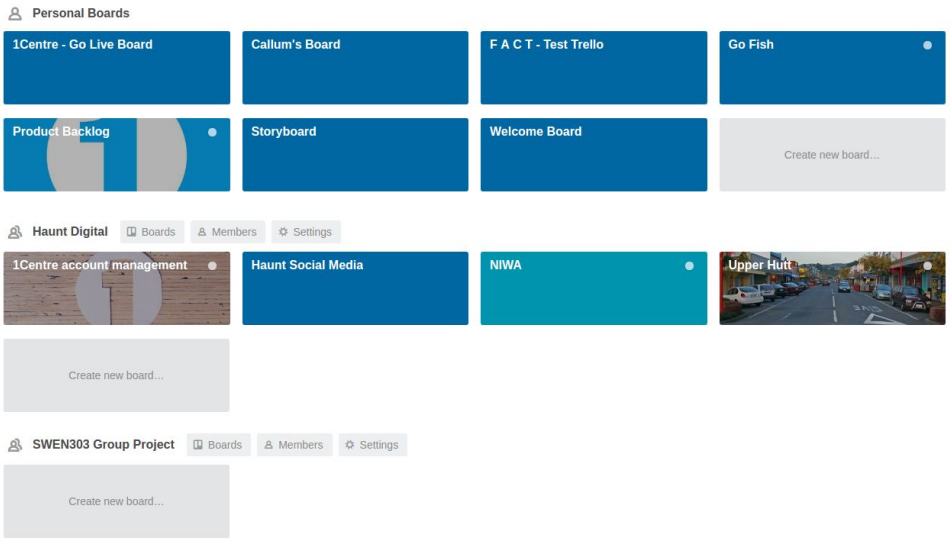
Appendix A

Card Research

F A C T - Card analysis and research

Case studies

Trello
Board cards only contain minimal information in a discrete but easily identifiable way with a background picture/color and title. And are organised by group.



Pros

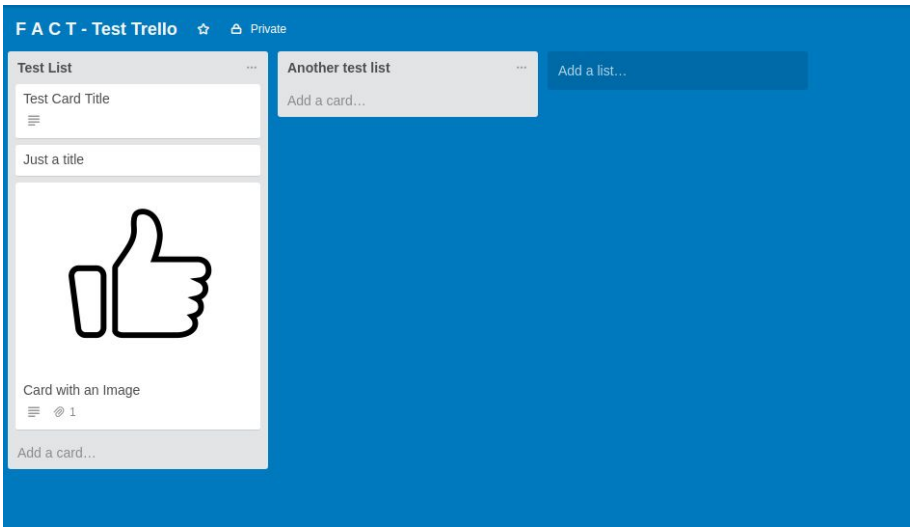
- Allows for easy to identify the board that you are looking for.
- Cards are not cluttered
- Contains new activity marker
- Favourite feature allows you to mark the boards you know you will be using alot

Cons

- Very little information can be seen from this screen, extra clicks required for more information like last edited card numbers, members etc.
- No sort feature

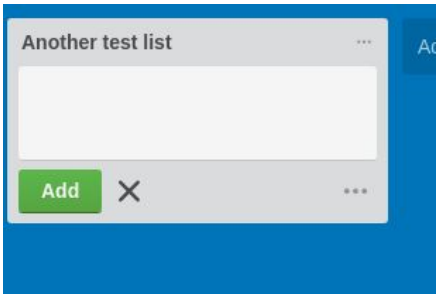
Cards

The idea of cards is to convey only the important information and do so in a concise and non overcrowded way.



Pros

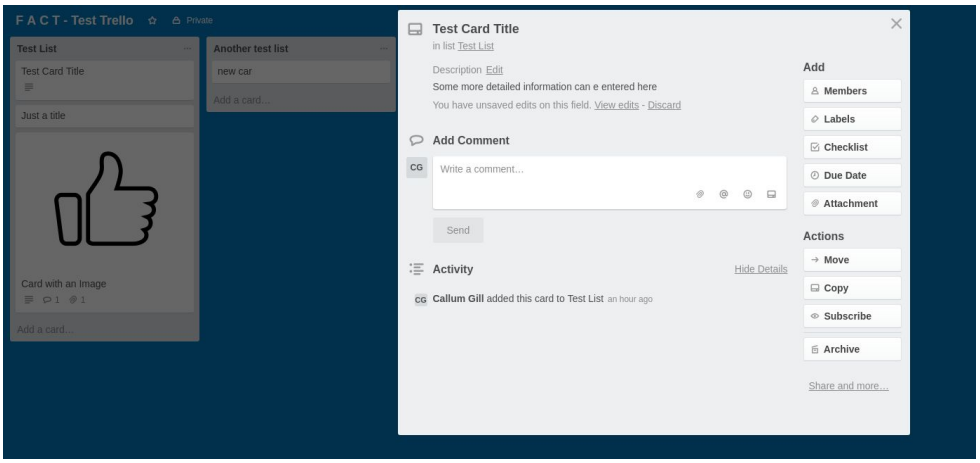
- Cards only display information if they contain that component (eg. 1 attachment, can see if more detailed description inside)
- Images can be easily used to identify cards, however images are automatically displayed (they don't needs to be set as header image)
- Add card clear and non-intrusive to quickly add cards, automatically starts new card after one is added that can easily be cancelled by just clicking somewhere else or the cross.
- Add card data is saved to be easily resumed if clicked elsewhere



- Only the title is asked for on the cards initially so can add lots of preliminary cards fast that can easily be edited later.
- Intuitive controls such as drag and drop/click to open etc

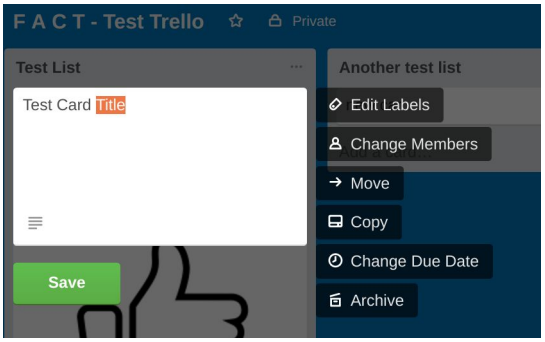
Expanded View

- The expanded view looks nice with text only cards and has an easy to follow comments section, however when multiple attachments are added the view quickly become cluttered
- Contains detailed activity that has happened on card
- More detailed actions that can be taken



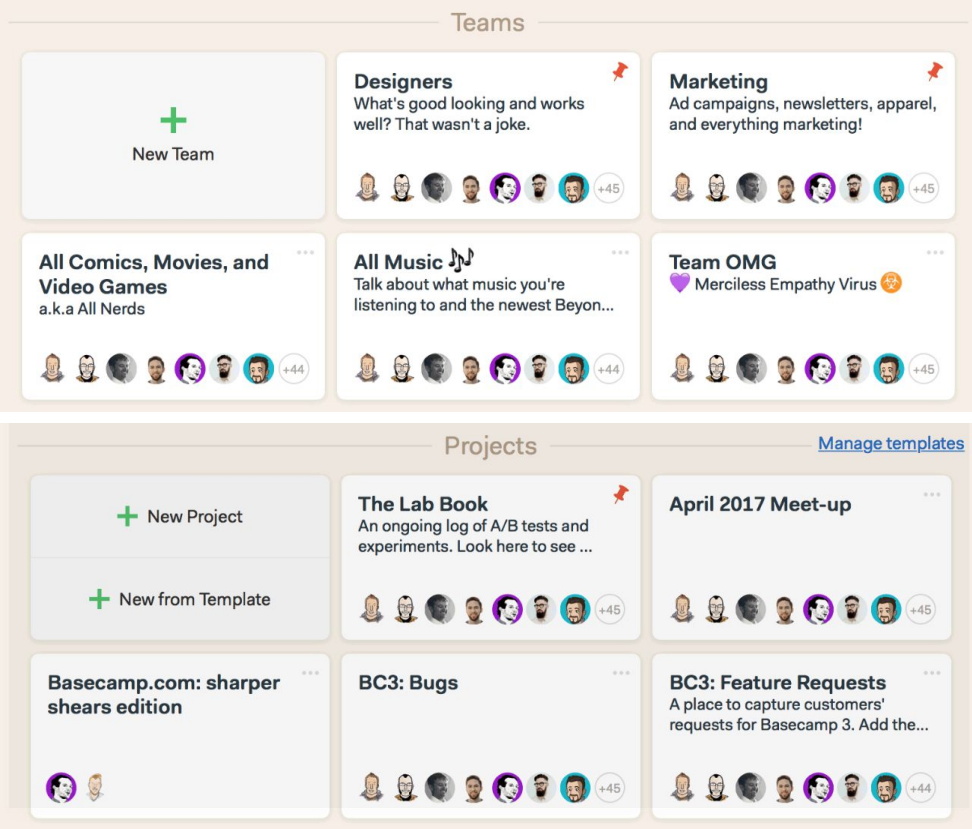
Mini expanded/edit view

- Quickly edit basic components of cards
- Highlight which card you are using by darkening surrounding
- Simple actions only/leave more complicated to the larger expanded screen

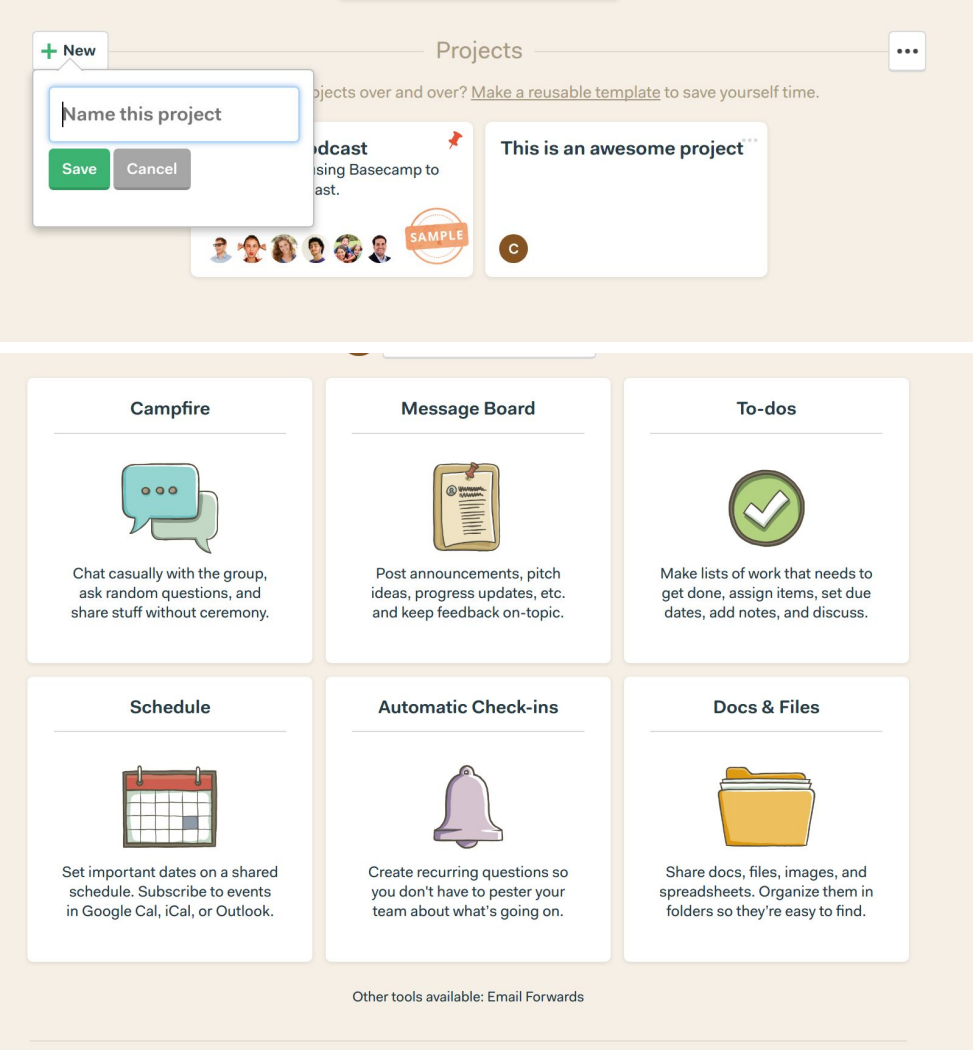


Basecamp

- Basecamp card use a lot more small icons/pictures to let you know stuff about the card. Eg the profiles associated with the card.
- Large clear buttons for creating new cards
- Title + short description
- The more iconographic style makes cards more distinctive



- They have moved away from the whole card buttons for creating new content and added them to the header of each section
- Adding a card only requires the title which makes sense for projects and teams where a description isn't always required
- Using cards for the 'dashboard' of each project is a clean way to present a menu



Things to take away

- In context editing is very smooth and intuitive for card layouts, ie you just have to click on a text box and it will highlight it for editing seamlessly
- When creating content make minimal information required
- Present important summary information on higher level cards concisely without over crowding them (two trains of thought on this one, more information vs cleaner design with less clutter)