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

The Author

Supervisor: NOT STATED

Submitted in partial fulfilment of the requirements for
Master of Computer Science.

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 implimentation of the product, engineering choices involved and the evaluation of the product with expert feedback.

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 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.

This problem can be split up into two main issues, the display and organization of the information and the notifications to the user along with displaying user relevant information.

1.3 Available Solutions

There are two potential types of product that partially solve the issues identified. These are learning management tools (LMS) and wikis. However these solutions are only partial solutions to the problem and are aimed at solving different but similar issues.

Chapter 2

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.

2.1 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.

2.1.1 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.

2.1.2 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 [4]

TODO

2.2 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 [5].

2.3 Summary

Chapter 3

The Solution

3.1 Goal

The aim of FACT is to be a solution for smaller to medium sized companies to provide required information 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 with when they were going through the early stages of their company development process was informing employees of essential information such as health and safety and other 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 things 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 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 subscribeto certain topics, or a manager to decide what decks are mandatory or recommended for an employee to complete and other similar actions.

Chapter 4

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.

4.1 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.

4.2 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.

4.3 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.

4.3.1 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].

4.3.2 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.

4.3.3 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.

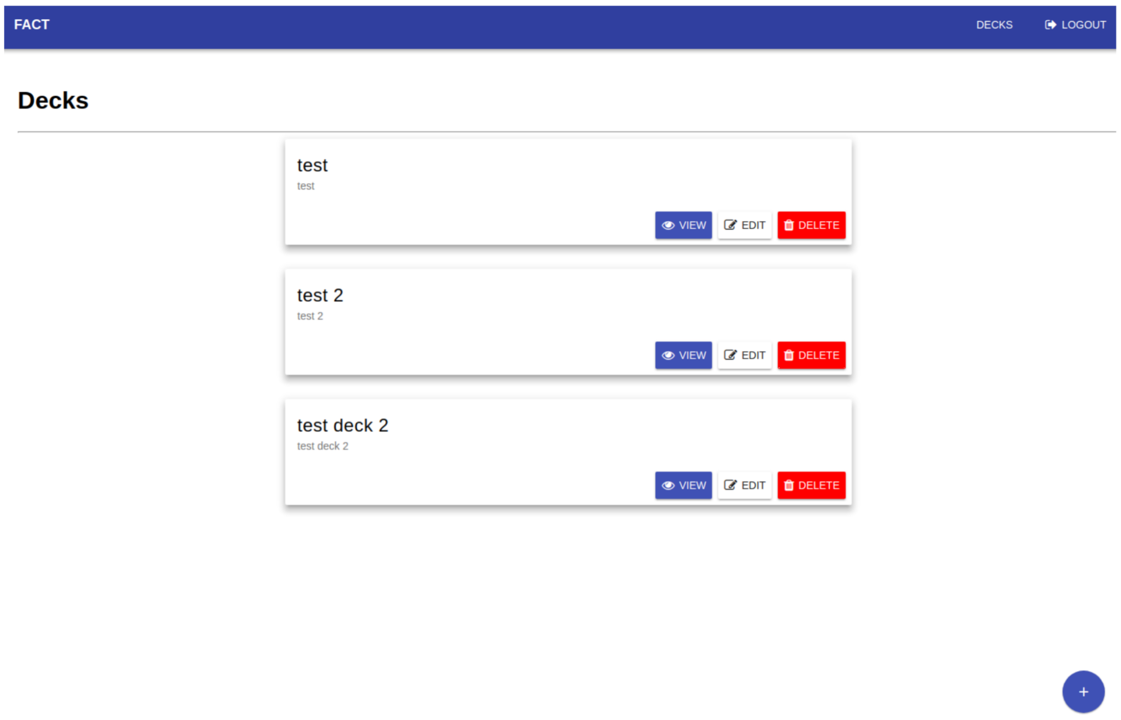

Figure 4.1: Display of a fact deck after development in sprint 1

4.3.4 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.

4.4 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.

Bibliography

[1] React.

[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] 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.

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