React Web App – Decisioning Dashboard

Benjamin Roberts

Level 4 Software Development

Contents

[Table of Figures 3](#_Toc171326741)

[Glossary of Acronyms 4](#_Toc171326742)

[Acronyms 4](#_Toc171326743)

[Terminology 4](#_Toc171326744)

[Introduction & Requirements 5](#_Toc171326745)

[Project Description 5](#_Toc171326746)

[Acceptance Criteria 5](#_Toc171326747)

[Summary of Stakeholders Involved 6](#_Toc171326748)

[Design 7](#_Toc171326749)

[Initial Draft for page template 7](#_Toc171326750)

[Wireframe 8](#_Toc171326751)

[Implementation / Development 9](#_Toc171326752)

[Initial setup of the React application 9](#_Toc171326753)

[Setting up Dependencies 9](#_Toc171326754)

[NPM Audit – Addressing Dependency vulnerabilities 10](#_Toc171326755)

[Setup a Node.JS backend for SQL Access 12](#_Toc171326756)

[Debugging Compile Errors (MSSQL) 12](#_Toc171326757)

[Backend Set up in Node.js 13](#_Toc171326758)

[Personal Data (PII) Protection 14](#_Toc171326759)

[Deployment 15](#_Toc171326760)

[Hosting in Azure Static Web App 15](#_Toc171326761)

[Confirming access 18](#_Toc171326762)

[Testing 19](#_Toc171326763)

[UAT – Gathering user feedback 19](#_Toc171326764)

[Review with Employer 20](#_Toc171326765)

[Conclusions 21](#_Toc171326766)

[Project Outcome 21](#_Toc171326767)

[Future Additions 21](#_Toc171326768)

[References 22](#_Toc171326769)

[Appendix 23](#_Toc171326770)

[Initial Backend attempt, with React & Node.JS using concurrently 23](#_Toc171326771)

# Table of Figures

[Figure 1- Initial draft Wireframe drawn on Whiteboard 7](#_Toc171327020)

[Figure 2 - Figma Wireframe of a Home Page for the React app 8](#_Toc171327021)

[Figure 3- Output of npm audit command, showing a list of known vulnerabilities which could compromise the application 10](#_Toc171327022)

[Figure 4 - Output of npm audit fix command, showing that breaking changes would be needed to fix the identified vulnerabilities 11](#_Toc171327023)

[Figure 5- Importing "mssql" into a React Component called "Home.jsx" 12](#_Toc171327024)

[Figure 6- Compile error from importing the "mssql" library 12](#_Toc171327025)

[Figure 7 - Initialising an Express app based on a templete 13](#_Toc171327026)

[Figure 8 - Install http-errors dependency 13](#_Toc171327027)

[Figure 9 - Setting up an Azure Static Web App to host the React application 15](#_Toc171327028)

[Figure 10 - The React App home page as deployed to the Azure Static Web App 17](#_Toc171327029)

[Figure 11 - Confirmation a user can access the deployed React App 18](#_Toc171327030)

# Glossary of Acronyms

Below is a series of Acronyms and terminology definitions that will be useful for understanding concepts throughout the project

## Acronyms

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Name** | **Meaning** |
| MI | Management Information |  |
| UAT | User Acceptance Testing |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Terminology

|  |  |
| --- | --- |
| **Term** | **Description** |
| Raw Data |  |
| Scorecard |  |
| Summary Data |  |
| CAIS |  |
|  |  |
|  |  |

# Introduction & Requirements

With my team being responsible for developing and maintaining our Credit Decisioning system, we require various kinds of Management Information (MI) and views of the data used by the platform (both in the aggregate and within individual applications). This is also important for monitoring the performance of the platform.

Today, we use various tools to monitor and report on the performance of the system e.g.:

* Ad-Hoc analysis done via querying a SQL Database
* Summarised Microsoft Excel Reports
* Microsoft Power BI reports
* Manually extracting and reviewing raw data (typically via SQL)

All these tools are useful for their specific purposes but what they don’t do is give us a presentable view of the low-level data used in making a decision on an application. We could license other software for this purpose e.g. the Credit Bureau providers Experian, TransUnion and Equifax all provide software to present the data they hold on an applicant. These however require software licenses to be purchased. I believe we could use the React Framework to build a similar tool that would be more bespoke to our business needs.

## Project Description

Using the React Javascript Framework, design and develop a simple Web Application to:

1. Leverage the Java Decisioning Application I developed in a previous project & it’s logged output (Link: [GitHub - Java Credit Decisioning Program](https://github.com/btr6566/qam1_java_decisioning_app))
2. Display a list of recent applications made to the Credit Decisioning system
3. Allow a user to click on a specific application to view the data used for it in more detail
4. Provide reference material for the data definitions of the data used (i.e. a Data Dictionary function)
5. Provide some high level dashboards on applications volumes

## Acceptance Criteria

1. Wireframe of the design of the website is provided
2. Final product aligns to the wireframe
3. Intuitive user interface created & evidenced via feedback from technical & non-technical stakeholders
4. The web app is able to query a database where logs from the Decisioning program/software are retained
5. Low level detail of the data used in the application, including:
   1. Data used for running Scorecards + their results
   2. Summary data on an applicants Credit file
   3. Raw data used to create the above summary data

# Summary of Stakeholders Involved

Below is a summary of the Stakeholders I engaged for support and feedback as I developed the project, along with the communication methods involved:

* **F** = Face to Face
* **IM** = Instant Messaging via Microsoft Teams
* **C** = Call via Microsoft Teams
* **E** = Email

|  |  |  |
| --- | --- | --- |
| **Stakeholder** | **Methods** | **Input** |
| Credit Systems | **F, IM, C** | * Sign-off for project with Line Manager |
| Credit Risk Strategy |  | * Details on what views exist in other tools today |
| IT Test Team | **E, IM** | * Consulted for advice on how to go about testing the application   + Asked for volunteers to help conduct UAT Testing |
|  | **IM** |  |

# Design

To give myself a starting point, I referred to the Software Development Life Cycle, which as “design” as a key starting point for build a piece of software (Amazon AWS, 2024).

I needed a design to work towards before starting the coding of my web app. To do this, I started with a draft wireframe and then expanded upon it with a 2nd iteration.

## Initial Draft for page template

To keep a simple process to start, I drew a basic template on a white board for that I had in mind for a “home” page:

A white board with a drawing on it

Description automatically generated

Figure 1- Initial draft Wireframe drawn on Whiteboard

Doing this simple approach gave me a way to get a starting point for what I wanted the layout of my application to look like, without having to decide on specific details (like styling).

## Figma Wireframe

The whiteboard draft was useful to get a foundation, but I wanted to go into a little more detail before I started creating the React Application. This meant I needed a more detailed wireframe.

Figma is a solution I was familiar with, as our internal Marketing team has shared wireframes they’ve created for internal projects with me previously in the past. I believed this made it a good tool to look at for developing my wireframe into something ready for development.

Below is what I created for the Home page:

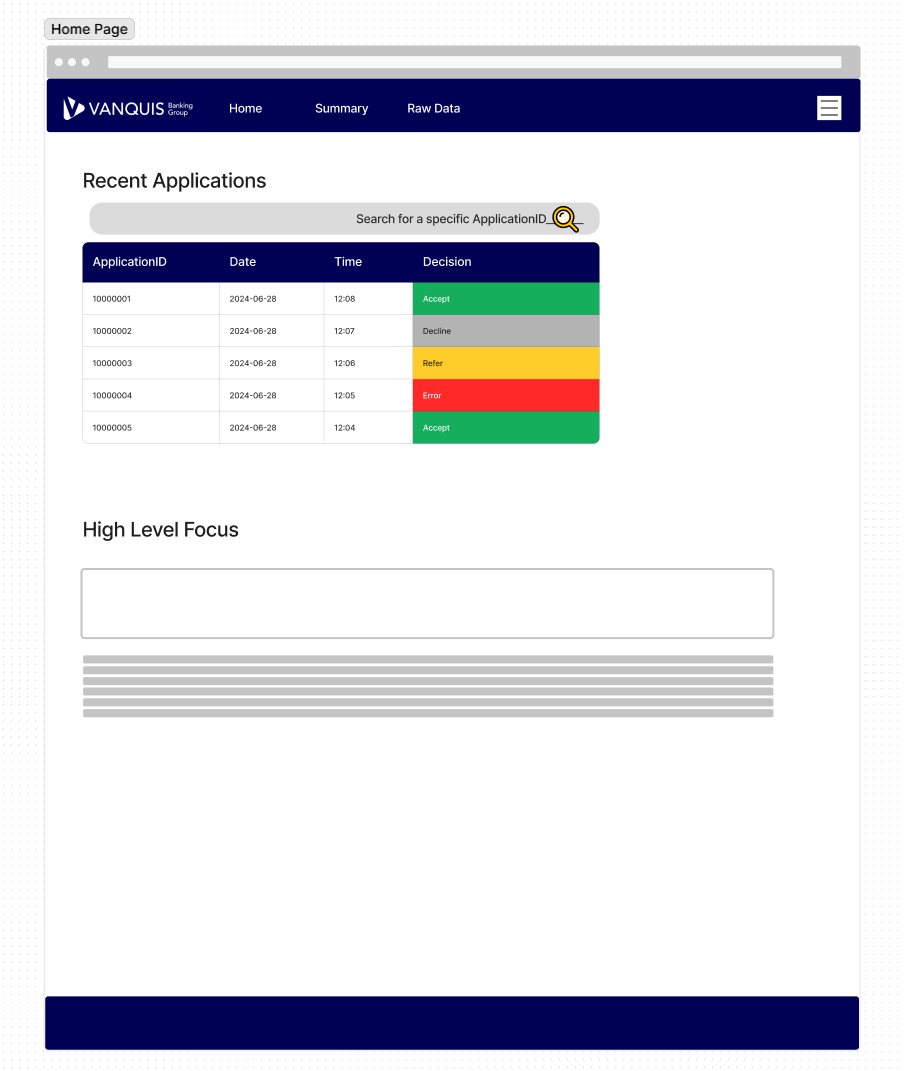


Figure 2 - Figma Wireframe of a Home Page for the React app

# Implementation / Development

## Initial setup of the React application

## Setting up Dependencies

* React-Router-dom

A screen shot of a computer

Description automatically generated

* MSSQL

A computer screen with white text

Description automatically generated

Missing:

A computer screen shot of a black screen

Description automatically generated

Fixed via (Stack Overflow, 2024).

## NPM Audit – Addressing Dependency vulnerabilities

A screenshot of a computer program

Description automatically generated

Figure 3- Output of npm audit command, showing a list of known vulnerabilities which could compromise the application

A screenshot of a computer program

Description automatically generated

Figure 4 - Output of npm audit fix command, showing that breaking changes would be needed to fix the identified vulnerabilities

## Backend – for supporting functionality

Some of the requirements I set required a full backend to be created to implement the functionality required. This included:

* Ability to connect to a database to retrieve application data (an Azure SQL Server Database specifically was used in the project)
* Searching an External file for definitions of data variables

### Why one was required

Initially I tried importing the “**mssql**” package directly into the React component to meet the 1st use case of connecting to a database to retrieve application data. When doing that however, I faced 26 compiler errors in React. These came about purely from importing the library for use:

A screenshot of a computer program

Description automatically generated

Figure 5- Importing "mssql" into a React Component called "Home.jsx"

A screenshot of a computer program

Description automatically generated

Figure 6- Compile error from importing the "mssql" library

One thing I didn’t understand was what this “**Polyfill**” concept the error messages referred to. That felt like the 1st step to resolving the error, so I did some quick research. I found that **Polyfills** are pieces of code to provide modern functionality to older browsers, at the expense of functionality and performance (Mozilla, 2024). From the context of the error message, I can see that this means that the “crypto” module mentioned in the error message is no longer automatically included, so creating a **polyfill** is necessary to resolve.

Attempting to create one of these polyfills did not resolve my issue. From further research I concluded that it would be significantly easier to set up a back-end application that the main React application could call over HTTP e.g. via the fetch method. This allows offloading complex logic into a separate app and avoids the need for complicated polyfill logic.

### Backend Set up in Node.js

After some trial and error, I came to think that it would be easiest (to develop and maintain) a separate Node.js backend and use HTTP calls within the main React application to get the necessary data. That creates a clear distinction between front-end and back-end, plus means they can be maintained separately.

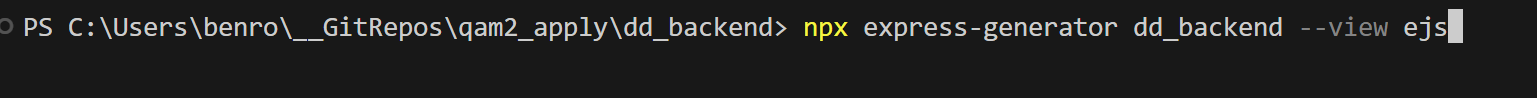


Figure 7 - Initialising an Express app based on a templete

A computer screen shot of a program

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

Figure 8 - Install http-errors dependency

### SQL - Search for Recent Applications

### SQL - Search for external Credit File data

### JS - Lookup Variable definitions

## Front End - React

### Top Navigation Bar

A screenshot of a computer

Description automatically generated

Figure 9-Top Bar Navigation for all pages on the application

A screenshot of a computer program

Description automatically generated

Figure 10 - CSS code used to make the Top Navigation bar scroll with the user

A screen shot of a computer program

Description automatically generated

Figure 11- Applying CSS for the Navigation Bar

A screenshot of a computer

Description automatically generated

Figure 12 - Top Bar Navigation scrolls with user

### Side Bar Navigation for longer pages

### 1st Feedback

A screenshot of a computer

Description automatically generated

### Data – Decoding for Business Users

* Cais transformations

## Environments configuration

One stretch goal I wanted to achieve with this project was to have the final application in a state where it would be ready to be deployed to an actual environment aside from the localhost I developed on. The aspiration was to have a UAT version of the application I could share with colleagues and gather feedback on the usefulness of the app & how easy it was to navigate.

To do this though, I had to deal with a logical flaw in my code. For example, to meet the requirement of displaying a list of recent applications, the front end React app makes a HTTP call to the Backend via the “fetch” method, which enables HTTP calls to be made in base JavaScript (**ref**). In my first version of this however, I hardcoded the URL the fetch method would call, as seen below:

A screenshot of a computer program

Description automatically generated

Figure 13 - Hardcoded URL within the React Application

This logically would not work anywhere but my local machine, so this was a barrier to meeting this deployment aspiration. To address this, I created a “.env” file that my React App could read from when it is started to get the appropriate variables for a given environment:

A screenshot of a computer

Description automatically generated

Figure 14 - .env file added to allow dynamic configurations without altering the raw code

I then altered my code to use this environment instead of the hard-coded value it previously had:

A screen shot of a computer screen

Description automatically generated

Figure 15 - Altered code to use the newly created .env file

For localhost development, I used the same localhost:port URL I had before, allowing the app to still work on my development machine. When it comes to a deployment however, I can now swap this .env file for a configuration appropriate to the environment e.g. a ”.env.uat” file with the URL for a UAT version of the back end.

The naming of the variable is due to React requiring the “REACT\_APP\_” prefix on all variables, which allows React to find the variables when using the default “npm start” command to start the server (**ref**).

### Back-End Environment version

## Personal Data (PII) Protection

~~This access management configuration can also be used for another purpose: protecting Personal Data (PII). Credit Decisioning Systems like this must collect personal data to be able to conduct a credit search on a person e.g. Experian’s DelphiSelect API requires at least:~~

* ~~Full name of the person in question~~
* ~~Date of Birth~~
* ~~At least the current address of the person in question~~
* ~~Previous addresses are also often needed if the person in question has resided at their current address for less than 3 years~~

~~(Experian, 2024)~~

~~Each of these Data Points are protected under the UK's Data Protection Act 2018, which also codifies the GDPR into UK Law (Data Protection Act, 2018). The program is currently collecting the Full Name of the user as a starting point to allow a full API integration in future. This means the program must be mindful of Data Protection/GDPR requirements.~~

~~Using a SQL databases within the program creates the risk of SQL Injection, where malicious input into the program can lead to arbitrary execution e.g. a user passes in a SQL command instead of a name (W3Schools, 2024). This is where the SELECT and INSERT permissions assigned above come into effect. The Full Name collected in the application is only saved to the “dbo.JavaDecisioningHistory” , which the integration user only has INSERT permissions on. This prevents a malicious user from entering a query to retrieve data from this table, as the database permissions will result in the query being rejected.~~

~~The only other permission the Integration User has been granted is the SELECT permission on the “Delphi.VW\_DelphiPremiumValueData” and the “Delphi.VW\_DelphiSummaryData” views. These are the only other views required by the program, so by limiting the access like this the possibility of risk incurred by SQL Injection is minimised and therefore acts as a control to protect Personal Data.~~

# Final Tech Stack

# Deployment

## Hosting in Azure Static Web App

One thing I want to do to allow testing of the application is to make it available to other users. To do this though, I needed a way to deploy the application.

I already had a Visual Studio Professional subscription (which has monthly Azure credits) from my employer, so this was my 1st idea for a hosting solution. I came across the Static Web App resources, which would allow me to link to the GitHub repository directly. Shown below is how I configured this:

A screenshot of a computer

Description automatically generated

Figure 16 - Setting up an Azure Static Web App to host the React application

A screenshot of a web application

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

After the deployment, I tried using the default URL Azure provided to see if I could access the application. This was successful:

A screen shot of a computer

Description automatically generated

Figure 17 - The React App home page as deployed to the Azure Static Web App

## Confirming access

I needed to know if this deployed application was accessible by other users. To confirm this I asked one of my colleagues (referred to as MO in the screenshot below) to try accessing the link and confirm what they see. This was successful:

A screenshot of a chat

Description automatically generated

Figure 18 - Confirmation a user can access the deployed React App

# Testing

## UAT – Gathering user feedback

## Review with Employer

# Final Application

# Conclusions

## Project Outcome

## Future Additions

# References

Amazon AWS. (2024). What is SDLC (Software Development Lifecycle)?. [Online]. AWS. Last Updated: 2024. Available at: <https://aws.amazon.com/what-is/sdlc/#:~:text=The%20software%20development%20lifecycle%20(SDLC,expectations%20during%20production%20and%20beyond>. [Accessed 28 June 2024].

Barger, R. (2021). How to Create a React App with a Node Backend: The Complete Guide. [Online]. FreeCodeCamp. Last Updated: 2021. Available at: https://www.freecodecamp.org/news/how-to-create-a-react-app-with-a-node-backend-the-complete-guide/ [Accessed 5 July 2024].

Express. (n.d.). Express application generator. [Accessed July 6, 2024], from <https://expressjs.com/en/starter/generator.html>

Mozilla. (2024). Polyfill. [Online]. developer.mozilla.org. Last Updated: 2024. Available at: <https://developer.mozilla.org/en-US/docs/Glossary/Polyfill> [Accessed 5 July 2024].

Stack Overflow. (2024). babel-preset-react-app, is importing the "@babel/plugin-proposal-private-property-in-object" package without declaring. [Online]. Stack Overflow. Last Updated: 2024. Available at: <https://stackoverflow.com/questions/76435306/babel-preset-react-app-is-importing-the-babel-plugin-proposal-private-propert> [Accessed 28 June 2024].

# Appendix

## Initial Backend attempt, with React & Node.JS using concurrently

The “mssql” module I need to connect my React application to an Azure SQL Database is a Node.js module, so I needed to create a Node backend to be able to make use of this. I follows I guide from (Barger, R, 2021) to do this.

For this to work, I required the “express” module, so used npm to ensure that was available:A screen shot of a computer program

Description automatically generated

This worked on my local machine. However it did not when I deployed to my Azure host. Rather then debug this, I thought it would be easier to implement a separate Node.js server and access as an HTTP call in the React Application.

## Learning Criteria

S2 - Develop effective user interfaces

S8 - Create simple software designs to effectively communicate understanding of the program

B10 - Committed to continued professional development

B4 - Works collaboratively with a wide range of people in different roles, internally and externally, with a positive attitude to inclusion and diversity

B7 - Communicates effectively in a variety of situations to both a technical and non- technical audience

B5 - Acts with integrity with respect to ethical, legal and regulatory ensuring the protection of personal data, safety and security

B1 - Works independently and takes responsibility. For example, has a disciplined and responsible approach to risk, and stays motivated and commited when facing challenges

B6 - Shows initiative for solving problems within their own remit, being resourceful when faced with a problem to solve

B8 - Shows curiosity to the business context in which the solution will be used, displaying an inquisitive approach to solving the problem. This includes the curiosity to explore new opportunities, and techniques; the tenacity to improve methods and maximise performance of the solution; and creativity in their approach to solutions

B9 - Demonstrates creativity and tenacity in their approach to solutions and the methods used to come to a solution, for example sees the task through to the end by devising new solutions and despite obstacles and problems along the way