**LEAN Digital Problem-Solving**

Project Proposal

A picture containing logo

Description automatically generated

**Supervisor: Mary Lyng**

**Name: Philip McNamara**

**ID: 20091414**

**Hdip in Computer Science**

**Date: 07/02/2022**

# Abstract

“Digital-Lean” is a web application & potential beginning of a digital Lean platform which allows people to enter Lean project data into a digital form. This data can then be extracted and trended in an easy-to-read dashboard. The application will be constructed with a back-end (Node.js) and Handlebar/JavaScript (Web) front end. The data will be stored in a non-relational Database utilising the MongoDB as the primary application. This data will then be accessed using ODBC connections in Power Bi for Dashboard creation. And displayed back to the user in a Dashboard page.

# Declaration of Authenticity

I declare that the work which follows is my own, and that any quotations from any sources (e.g. books, journals, the internet) are clearly identified as such by the use of ‘single quotation marks’, for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Date: 07/02/2022

# Preface

Document to be read in conjunction with …

GitHub Repo : <https://github.com/philipmcnamara/FinalProject.git>

# Acknowledgments:

First and foremost, I would like to thank my wife, Grace. Without her infinite patience and support I never would have finished this course, let alone this project… on reflection has an infinitely higher natural programming capability than I do…I mean if I’m honest she should probably have her name on degree if I get one.  
I’d also like to give a special shout out to my kids Penelope & Elijah who I almost didn’t recognise the other day it’s been that long since I’ve actually had the free time to spend with them.

It’s also probably worth acknowledging my employers, but honestly, they were no help at all so I won’t.

I’d defiantly like to give a big shout out to all my lectures who made this a truly interesting and stimulating course along with just being really helpful and friendly people. Special thanks also to my project supervisor Mary Lyng.

# Keywords

JavaScript, Handlebars, Hapi, Lean, Node, MongoDB, PowerBi, AWS.

# Commercial Title:

“Digital Lean” – The problem solver.

# Academic Title:

Digitisation & data trending of Lean problem-solving techniques via Web application utilising mongo cloud atlas and AWS.

Table of Contents

[Abstract 2](#_Toc96857590)

[Declaration of Authenticity 2](#_Toc96857591)

[Preface 3](#_Toc96857592)

[Acknowledgments: 3](#_Toc96857593)

[Keywords 3](#_Toc96857594)

[Commercial Title: 3](#_Toc96857595)

[Academic Title: 3](#_Toc96857596)

[1 Introduction 6](#_Toc96857597)

[1.1 Background 7](#_Toc96857598)

[1.2 The Problem 7](#_Toc96857599)

[1.3 Scope 8](#_Toc96857600)

[1.4 Objective 10](#_Toc96857601)

[1.5 User Requirements 10](#_Toc96857602)

[1.6 Unified Model Language of Web App 10](#_Toc96857603)

[2 Design 11](#_Toc96857604)

[2.1 Web Application 11](#_Toc96857605)

[3 Methodology 13](#_Toc96857606)

[3.1 Waterfall 13](#_Toc96857607)

[3.2 Drawbacks of the Waterfall Approach 14](#_Toc96857608)

[3.3 Benefits of the Waterfall Approach 14](#_Toc96857609)

[3.4 Agile 15](#_Toc96857610)

[3.5 The 12 principles articulated in the Agile Manifesto 15](#_Toc96857611)

[3.6 Iteration Plan 17](#_Toc96857612)

[4 Technology Choices 18](#_Toc96857613)

[4.1 Front End 18](#_Toc96857614)

[4.1.1 Hapi 18](#_Toc96857615)

[4.1.2 React 20](#_Toc96857616)

[4.1.2 Angular 21](#_Toc96857617)

[4.2 Server 22](#_Toc96857618)

[4.2.1 Node.JS 22](#_Toc96857619)

[4.2.1 Meteor.JS 23](#_Toc96857620)

[4.3 Database 24](#_Toc96857621)

[4.3.1 MySQL 24](#_Toc96857622)

[4.3.2 mongo DB 24](#_Toc96857623)

[4.3 Dashboard Tools 25](#_Toc96857624)

[5 Implementation 26](#_Toc96857625)

[5.1 Iteration 1 26](#_Toc96857626)

[5.2 Iteration 2 26](#_Toc96857627)

[5.3 Iteration 3 26](#_Toc96857628)

[5.4 Iteration 4 26](#_Toc96857629)

[5.5 Iteration 5 26](#_Toc96857630)

[5.6 Iteration 6 26](#_Toc96857631)

[5.7 Iteration 7 26](#_Toc96857632)

[5.8 Iteration 8 27](#_Toc96857633)

[5.9 Iteration 9 27](#_Toc96857634)

[5.10 Iteration 10 27](#_Toc96857635)

[6 Problems Encountered 28](#_Toc96857636)

[6.1 Dynamic Table 28](#_Toc96857637)

[6.2 Mongo Data format in PowerBi 28](#_Toc96857638)

[6.3 Git Error during deployment 29](#_Toc96857639)

[6.4 Mongo Cloud Paywall 30](#_Toc96857640)

[7 Reflection 31](#_Toc96857641)

[7.1 What I learned 31](#_Toc96857642)

[7.1.1 Personally 31](#_Toc96857643)

[7.1.2Technically 32](#_Toc96857644)

[7.2 What I achieved 33](#_Toc96857645)

[Works Cited 33](#_Toc96857646)

Figures

[Figure 1LEAN A3 WORK-SHEET 8](#_Toc96791838)

[Figure 2UNIFIED MODEL LANGUAGE OF WEB APP 11](#_Toc96791839)

[Figure 3WEB APP DESIGN OVERVIEW 12](#_Toc96791840)

[Figure 4 Design Sketches 13](#_Toc96791841)

[Figure 5 Iteration Plan 14](#_Toc96791842)

[Figure 6 AGILE VS WATERFALL 18](#_Toc96791843)

[Figure 7 Hapi Framework 19](#_Toc96791844)

[Figure 8 React Framework 21](#_Toc96791845)

[Figure 9 Framework Popularity 21](#_Toc96791846)

[Figure 10 Angular Framework 22](#_Toc96791847)

[Figure 11 Node.js 23](#_Toc96791848)

[Figure 12 Meteor JS 24](#_Toc96791849)

[Figure 13My SQL 25](#_Toc96791850)

[Figure 14 MongoDB 25](#_Toc96791851)

[Figure 15 Power Bi 26](#_Toc96791852)

[Figure 16 MongoDB data in Power Bi 29](#_Toc96791853)

[Figure 17 Github mistakes 30](#_Toc96791854)

[Figure 18 MongoDB Cluster paid version features. 31](#_Toc96791855)

# 1 Introduction

The main purpose of this project will be to create a functional digital substitute with a data trending dashboard for a currently paper or excel based process.

## 1.1 Background

The LEAN problem-solving mythology is a relatively standardised process and has remained mostly unchanged since its inception nearly a century ago. The biggest update to the problem-solving process was to move it to excel from paper but many large-scale companies such as Toyota (who invented the process) still routinely use a printout for their LEAN workshops. I also utilise this approach in many workshops and then have to convert my data into csv format for loading into a SQL so I can then utilise the data in analytical software such as PowerBi or Tableau.

The Problem-solving Process itself if relatively straight forward and enables novices to get to the root of many business issues using the methodology (often referred to as an “A3” as it is usually printed out on an A3 page). It follows a series of 7 steps and guides the user to ask several probing questions, record the answers then create actions and assign them to people on the team for completion within a timeframe. The actions can be tracked, and the project managed through metrics.

## 1.2 The Problem

This process of extracting the data from the A3 can be very time consuming especially when the number of projects I manage can be >40 per month. There is also an issue of handwriting being indecipherable and this leads to data loss. There is also an environment impact to consider from hundreds of companies utilising this method every year wasting thousands of sheets of paper.

Graphical user interface, application, table, Excel

Description automatically generated

Figure 1LEAN A3 WORK-SHEET

## 1.3 Scope

The Digitalisation of the A3 into an interactive web app should have functionality that enables it to be logged into. Depending on who is logged in the site should display the stored persisted data back to the user in a dashboard. This satisfies the primary goal of this project, and it allows other members of my project team to utilise this tool to track their projects as well.

The application is more valuable as a web app as it can be accessed by many computer types (IOS, Windows, Linux) while an android app limits the availability to android users of which there are few as the company issues iPhones.

The scope therefore includes a web application front end. The web app will then require a server to access the data and perform the logical requests from the user. For this reason, a Node.js backend will be included in the scope. It will handle user authentication, user account creation, recording of data and then retrieval and presentation of said information.

The front-end dashboard will need to communicate with the database in order to display this information to the user via a homepage. This dashboard can be created using either open-source tools such as Metabase, Redash or via an embedded PowerBi Dashboard if the connection to the Mongo data once the connection is established. The dashboard is essential to the user and is included in the scope of the project for this reason.

The UI is designed to accommodate for speedy on the go recording of data but also provide information & advice on the process for each input field to guide unfamiliar users through the problem-solving process. This as achieved via the tool tips and ghost writing in each input field. As a failsafe some data validation will be employed as a means of catching bad data which may negatively affect the dashboard if it is added in error. This is achieved by adding in date pickers or int only fields were relevant. Additional basic data validation is included where possible in the form of prompts, checks and dropdown options.

The application will need to be accessed for use and testing by myself and potentially by my larger team as this product may be expanded on and a feedback loop would be highly beneficial towards future development. To this end the web app is to be deployed using the AWS ec2 instance using Linux with Ubuntu distro so it can be accessed not just from my localhost, additional deployment on Heroku is also possible as the process is quicker than AWS and the logs are highly beneficial for bug fixing.

The project is essentially an “Iceberg tip” for what could potentially become a viable LEAN business software package once several more features are added into it. This project will stick to delivering a digital replacement for problem solving. If time permitted, it would be developed in a fully modularised fashion but as it stands and due to its small size it is potentially more practical to stick with a basic monolithic design with full modularisation remaining out of scope.

Further expansion and incorporation of additional Lean tools and info guides covering JIT, bottleneck Analysis, Value Stream Mapping (VSM) & OEE calculation software are all very likely expansions, but for now these features are out of scope and will be developed using featured releases on GitHub.

## 1.4 Objective

* Create an online web app that allows unique user login.
* Data persistence for all recorded data that is retrievable.
* Data storage in MongoDB.
* Deploy application.
* Data trending with basic metrics or an embedded custom-built Power Bi application.

## 1.5 User Requirements

* Login in and register screen.
* A dashboard homepage landing screen, illustrating the number areas responsible for causing issues, & no of projects per person.
* 5 – 7 pages to illustrate each section of the A3.
* Ability to navigate to each screen without losing data.
* Ability to edit a project that is “Competed”.
* Projects must be able to be deleted.

## 1.6 Unified Model Language of Web App

Diagram

Description automatically generated

Figure 2UNIFIED MODEL LANGUAGE OF WEB APP

Figure 2 indicates how the web application will operate. The user will be presented with a Login or Signup screen to start, the application will not proceed until the user has an account.

From there they will be presented with a Dashboard homepage with option to start a new project/ Continue a Project or to go to the settings and update their personal info. The Project will consist of 6 pages each representing a section form the original A3.   
There will be no Measures page though as this will now be captured automatically by the Dashboard page.

# 2 Design

## 2.1 Web Application

As illustrated in Figure 3 below the user can access the application via a pc browser. This is referred to as the “client-side” which implies that the application runs on the users' computer. Client-side is typically a stand-alone application but, in this instance, it represents a web browser-based interface and anything that is displayed to them on the screen, incorporating the UI and dashboard. The client-side allows the application to become dynamic which changes the view, based on the user’s input.

As the user navigates and interacts with the web application, the client will communicate with the node application. The node application can be referred to as either the Server as illustrated in Figure 1.2 or “back-end” which is more commonly used. Back-end (not to be confused with Bag-end, as that’s where Bilbo lives) refers to processes and services that are running either on another server or in the background of the users' computer. More often than not, it refers to processes that are not on the end users' computer and in this instance that is the Node.js server. The node application itself is responsible for running the logic within the application while communicating with the Mongo database. This communication will provide CRUD database functionality. CRUD is an acronym for Create, Read, Update and Delete. It is a set of operations that servers can execute (POST, GET, PUT and DELETE requests) these four functions that are considered necessary to implement a persistent storage application like this.

The final element to the project is the database. This is where the application data will be stored. For this app I will use MongoDB, this is an open-source document-oriented database. It is typically used to store larger volumes of data and then allow the user to easily interact with it. MongoDB is not based on the table-like relational database structure commonly known as SQL but instead provides an altogether different mechanism for storage and retrieval of data, which is creatively known as a NoSQL database. NoSQL simply means “non-relational”. The format of storage is called BSON which is similar to the increasingly popular JSON format.

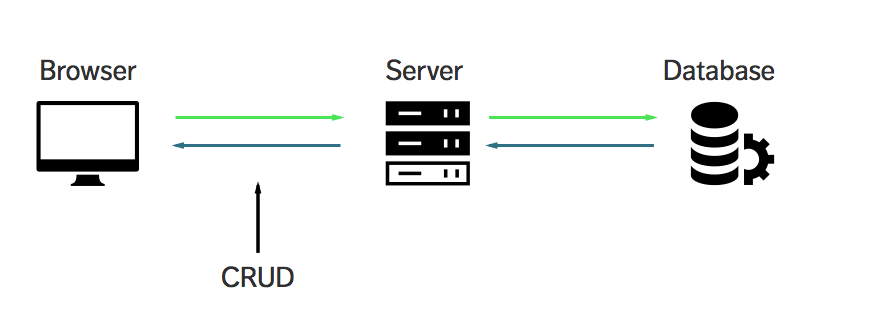


Figure 3WEB APP DESIGN OVERVIEW

The UI design approach is to keep each screen as empty as possible. To that end a series of sketches were done before any of the coding was started. These are visible below in Figure 4.

Chart

Description automatically generatedChart, bar chart

Description automatically generated

Figure 4 Design Sketches

These sketches are focused entirely around the “Form” that will be used to replace the old A3 form. The rest of the web site will be designed using the Ulkit font end cosmetics following a less is more approach in order to accommodate the use of negative space.

# 3 Methodology

For the creation of this project, I will employ both the Waterfall Development Methodology & the Agile Methodology. Simply put because I prefer using Waterfall and it will suit the early stages of the Web app allowing for rapid development of a basic Web app. I will then switch over to a sprint model as the more complex element of the project like creating the dynamic tables and data retrieval/editing are not something I’m entirely comfortable planning out in Waterfall. Switching over to agile at this point allows for increased flexibility, thus, will be better suited to dealing with this workload in a manageable timeframe.

## 3.1 Waterfall

Waterfall is a form of project management I’m familiar with and as the app is quite simple to begin with, this will allow me to plan and release it without adding unnecessary complexity and time. A version of waterfall is also used in lean called Kanban. There are however some subtle differences between the two with Kanban typically utilising physical boards or tables, while their columns depict particular stages of work (Sergeev, 2016). The tasks are placed on Kanban boards in accordance with the stage of their performance. While waterfall is a sequential approach that divides the software development life cycle (SDLC) to distinct phases such as requirements gathering, analysis and design, coding and unit testing, system and user acceptance testing then finally deployment. (SANTOS, 2021) With waterfall the next phase can only proceed if the previous phase has been completed. In between phases, a deliverable is typically expected, or a document is signed off.

Or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or, or,

## 3.2 Drawbacks of the Waterfall Approach

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage. A Tutorial Point article highlights this as an issue as no working software is produced until late during the life cycle, so obvious errors or failings can often be missed. (Tutorial Point, 2019) Waterfall is also not ideally suited for projects where requirements are at a moderate to high risk of changing, as these projects are often complex, and waterfall allows for little adaption so everything must be considered during the planning stage and unforeseen errors can throw the whole project off.

Adjusting scope during the life cycle can add a significant level of complexity to the project. This is often referred to as “Scope creep” in industry and is the largest single cause of project failure I’ve seen in my 7 years in ITC project management. Software development is a process that is by its very nature subject to a high degree of variation and hampering this with a linear approach such as waterfall can cause missed deadlines and frustration amongst developers and clients alike.

## 3.3 Benefits of the Waterfall Approach

The requirements are clearly and accurately stated, and typically remain unchanged throughout the entire project. Careful planning of the project development structure reduces the number of problematic issues. The start and end points for each phase are set, which makes it easy to measure progress. The tasks remain as stable as possible throughout the development process. Release date for the finished product, as well as its final cost can be calculated prior to development. (Lvivity, 2018). Thus for the first 50% of the project where I need to create a basic web app with a database connection and I know all of my goals, Waterfall will allow me to draw out a comprehensive plan and rapidly produce my foundations without any unnecessary time delays that the planning phase can cause. However, it will not be a sufficient approach during the later stages as there are element of the programming I simply do not know how to do, so any waterfall planning I incorporate them into have huge potential to substantially inaccurate which can cause time delays and stress.

## 3.4 Agile

The other obvious choice for software development is the increasingly popular, and in my personal experience, largely misunderstood by most senior managers is the agile approach. “Agile” is now a trendy buzz word in the business community along with “the blockchain” and “SCRUM” but beyond the hype is a comprehensive and structured system. Agile methodology is a type of incremental approach to software development based on principles that focuses more on people, results, collaboration, and flexible responses to change. Instead of planning for the whole project, it breaks down the development process in small increments completed in iterations, or short time frames, these are often called sprints and typically last about 1 - 2 weeks, although they can be any timeframe up to a month. I will be using 1-week sprints as a way breakdown complicated processes into manageable workloads.

## 3.5 The 12 principles articulated in the Agile Manifesto

According to (Mary K. Pratt & Christina Torode, 2020) there are 12 principal of the agile manifesto which are as follows:

* Satisfying customers through early and continuous delivery of valuable work.
* Breaking big work down into smaller tasks that can be completed quickly.
* Recognizing that the best work emerges from self-organized teams.
* Providing motivated individuals with the environment and support they need and trusting them to get the job done.
* Creating processes that promote sustainable efforts.
* Maintaining a constant pace for completed work.
* Welcoming changing requirements, even late in a project.
* Assembling the project team and business owners on a daily basis throughout the project.
* Having the team reflect at regular intervals on how to become more effective, then tuning and adjusting behaviour accordingly.
* Measuring progress by the amount of completed work.
* Continually seeking excellence.
* Harnessing change for a competitive advantage.

Agile, as outlined in the Agile Manifesto above, is considered a philosophy, although the same can also be said of Kanban. This is not really the case for Waterfall however as it a more cascading linear approach as illustrated in Figure 6.

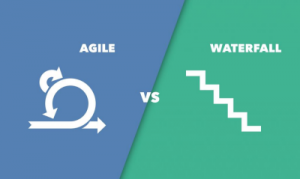


Figure 6 AGILE VS WATERFALL

Agile relies on incremental releases and customer feedback, as I will not be releasing to customers per say I will instead be releasing my update to my Wife who I will pretend is a customer and get her feedback on my approach to solving each problem addressed in my sprints. This feedback loop is essential in delivering a functional product. There are “easy” ways to achieve the creation of the digitalised form that would most like result in hard to follow or an overly busy UI. Like wise there are incredibly time consuming but highly impressive minimalist UI approaches the form could also take. Estimating the time required to create the desired outcome with my limited software experience is a fool’s errand. Waterfall is entirely useless at this point but Agile with its degree of flexibility is realistically the only appropriate methodology available to complete this task.

## 3.6 Iteration Plan

The project was developed using a hybrid approach as specified above. Figure 5 demonstrates the original waterfall list I used to plan out the project. Steps 1- 10 were Waterfall, and the rest of the project was broken down into 1-week sprints with more detailed plans drawn up for each subsequent step/target. This was the very first plan for the project I had drawn up back in November and it was obviously added to and augmented quite a few times as the project progressed. The reason I add this version of the Waterfall plan was to illustrate simply how quick it is to put together a plan and begin working on it. It’s easy to procrastinate or become distracted when you have a massive mountain of work to start. Basic Waterfall like below removes the temptation to digress into other less relevant areas of work and instead keep working entirely on the single problem you are trying to solve.

A picture containing timeline

Description automatically generated

Figure Iteration Plan

# 4 Technology Choices

There are several technology choices available to create the proposed web app all of them frameworks. According to research conducted, frameworks are designed to boost web development efficiency and performance by providing a consistent structure. This essentially means that developers don’t have to keep rebuilding their code from scratch meaning that frameworks are time savers that offer developers a host of extra features that can be added to software without requiring much additional effort. (Deshpande, 2021).

Consideration must also be given to some of the out-of-scope features that will eventually be added into the application should a successful business model actually be created. To ensure I have made the correct choices for my app now and its potential future application I have broken down each major section i.e., front end, server, and database into all potential options I feel I would be capable of adapting. Some consideration has been made for learning however, particularly around the database side of things where SQL is quite clearly the better choice but despite this no-SQL is the approach chosen. I did this as wanted to learn more about how to use and interact with no-SQL whilst I still have an academic opportunity to do so. At any rate this can quite easily be augmented into a SQL database at a later point by adding an SQL model class into the application and changing the settings in the index & .env files.

## 4.1 Front End

### 4.1.1 Hapi

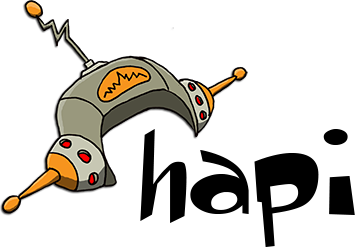


Figure 7 Hapi Framework

Hapi.js (also known as hapi) is an open-source Node.js framework used to build scalable web applications. Hapi is commonly used to build Application Programming Interface (API) servers, HTTP-proxy applications, and websites. (Malik, 2018).

Hapi.js comes with a fair number of unique features that enable developers to build secure and scalable applications. The main features available and relevant to myself are:

* End-to-end code hygiene which helps write manageable and controllable code
* Hapi blocks error messages that may leak information or echo back exploits.
* Encrypted and signed cookies, secret or key rotation, and HTTP security headers. Combined these are all meant to enhance the security of applications. Which given the current climate, is a big win.
* There are an extensive set of official plugins that are meant to replace middleware used in frameworks. Middleware is software that provides common services and capabilities to applications outside of what’s offered by the operating system. Data management, application services, messaging, authentication, and API management are all commonly handled by middleware so having this managed in the framework to a small extent is highly beneficial.
* Integrated Authorization and Authentication Architecture which is the most comprehensive authorization API available in Node.js. (Kayere, 2020)

The main reason I have chosen hapi.js is that it easily integrates with MySQL & MongoDB. It is an easy-to-use front-end framework now capable of rivalling the more popular Angular and Vue.js to create single page applications. Hapi allows you build your apps in a modular way by making use of its fairly robust plugin system and npm package manager, "npm install” being an extraordinarily easy and quick way to add libraries via the console. Although this specific web application is likely to be built in a relatively monolithic fashion due to time constraints, I will modularise it as much as possible for future out of scope applications I plan to eventually expand upon.

### 4.1.2 React

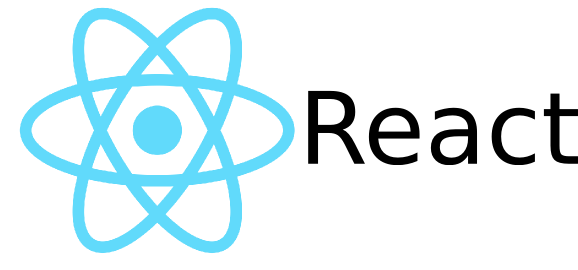


Figure 8 React Framework

Easily one of the more popular frameworks to arise in the past 5 years as illustrated in Figure 7 below, React is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It’s used for handling the view layer for web and mobile apps. React also allows us to create reusable UI components. React was first created by Jordan Walke, a software engineer working for Facebook. React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application. (Pandit, 2021).

Graphical user interface, chart

Description automatically generated

Figure 9 Framework Popularity

The main reasons I would consider react for this application are:

* Simplicity - ReactJS is just simpler to grasp right away and this will save me time in the creation of front end functions so I can focus more on getting my back end logic working effectively.
* Easy to learn - Angular and Ember are referred to as ‘Domain-specific Language’, implying that it is difficult to learn them. (Pandit, 2021).
* Native Approach - React can be used to create mobile applications (React Native). And React emphasises the DRY approach, meaning extensive code reusability is supported saving time and code.
* Testability - ReactJS applications are quick & easy to test.

### 4.1.2 Angular



Figure 10 Angular Framework

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your applications.

The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks of the Angular framework are Angular components that are organized into NgModules. NgModules collect related code into functional sets; an Angular application is defined by a set of NgModules. An application always has at least a root module that enables bootstrapping, and typically has many more feature modules. (Deshpande, 2021).

The main reason I considered Angular are that it’s highly usable straight out of the box and comes with a range of features that allow developers to achieve a high speed to market (Stodulski, 2020) or in this case project completion.

Its ecosystem is vast, with tons of ready-made components that make scaling a project easy. What’s more, TypeScript reduces the burden of debugging, with most common errors being flagged during development, however as I would have to learn this language anew, I will likely only use Angular if I do not have the tools, I require from either React of Hapi.

## 4.2 Server

The server is responsible for running the logic within the application while communicating with the Mongo database. This communication will provide CRUD database functionality.

### 4.2.1 Node.JS

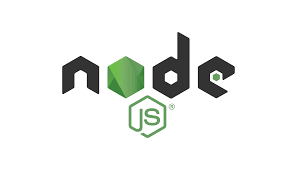


Figure 11 Node.js

Node.js is a software development technology that is used by some of the biggest players in global business, from Netflix and PayPal to LinkedIn, to build fast, high-quality applications. With this great popularity comes a great responsibility for developers to create frameworks that make Node.js application development easier. There are various frameworks currently available such as express, hapi.js (as mentioned above) and Koa, along with several others. Node.js is an open source, cross-platform runtime environment and library that is used for running web applications outside the client’s browser. (Capan, 2016). So essentially, it’s exactly what I’m looking for.

It is used for server-side programming, and primarily deployed for non-blocking, event-driven servers, such as traditional web sites and back-end API services. Node.js is neither a framework or a library but is instead a runtime environment, which contains Web API’s that a developer can access to build a code, and a JavaScript engine that parses that code. (Sufiyan, 2021).

There are several reasons to choose Node.js, but above all else then main reason I have gone with it, is it accommodates full stack JavaScript Web apps. This allows me to write front-end and back-end Java Script making the entire process quicker and much easier to complete to a higher standard. The integration of Hapi is also a major benefit as it enables my app design to be created more easily and implemented into my chosen hybrid Waterfall/Agile methodologies more effectively.

### 4.2.1 Meteor.JS

Logo, company name

Description automatically generated

Figure 12 Meteor JS

Meteor is a full-stack JavaScript platform for developing modern web and mobile applications. Meteor includes a key set of technologies for building connected-client reactive applications, a build tool, and a curated set of packages from the Node.js and general JavaScript community. Like Node.js Meteor is also a framework that allows for full stack JS development. (Matviiok, 2021). Meteor also uses “data on the wire”, meaning the server sends data, not HTML, and the client renders it. Meteor integrates with all the Font end frameworks React and Angular, but notably not with hapi.js which is specific to Node.js only.

The Meteor Framework is designed for speed. Which is why I have included it for comparison to what would have ultimately been a very easy choice in Node.js. Meteor is reactive, this concept presumes a rapid automatic change of the user interface with each change in data meaning for example, once somebody posts a comment under an article, it can be seen it immediately, without reloading the page. So, from a UI perspective Meteor is a powerhouse. Although Node.js also offers this functionality the Meteor team press home the point that Meteor does it faster (Matviiok, 2021).

## 4.3 Database

A Web database is a database application designed to be managed and accessed through the Internet. Website operators can manage this collection of data and present analytical results based on the data in the Web database application. Databases first appeared in the 1990s, and have been an asset for businesses, allowing the collection of seemingly infinite amounts of data from infinite amounts of customers. (North, 2015).

### 4.3.1 MySQL



Figure 13My SQL

Relational Databases are ordered in rows and columns which form tables. Multiple tables can be related to one another which allows the ability to join tables in order to query the data. I use SQL tools in work all the time. Relational data is easy to read, easy to manage and easy to generate dashboards from. As I already know how to do all this, I will not be using relational data. It is 100% the best choice for my application as I will be generating dynamic tables of data. But I won’t learn anything new, and this is my last opportunity to do in an academic environment whilst I have the support and guidance from experienced lecturers.

### 4.3.2 mongo DB



Figure 14 MongoDB

Allows the storing of unstructured data which is not in a table, rows and columns format. Non-relational data often called no-SQL is rare in industry but does pop up occasionally. There is a no-SQL database in work I have been asked to extract info from occasionally. I have been mostly unable to do this and have had to pass on the work. For this reason, I will be using no-SQL and specifically mongo DB. It is open source and the most common no-SQL data base in use. The data in my Web app is far more suited to a relational database as mentioned above so there is a significant opportunity to learn how to manage a no-SQL database to a higher standard than I currently possess. For this reason alone, I will be using no-SQL and mongo DB.

## 4.3 Dashboard Tools

A picture containing text

Description automatically generatedFigure 15 Power Bi

There are several open-source dashboard tools available, Meta base was one I investigated along with the more popular Dashboard applications on the market such as Tableau and PowerBi. While open-source options are interesting it has been my experience in industry that there are rarely used and a quick look on Irishjobs.ie gives a compelling argument that Industry requirement around data visualisation is currently being dominated by PowerBi. Given that my future work prospects are heavily data focused I have chosen PowerBi as my dashboard application for this reason.  
  
Complications arising from this decision are obvious being that Power Bi and MongoDB do not play well together and are described in more detail in section 6.2 “Problems Encountered”.

# 5 Implementation

## 5.1 Iteration 1

Created a GitHub repo and a basic node web app then added in handlebars and converted the app to utilise them. Implemented simple Login/Sign-up functionality. Created simple textbox to be expanded into the Add Project functionality.

## 5.2 Iteration 2

Added a Settings page where the user can update their information. Installed MongoDB and Rob3T. Linked Mongo to the Web app and added the models into the structure. Configured Hapi and added in some basic data tests.

## 5.3 Iteration 3

Updated the tests significantly and ensured they were all running. Added in basic webapp security with password hashing/salting. Added data acceptance criteria around passwords. Added a Navigation bar to the site. Added edit/delete functionality to the project section.

## 5.4 Iteration 4

Added in forms for the Add Project functionality, there were several unsuccessful attempts at this. Form data now linking to Mongo.

## 5.5 Iteration 5

Added in dynamic tables to the Action plan section of the form. Table data updates to mongo dynamically. Table data displays in the update page dynamically. Delete Project functionality rebuilt.

## 5.6 Iteration 6

Rebuilt the Update Project screen massively. Table data had to be extracted and repackaged before it could be sent back into Mongo correctly. This was the hardest thing I’ve done to date.

## 5.7 Iteration 7

Power Bi downloaded and several training modules done online. Mongo drivers & power Bi connection tools installed. Computer ODBC setting configured. Mongo data loaded into Power Bi via ODBC connection. Data relationships created. Dashboard Built. Refresh working as expected. Dashboard published into PowerBi online.

## 5.8 Iteration 8

Power Bi embedded code created and added into dashboard.hbs file and configured. Data tested and refreshed functionally updated. PowerBi gateway downloaded and installed on machine.

## 5.9 Iteration 9

Tooltips added into the main form. Over 50 iterations of changes on GitHub were done to successfully enable deployment on Heroku. There were several issues form my node.js version not being compatible with my mongoose version & Heroku. Eventually the .env file had to be removed form my Gitignore and code was removed from my db file in the model. Heroku deployment was achieved.

## 5.10 Iteration 10

AWS account was set up and an ec2 instance with appropriate security settings was created.   
Mongo Cloud atlas server was created and linked to the Web App.

Massive issues with bcrypt in my app not being compatible with my ec2 instance were encountered. There were resolved by upgrading my instance permission using the following steps :  
$ sudo chown root:root /home

$ sudo chmod 755 /home

$ sudo chown ec2-user:ec2-user /home/ec2-user -R

Successful deployment to AWS was achieved along with the linked cloud mongo server.

A mongo DB paywall issue was encountered at this point. This is described in further detail in section 6.4.

The project was marked as complete at this point. Food tasted better, the birds started singing, children played in the fields … life as it were, began to emerge anew from the creature that had been arched over a keyboard every free moment for 2 solid months.

# 6 Problems Encountered

## 6.1 Dynamic Table

Basic HTML forms are not designed to be used in the dynamic way I needed mine to run in & link with MongoDB. It would be reasonable to assume had I used MySQL as opposed to Mongo I would have saved myself a lot of hardship. Instead, I had to add several loops into my controller methods and then again into my Handlebars front end. As the information returned to the DB was expected to be a string, I then also had to run several parsing steps to collect it back in the correct format so the editing process can run in friendly UX.

## 6.2 Mongo Data format in PowerBi

PowerBi and MongoDB don’t exactly play nice. PowerBi is designed to be used with Structured SQL data. That being said I managed to find the mother of all workarounds. MongoDB actually have a PowerBi connector and the drivers to run it, so from there I Configured ODBC to link to these drivers on my machine and set up System DSN on the corrects ports for mongo. Then in PowerBi I could connect to Mongo using the ODBC connection channel and convert this no relational data into relational data using in the PowerBi system, as seen in Figure 14 below. I then created an Azure account, then created a company off myself in Microsoft active Directory. I then gave myself admin access on PowerBi and signed up for a free “premium” trial. The allowed me to generate an embed coded for my dashboard. I embedded it into my site and adjusted the sizes.

Graphical user interface, application

Description automatically generated

Figure 16 MongoDB data in Power Bi

## 6.3 Git Error during deployment

During an attempt to deploy out to Heroku I encountered several Node errors. Whilst experimenting with solution I accidently removed my .env file and ended up pushing all of my private files to git. I then removed these in git which caused a pull in my app and then the important files were permanently deleted from my app and the entire thing crashed. These steps are visible here in Figure 15 :

A screenshot of a computer

Description automatically generated with medium confidence

Figure 17 Github mistakes

These errors were unfixable. So in my panic I pulled down the last stable iteration and push it into a new branch I called “Master” and set it as my default branch.   
Further info I found on git now allows me to understand I could have simply used a git rollback.

## 6.4 Mongo Cloud Paywall

Setting up my local mongo server, I was able to connect my Power Bui fil via the ODBC connection type. Once I deployed my site to ec2 I had to switch my server over to a free tier M0 mongo atlas cluster. It is technically impossible to connect Power Bi to any free mongo tier. I found several videos and guides on how to do this with the paid version as shown in Figure 16 below. There is simply a Business Intelligence Connector function that enables ODBC connection to Power Bi which my M0 cluster does not have.   
For this reason, I am unable to connect my Power Bi to my cloud mongo database.   
All of the same connectivity functions I previously used on my localhost mongo dB are applicable if the paywall wasn’t present.   
I therefore will be forced to demonstrate my successful deployment and connection as a small proof of concept at the end of my demo video and use the localhost connection as my primary system.   
  
*Graphical user interface, text, application, email

Description automatically generated*

Figure 18 MongoDB Cluster paid version features.

# 7 Reflection

## 7.1 What I learned

### 7.1.1 Personally

**Development Methodologies.** I took a hybrid approach to development by planning out using the waterfall approach over the Christmas break what I would be aiming to achieve for the first 4-6 weeks of the project. Then switched to agile once I entered an area were the approach to achieve the goal were not entirely specified, I was able to understand fully were each approach falters and gains strength. If starting another project of this scale or adding in additional modules I would employ this approach again as the project was completed far ahead of schedule by allowing for rapid development where available and careful development where necessary.

**Planning Software.** I use Jira in work and used to use Trello in my last job every day to plan and keep track of things. As I was planning out my waterfall approach over the Christmas break I did all of my brainstorming on paper and just used simple bullet point lists of task to complete in order to create an MVP. I had planned on using Trello but soon realised I was just better off spending all of my available time developing to complete the bullet points objectives with as much haste as possible than wasting my time doing up fancy planning & tracking boards. If I was developing in a team, I would absolutely use one of these online resources but for myself the is simply no need to waste time doing so. The output is what is important to me, everything else is essentially waste.

**Git.** Console commands are infinitely faster than messing about with a UI. I will never use a UI for git again.   
**Pointless stress.** While stuck on a bug of trying to figure out how to make that table dynamic I spent days staring at my screen completely stressed out and getting nothing productive completed. Small breaks every 40 – 50 minutes even just pacing around the kitchen chatting make such a big difference there really is a “Can't see the wood for the trees” issue with development and these breaks are not just important but essential. It’s hard to walk away when you’re halfway down a rabbit hole thinking “this is defiantly the solution to my problem” as 99% of the time it usually isn’t so just take the break and really think about the problem.

### 7.1.2Technically

**Loops in Handlebars.** These were the solution to a major issue I spent a long time working on. They are simple “each” loops as “for” loops do not currently exist in Handlebars. That being said, Each loops are very powerfully and opening <script> tags to perform front end JS function in was invaluable to this project.

**Too many Dependencies.** The more libraries and frameworks you use the more likely you are to encounter issues with compatibility. I was unable to deploy to Heroku for a significant period of time as my node version was too new and Heroku couldn’t run the npm steps. Assumption that everything will play nice are dangerous and can eat up a lot of time. It’s entirely possible I spent as much time configuring my dependencies as I did coding the entire project. From bcrypt not installing on my ec2 instance correctly to the .env configuration not allowing me to deploy at all resulting in weeks of bug fixes. I am far more cautious and cognisant about which dependencies I will be using in future and how they may interact with each other. That being said, every issue I encountered (bar my mongo DB paywall blocker) did have a solution, it just took time and patience to resolve. I can see why experienced developers are in such high demand as if I started this project with the knowledge I have now it would be reasonable to assume I could fire the whole thing out in about 3 – 4 weeks top to bottom. On reflection there is no amount of theory that can replace actual hands-on coding and bug fixing. The best method to learn (for me anyway) is to simply, do.

## 7.2 What I achieved

I have successfully built a Web app that takes in data, dynamically adjust tables for viewing and editing. The system then links to Power Bi where the NoSQL data is structured and used to create dynamic dashboards which are embedded into the webpage.   
The web page is relatively modularized so that additional digitalised features can be easily added onto it giving me a base to being to develop a digital lean platform.

I have also built a separate cloud mongo database and deployed the Web app using and AWS ec2 instance.

# Works Cited

Capan, T. (2016). *Why The Hell Would I Use Node.js? A Case-by-Case Tutorial*. Retrieved from www.toptal.com: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

Deshpande, C. (2021, 09 14). *What is Angular?: Architecture, Features, and Advantages*. Retrieved from www.simplilearn.com: https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular

Kayere, P. (2020, 07 17). *Introduction to hapi.js Framework*. Retrieved from www.section.io: https://www.section.io/engineering-education/introduction-to-hapi/

Lvivity. (2018, 03 15). *Waterfall Methodology: Advantages and Disadvantages*. Retrieved from lvivity.com: https://lvivity.com/waterfall-model

Malik, H. (2018, 01 05). *Introduction to Hapi.Js*. Retrieved from simpleprogrammer.com: https://simpleprogrammer.com/introduction-hapijs/

Mary K. Pratt & Christina Torode. (2020, 04). *Agile Manifesto*. Retrieved from searchcio.techtarget.com: https://searchcio.techtarget.com/definition/Agile-Manifesto

Matviiok, T. (2021, 04 16). *7 Reasons to Develop Your Next Web App With Meteor*. Retrieved from keenethics.com: https://keenethics.com/blog/7-reasons-to-develop-meteor-apps

North, M. (2015). *What Is a Web Database?* Retrieved from www.techwalla.com: https://www.techwalla.com/articles/what-is-a-web-database

Pandit, N. (2021, 02 10). *What and Why React.js*. Retrieved from www.c-sharpcorner.com: https://www.c-sharpcorner.com/article/what-and-why-reactjs/

SANTOS, J. M. (2021, 08 21). *Agile vs. Waterfall: Differences in Software Development Methodologies*. Retrieved from project-management.com: https://project-management.com/agile-vs-waterfall/

Sergeev, A. (2016, 06 09). *Waterfall vs. Kanban*. Retrieved from hygger.io: https://hygger.io/blog/waterfall-vs-kanban/

Stodulski, F. (2020, 01 15). *7 Benefits of Angular You Should Know if You Want to Build a Digital Product*. Retrieved from www.netguru.com: https://www.netguru.com/blog/benefits-of-angular

Sufiyan, T. (2021, 06 03). *What is Node.js: A Comprehensive Guide*. Retrieved from www.simplilearn.com: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs

Tutorial Point. (2019). *Tutorial Point*. Retrieved from www.tutorialspoint.com: https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm