

Project Design and Specification Document



Niamh Gilligan

National University of Ireland Galway

Supervisor: Dr.Owen Molloy

April 2018

0.1 Acknowledgements

I would like to thank Dr.Owen Molloy for his support and advice throughout the development of this project.

I would also like to thank Ciaran O'Toole , Weston Fleming and Liam Shaw from SmartBear for taking the time to meet with me and show me where to begin learning React.

Contents

0.1	Acknowledgements	2
1	Introduction	5
1.1	Document Overview	5
1.2	Document Description	5
1.2.1	Introduction	5
1.2.2	Overview of Industry Partner	5
1.2.3	Application Details	5
1.3	Requirements elicitation	6
1.4	System Overview	6
2	Design Considerations	7
2.0.1	General Constraints	7
2.0.2	Assumptions	7
3	System Architecture	8
3.0.1	Architecture Strategies - MERN	8
3.0.2	Client Architecture	11
3.0.3	Server Architecture	16
4	System Development	18
4.1	Project Timeline and Key Challenges	18
4.2	Learning Outcomes	20
4.3	Development Methods	20
4.3.1	Method Adopted.	20
4.3.2	Weaknesses	20
4.3.3	Strengths	21
4.4	Compromises	21
4.5	Extensions	21
5	Detailed System Design	22
5.1	Use case Documentation	22
5.2	Code Breakdown	25

5.2.1	Register / Log-in / Log-out :	25
5.2.2	Search Bar	26
5.2.3	Navigation Bar	27
5.2.4	Creating a Lesson Plan.	28
5.2.5	Sending a message.	28
5.2.6	Signing up to email subscription	29
5.2.7	Delete / Edit buttons	29
5.2.8	Security	29
6	System Test Design	31
6.1	Testing Overview	31
6.2	White Box Testing	31
6.2.1	API testing	31
6.2.2	Test Framework- Enzyme	33
6.2.3	Enzyme Testing :	33
6.2.4	Snapshot Testing :	34
6.3	Black box Testing	36
6.3.1	Detailed Test Cases	36
6.3.2	User Evaluation	46
	Bibliography	47
.1	Appendix	48
.1.1	Learning Journals	48
.1.2	Requirements Document	53

Introduction

1.1 Document Overview

This document presents a detailed overview of the Design , Implementation and Testing phases encountered throughout the development of this application. This document has been created to give the reader a comprehensive understanding of the project undertaken and detailed breakdown of the software utilised to develop this application prototype.

1.2 Document Description

1.2.1 Introduction

1.2.2 Overview of Industry Partner

SmartBear equips teams with the tools to ensure quality at every stage of the software cycle. With products for API testing, UI testing, code review and performance monitoring across mobile, web and desktop applications. I will be working at the SmartBear European Headquarters located in Galway city. Here I will be working with the Front-end developer team on a product called SwaggerHub. SwaggerHub is a platform for designing and documenting API's.

[1]

1.2.3 Application Details

Task : Since I will be working on the front-end team in SmartBear, the main technologies I will be working with are React, css pre-processors and Javascript. My challenge was to develop my understanding of these technologies and design an application which utilises these technologies. Having no prior knowledge of React , css pre processors and a basic Javascript understanding. This was quite the challenge for me.

Application Goal : The goal of this project is to develop a single page web application called "Teachers Toolbox". Teachers Toolbox provides a quick , simple to use web application where a user can upload , search and download Lesson plans. Users can also request specific lesson plans and sign up to a mailing list for the latest lessons. The main concept behind Teachers Toolbox is to help teachers and students save time when planning as is reflected on the homepage 'Search , Share and Save time'.

1.3 Requirements elicitation

From my time as a primary school teacher I have gotten to know quite a few teachers and I reached out to them to discuss my idea. The feedback I received was valuable for the requirements stage of my project. Here is some of the feedback I received.

'I think it it would be handy if the website had all the subjects and teachers could click on them and then it would have the lesson and resources for the lesson. '

Rachel 27, Primary school teacher Galway.

'It would be great if you could search for lessons even by just saying the resources you have.'

Siobhan 25, Primary school teacher Dubai.

'I would love a simple free website to share lessons and get them quickly too.'

Jessica 32, Primary school teacher Galway.

From my online research I have found that there are not any free websites where you can search using just resources/title/description/class level , upload and download both Irish and English Lesson plans. I believe this would have been very useful for me when I was teaching.

1.4 System Overview

Teachers Toolbox is a full-stack MERN application. MERN uses MongoDB as the database , Express and Node.js on the back-end / server and React on the front-end creating the UI for the application. I am also integrating webpack which is explained in more detail in the System architecture section. An Agile approach was used to develop this application as I created different features incrementally and iteratively. Also re-factoring of code was essential throughout my project.

Main System Features :

- User Authentication Register and Login functionalities.
- User cannot view certain pages unless they are logged in.
- User can delete any Lesson they uploaded.
- User can Logout of a session.
- User can edit title of any Lesson they uploaded.
- User can upload/download a Lesson Plan.
- User can send contact form which is stored in the database.
- User can send their email to a mailing list which is also stored in the database.
- Comprehensive search functionality.
- Password encryption using bcrypt.
- Snapshot testing.
- If page is not found a Page is not found message will appear with a link back to the homepage. eg: if `http://localhost:8080/creattte` was entered or any link with a mis-spelling.

Chapter 2

Design Considerations

2.0.1 General Constraints

General constraints Identified :

- It must be able to search through a large number of lesson plans.
- It must be secure.(Only logged in users may send messages/upload /download /delete and edit)
- Must be available over the internet.
- It must have functionality to upload and store Lesson plans. -It must be User friendly.
- Time was one of the main constraints for this project as the MERN stack was new to me I found it difficult to estimate how much I could complete in the 12 week time frame.
- React needed to be one of the main technologies used. React is a javascript library for building user interfaces , so in order to make an functional application I needed to add more technologies to make it a full-stack application. I researched which technologies are best to use with React and the MERN stack seemed like the obvious choice. This meant I had to learn all of these individually and understand how they work together.

2.0.2 Assumptions

- Assumptions for this application are that a user would have access to the internet and have an email address.
- I also assumed the amount I could learn and what I could produce in the 12 week time frame.

System Architecture

3.0.1 Architecture Strategies - MERN

[2] [3] [4] For this project I have chosen to use the MERN stack. My core project foundation is MERN but I also have included webpack bundler. I realise this is not essential as I have created a relatively small project. However, it is important if I were to build a larger scale so I thought it would be best to include it. I also integrated bootstrap into my project to give an enhanced user experience.

ReactJS – Rise of the MERN Stack

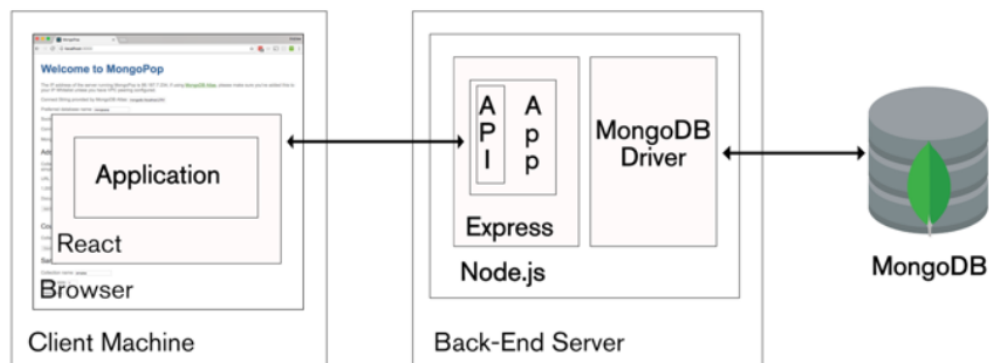


Figure 3.1: MERN Stack [5]

MongoDB:

MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. I have been working with relational databases before this so I thought it would be interesting to show how MongoDB differs from these. MongoDB stores its data into collections of JSON documents. The schemas define related data. In figure 3.2, we can see on the left how data is stored in MongoDB compared to relational database tables on the right. Instead of a primary key used in relational databases MongoDB uses a unique identifier for each set of data stored

ie:Every Lesson plan saved in MongoDB has its own unique identifier.

I am running MongoDB locally and using mongoose to connect to the database and create schemas. I will explain this in detail in the server architecture section. I am also using Robomongo as a MongoDB GUI as I find it fast and easy to work with.

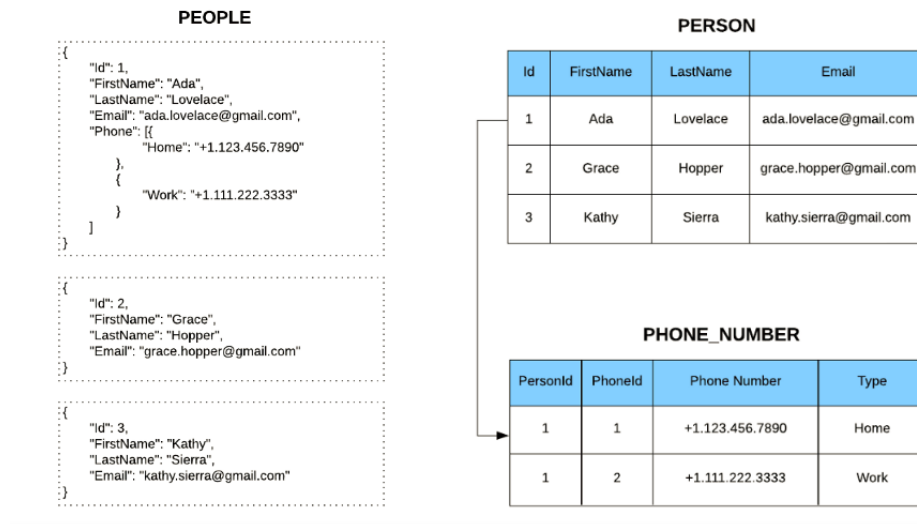


Figure 3.2: MongoDB vs Relational Database[5]

Webpack :

'At its core, Webpack is a static module bundler for modern JavaScript applications.' Webpack is a build tool that puts all of your assets, including Javascript, images, fonts, and CSS, in a dependency graph which describes how modules relate to each other and packages them into one bundle to be loaded by the browser. For small applications it not essential to have webpack but for larger applications it is extremely useful. Webpack is commonly used with React so I thought it would be beneficial for me to include this in my project. An advantage to Webpack is that it can act as a code splitter so that it can modularize sections of code and use it only when it is needed. This means faster load time and better performance for applications. [6]

Express:

Express is a back-end web application framework running on top of Node.js. Express runs as a module within the Node.js environment. In this application Express is used to send the front-end application code to the browser when the application is running. Express also provides REST API's that the front-end can access using HTTP network calls.

[5]

Node.js

Node.js is an open-source server framework. It is a JavaScript runtime environment that lets you implement your application back-end in JavaScript. Node.js is an asynchronous, event-driven engine where the application makes a request and then continues working on other tasks rather than waiting for a

response. Node.js eliminates the waiting, and simply continues with the next request. The Application is alerted when the task is completed by a callback. This means that large numbers of operations can be performed in parallel which is essential when scaling applications. MongoDB was also designed to be used asynchronously and so it works well with Node.js applications.

[5]

Bootstrap styling and SASS

Bootstrap styling was used through the application in conjunction with SASS. I used SASS CSS pre-processor in this application. Using CSS pre-processors has lots of advantages for a developer as it allows you to use variables, makes your CSS more modular and helps you to write less code by making elements re-usable.

React :

React anchors the MERN stack. It is described as the defining component of the MERN stack. React is a JavaScript library developed by Facebook to build interactive/reactive user interfaces. React is often called the 'V' in the (MVC) Model-View-Controller. React allows you to create re-usable components. For example in my project I have created an EmailBox which is made up of three components EmailBox, NameField and EmailField. Namefield and Email Field control their own state and as individual components they can easily be re-used. React allows HTML to be embedded within the code.

```
if (!token) {  
  return(  
    <div><h1>Please sign in to view Create Lesson Page.</h1></div>  
  );  
}
```

Figure 3.3: HTML embedded in React component.[5]

React is also commonly written in JSX. You can embed any JavaScript expression in JSX by wrapping it in curly braces. A parent component can pass its state down to its child components and those components can pass changes back to the parent through the use of callback functions. I have used this in my project as you will see in the React-components section.

React Router: React Router is the standard routing library for React. This is the most popular router for React. I have used this in my application for navigating between different components. React router is considered dynamic routing as the routing takes place as the app is being rendered. [7]

React - State vs Props : Props(short for properties) and state are very important concepts in React. Props are passed into a component similar to how an argument is passed to a function.Props should not change. State is similar to props in that it also holds information.State can be changed and is initialised within the component. A component can create and update state. You will see state and props throughout the components in my application.

3.0.2 Client Architecture

The Client folder contains all the front-end code for the UI. This folder contains all the React components , HTTP requests, styling , images and fonts that create the UI.

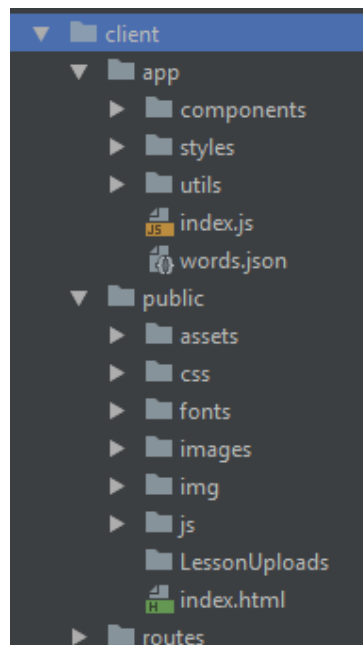


Figure 3.4: Client Folder.

[7]

React - Components

Babel is a tool that helps you write code in the latest version of JavaScript. When your supported environments don't support certain features, Babel will help you compile those features down to a supported version. I have used ES6 syntax in this application and Babel is used to transpile this so the browser can interpret this.

In React it is considered good practise to break up components into smaller components so that data and state changes can occur logically. It also keeps code clean. Ideally when creating react components you should follow the single responsibility principle which means each component should ideally be responsible for one task. For example in my email form as you can see in the picture below is broken into three components. [7]

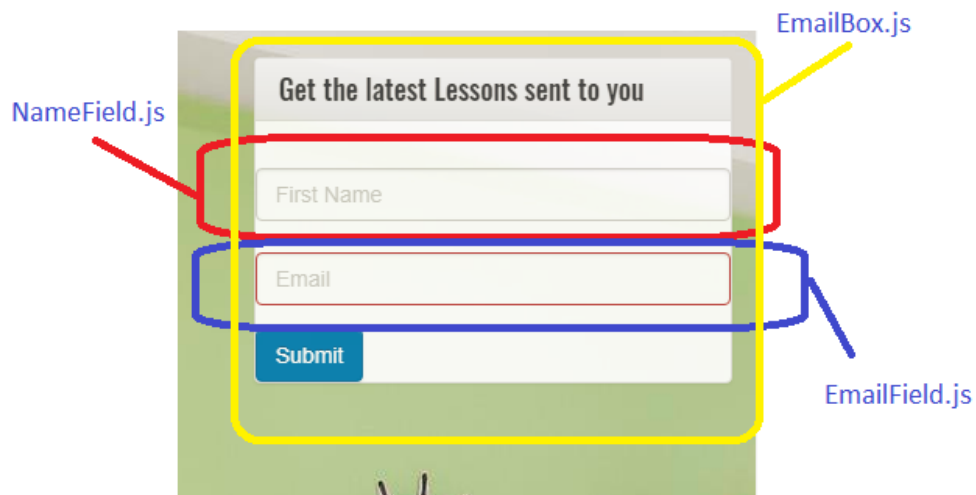


Figure 3.5: Components

React - API /HTTP requests

There are many ways to connect React front-end API requests to the back-end requests on the server. The ways I chose to use are axios, Fetch and superagent. These are three completely separate methods of carrying out the same function. I used a variety so that I could have a good knowledge of these and broaden my understanding of how they function. Here I have an example of each from my project code. Axios is a HTTP client for JavaScript which works well with react. The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses. Superagent can provide the same functionality as Fetch and axios but in only a few lines of code. It is a client side HTTP request library.

```

return (
  <div className="col-sm-3" style={boxStyle}>
    <div className="panel panel-default">
      <div className="panel-heading">
        <h4>Get the latest Lessons sent to you</h4>
      </div>
      <div className="panel-body">
        <NameField type="First" ref="fieldName" />
        <br/>
        <EmailField ref="fieldEmail" />
        <div className="row">
          <div className="col-sm-6">
            <button className="btn btn-primary" onClick={this.onSubmit}>
              Submit
            </button>
          </div>
          <div className="col-sm-2">
            <h5 style={successStyle}>Success</h5>
          </div>
        </div>
      </div>
    </div>
  </div>
)

```

Figure 3.6: Importing NameField and EmailField into EmailBox.

Here is an example of how the email and name components are imported into the parent component in this case EmailBox. Email box controls the post request which sends the data it receives from email and name fields. The main responsibility of email field is to validate the email and save the state when a user inputs their data. The emailfield saves the state and passes this to the emailbox. The same happens with the namefield.

```

// super agent post request where request is sent to server
request.post('/api/account/emailsignup')
//url for post request matching server post request url
//.send sending the users input from email and firstname
.send({ email: this.refs.fieldEmail.state.value })
.send({ firstName: this.refs.fieldName.state.value })
.then(callback);

```

Figure 3.7: Superagent API request in EmailBox.js.(client/app/components/Forms/EmailBox.js)

```

    handleDelete(id) {
      console.log(id);
      axios.delete('/api/account/getLesson/' + id)
        .then(res => {
          console.log('Comment deleted');
        })
        .catch(err => {
          console.error(err);
        });
    }
  }

```

Figure 3.8: Axios delete request.(client/app/components/UserHomepage/UserHomePage.js)

```

//fetch request posting to MongoDB
fetch('/api/account/createLesson', {
  method:'POST',
  headers:{
    'Content-Type':'application/json'
  },
  body:JSON.stringify({
    title : setTitle,
    description : setDescription,
    resources: setResources,
    classlevel:setClass,
    subject: setSubject,
    file:setFile,
  })),
}).then(res => res.json())
  .then(json => {
    if (json.success){
      //resetting state to clear fields
      this.setState({
        isLoading: false,
        setTitle,
        setDescription,
        setResources,
        setClass,
        setSubject,
        setFile,
      });
      alert("Congratulations! You uploaded a Lesson!.");
    }
  });
});

```

Figure 3.9: Fetch request for posting lesson plan.(client/app/component/CreateLesson/CreteLesson.js)

```

app.post('/api/account/createLesson',multer(multerConf).single('LessonPlan'),
  var {body} = req;
  var {
    title,
    description,
    resources,
    classlevel,
    subject,
    file
  } = body;
  //save new user
  var newLesson = new Lesson();

  newLesson.title = title;
  newLesson.description = description;
  newLesson.resources = resources;
  newLesson.classlevel = classlevel;
  newLesson.subject = subject;
  newLesson.file = file;

  if(req.file){
    req.body.file = req.file.filename;
    console.log(req.file);
  }

  newLesson.save((err, lesson) => {
    if (err) {
      return res.send({
        success: false,
        message: 'Error:Server error'
      });
    }
  })

```

Figure 3.10: Back-end request which connecting with Fetch request figure3.(server/routes/api/LessonPlan.js)

3.0.3 Server Architecture

The Server Architecture contains all the mongoose models for the data which will be stored in MongoDB , it also contains the API routes which work with the front-end HTTP requests and the configurations for the application in server.js.

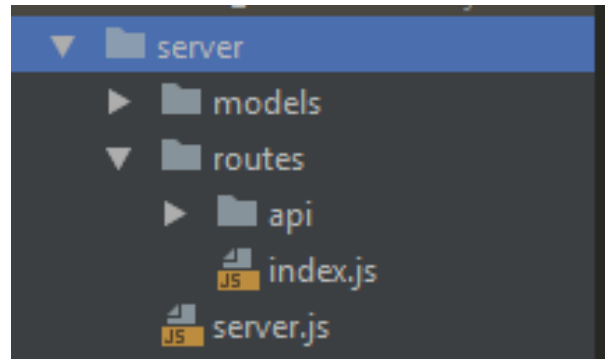


Figure 3.11: Server Folder.

MongoDB Database In conjunction with MongoDB, I am also using mongoose to model my application data. Mongoose is a MongoDB data modelling tool. In my application I created mongoose schemas for my Lesson plans, Users , User session ,contact page details, and email sign up box. These are stored in server/models folder. Here is my schema for a User. I also use bcrypt encryption here to hash the password when it is saved.

```
var mongoose = require('mongoose');
var bcrypt = require('bcrypt');
var UserSchema = new mongoose.Schema({

  firstName: {
    type: String,
    default: ''
  },
  lastName: {
    type: String,
    default: ''
  },
  email: {
    type: String,
    default: ''
  },
  password: {
    type: String,
    default: ''
  },
  isDeleted: {
    type: Boolean,
    default: false
  }
});

UserSchema.methods.generateHash = function(password) {
  return bcrypt.hashSync(password, bcrypt.genSaltSync(8), null);
};

UserSchema.methods.validPassword = function(password) {
  return bcrypt.compareSync(password, this.password);
};

module.exports = mongoose.model('User', UserSchema);
```

Figure 3.12: User mongoose Schema (server/models/User.js)

Robomongo

I am using Robomongo to view the data in my database as I find it excellent as GUI for that purpose. Here are my collections based on my mongoose schemas. Any data sent to the database will show up here. MongoDB assigns a unique ID to the data sent to it and this is what we use to uniquely identify individual users / lesson plans in order to retrieve them from the data base to edit or delete.

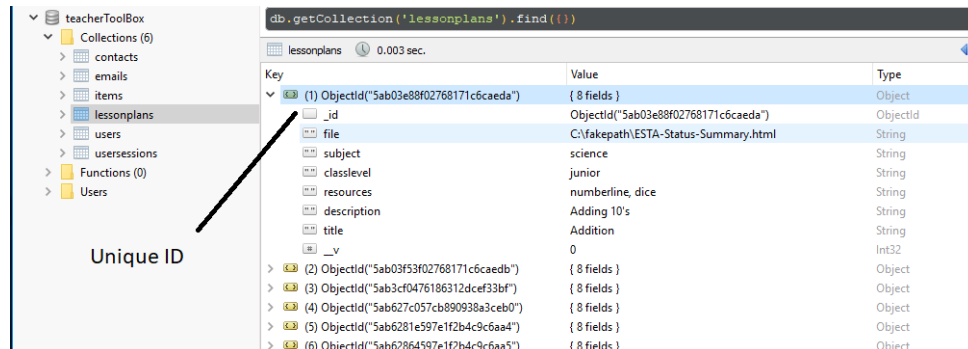


Figure 3.13: Robomongo-My application collections.

Chapter 4

System Development

This section documents the project timeline and key challenges encountered. It also describes methods used and identifies strengths and weaknesses to the approach taken.

4.1 Project Timeline and Key Challenges

The project was completed over a period of 12 weeks. Details of the tasks I undertook each week and issues that occurred can be found in Appendix 1 Learning Journals. These Learning Journals outline the steps it took me to get to this point.

Starting point - No experience with React. Basic introduction to Javascript, MongoDB , Node.js and Express from semester one.

Key challenges and how I overcame them :

Challenge 1: Javascript , React and the MERN stack. One of the first key challenges for me was understanding the technologies and how best to use these in my project. I had a basic understanding of Javascript from semester one and no previous experience of React. This was a major learning curve for me. I immersed myself in Javascript and React tutorials. I also completed a React course on Udemy which I found very helpful. Throughout my research it was clear that the MERN stack was best suited to my project and it would allow me to fulfill my project requirements. This meant I also had to get to grips with MongoDB , Express and node.js. The majority of my problem solving involved reading the latest documentation on the technologies I was using , exploring similar issues on GitHub and watching tutorials. This project was a huge challenge for me and I am proud of how much I learned throughout this experience and of the work I have produced.

Challenge 2: Which technologies and methods to use ? Choosing certain technologies and methods was certainly a challenge for me. I found that there are so many alternatives to carry out the same function that at times it was difficult to know which was the best option. I overcame this by trying out a few methods for myself. For example, In React I found three ways of sending requests.e.g: to send a delete request I could use Axios , Fetch or Superagent. I tried each of these in my project and have included the code which you will find in the code break down section. Also with my file upload , I found there are so many different ways to approach this function that it is difficult to know which is most appropriate for

your situation. I ended up using Multer which I discovered works well with Express and Node. My search bar was another example of this. I found that there were many ways to approach this task and difficult to know which to choose.

Challenge 3: Out-dated material. Throughout this project one of the main challenges I found was outdated tutorials or information due to new releases of software. This was the cause of many headaches and delays. I overcame this by always going to the latest documentation for the chosen technology. For example in React there has been a very recent change in the way you create classes. 'React.createClass' was the old syntax and I discovered that for the classes to work now you need to install the createReactClass package and use this instead.

```
import Callback from 'superagent';  
import createReactClass from 'create-react-class';  
  
var EmailBox = createReactClass({
```

Figure 4.1: Create React Class.

Challenge 4: MERN - working together. Another aspect I found challenging was getting all the technologies to work together in my case MongoDB, express, react and node. I first had to learn about these individually. Then when it came to putting all the pieces together it took some problem solving. I got the Server side (MongoDB, Express and Node) to work together initially. I first got the server side API's to work and tested these using postman. I also got the client side react / bootstrap / react router working all together. It was challenging when I had to connect the API requests from server side to client side. These took a lot of time and effort to ensure they were working correctly. Tutorials and documentation really helped me to overcome this.

Challenge 5: Delete / edit buttons A major issue I encountered was getting the Object ID for each individual lesson which MongoDB adds to each Lesson saved to the database. The issue was that it is stored as an object. This meant I needed a way to just get the ID(unique set of numbers and letters in the object). I attempted a few different ways of completing this but found in React I could use (this.props.obj-id) to get this unique ID and send this in the API requests on the client side to the server side requests where this could be used to identify the specific lesson to be deleted/edited.

_id	ObjectId("5ad1eb0ace94921f8094d456")	ObjectId

Figure 4.2: ObjectID as stored in MongoDB.

Challenge 6: Security Adding security measures was challenging for me. I wanted to ensure a user had to be logged in to view certain pages and use certain functions. I first tried this using jwt web tokens but had issues implementing this. I decided to give a user a token(stored as a cookie) each time they log-in or register. When a page is rendered the method componentDidMount is automatically called so I decided to put an API request here to verify if there is a token present. I then used an if statement to determine what will be rendered depending on the presence of a token. If there is a token present the page will be

rendered but if no token is present a message to log in will be rendered.

Challenge 7 : Configurations I spent a lot of time configuring dependencies. For React I needed to have Babel and to use this it needed to be configured. Also I wanted to add Bootstrap to my project and discovered the best way to do this was to keep the files needed in my project instead of creating links to the files which would take longer to load. These files are imported into my index.html so that they are available for each component. Jest testing also required configurations in order for it to work with my project. Enzyme adapter needed to be configured for the testing. SASS pre-processor also required certain configurations. I found these configurations very time consuming as it took a lot of learning and researching. I overcame these challenges by reading the documentation carefully and watching tutorials.

Challenge 8 : File upload / download. The file upload was a challenge for me as when it is uploaded the file name needs a unique identifier in case two lessons with the same name were uploaded. For the download I was having trouble finding the path name to the original file. MongoDB hides the file path for security reasons. I found it a challenge to find the new name of the file to download it. Given more time I feel I could have worked this out more professionally.

4.2 Learning Outcomes

This has been a very challenging project for me but I am proud of the amount of challenges I managed to overcome and the amount of knowledge acquired in such a short space of time. Given the restricted schedule for delivering the application learning and implementation occurred in some-cases simultaneously. I understand I have so much more to learn but I feel this has given me the foundation I need to prepare me for my internship.

4.3 Development Methods

4.3.1 Method Adopted.

The MERN stack is the method adopted for this project. In this section we will look at weaknesses and strengths to using this approach.

4.3.2 Weaknesses

When it comes to heavy computation and data processing on the server side, Node.js is not the best option. However in relation to this project I feel node was a good choice due to the relatively small amount of data processing required. If given more time I would have broken up some of the react components to smaller re-usable components. I implemented this with parts of my project but I would have liked to explore this more.

4.3.3 Strengths

Throughout my research into react, I learned that React has a virtual DOM for every DOM object. The virtual DOM contains the same properties as the real DOM but doesn't have the control to change what is rendered on screen. This is why I would consider using react again in future projects. Manipulating this virtual DOM is very fast as nothing is rendered. React uses this virtual DOM to see what has been changed and updates the DOM accordingly making it very efficient. Node.js uses event-driven, non-blocking IO model that makes it fast and lightweight when compared to other commonly used back end technologies.

4.4 Compromises

While the majority of the applications core functionality was implemented there are some aspects I would have liked to develop further. Given the time frame I feel I divided my time well to ensure a well rounded understanding of the development process from start to finish. I could have spent more time on functionality and security but I chose to invest time learning about the different types of testing.

4.5 Extensions

My project has changed and developed a lot since the requirements document. The main idea is similar but I was unaware of the amount of technologies, frameworks and dependencies I would encounter along the way. I used babel, webpack, superagent, fetch, multer, axios, bcrypt to name a few. I also integrated bootstrap to help with the UI design. For my testing I used a mix of white box and black box testing. The white box testing included Jest and enzyme which I initially had not planned to use. There are more design features which I would add if given more time, I would also address security issues and add more testing to my application. Possible extensions would include adding a forum so that users can comment on Lesson plans and share ideas.

Detailed System Design

5.1 Use case Documentation



Figure 5.1: Use Case Diagram

Use Case 1 : Register / Login

Actor : Teacher / Student

Purpose:

Register :To allow a user to enter their name , email and a password which will be saved to the database and allow them access to the website.

Login: Allows user to enter their password and email to allow them access to the website.

Scenario:

- 1.Actor logs on to Teachers Toolbox and given option to Login or Register.
- 2.Actor chooses login or register.
- 3.Actor inputs their details.
- 4.Actor is now able to access all parts of the website.
- 5.Actor can Logout using logout button.

Use Case 2 : Search for Lesson.

Actor : Teacher / Student

Purpose:

To allow a user to search for a lesson plan by title, description, resources , class level or subject.

Scenario:

- 1.Actor logs in to Teachers Toolbox
- 2.Actor clicks on search in the navigation bar.
- 3.Actor inputs what they want to search.
- 4.Results will be shown and actor has option to download the lesson.

Use Case 3 : Upload a Lesson.

Actor : Teacher / Student

Purpose:

To allow a user to upload a lesson plan with title, description, resources , class level , subject and file attributes.

Scenario:

- 1.Actor logs in to Teachers Toolbox
- 2.Actor clicks on create in the navigation bar.
- 3.Actor inputs what they want to upload and attaches a file.
- 4.Actor clicks upload and is redirected back to their account page.

Use Case 4 : Delete/Update a Lesson.

Actor : Teacher / Student

Purpose:

To allow a user to delete any of their lessons or edit the title of any lessons they have uploaded.

Scenario: Delete

- 1.Actor logs in to Teachers Toolbox
- 2.Actor is directed to their account page.
- 3.Actor clicks delete button.
- 4.Lesson plan is deleted.

Scenario: Edit

- 1.Actor logs in to Teachers Toolbox
- 2.Actor is directed to their account page.
- 3.Actor clicks edit button.
- 4.Actor redirected and given option to edit title.
- 5.Actor enters new title and clicks update.

Use Case 5 : Send Message.

Actor : Teacher / Student

Purpose:

To allow a user to send a message to Teacher Toolbox staff either to request a lesson, issues they are having or with any question.

Scenario:

- 1.Actor logs in to Teachers Toolbox
- 2.Actor clicks on contact page.
- 3.Actor enters details of their request.
- 4.Actor clicks send and message is sent to the data base.

Use Case 6 : Download Lesson.

Actor : Teacher / Student

Purpose:

To allow a user to download any lesson plan.

Scenario:

- 1.Actor logs in to Teachers Toolbox
- 2.Actor clicks on search page.
- 3.Actor clicks download on a chosen lesson.
- 4.Download starts immediately.

Use Case 7 : Sign up to receive emails.

Actor : Teacher / Student

Purpose:

To allow a user to send their email to sign up for an email list.

Scenario:

- 1.Actor logs on to Teachers Toolbox
- 2.Actor enters their details into the email form.
- 3.Actor clicks send and their data is sent to the database .

5.2 Code Breakdown

My code is split into 2 main folders Client and Server as shown in the System Architecture section. These two folders keep the front end and back-end code separated. In this section I will go into detail of how I created some of the main functionalities of this project and some of the logic behind them.I have included some small snippets of the code written for these elements.

5.2.1 Register / Log-in / Log-out :

For Register, Log-in and Log-out there are Client and Server requests which work together to provide these functionalities. I have provided just small snippets of the code involved.

Register : Emails are checked upon registering to ensure emails are unique to each user using mongoose query `User.find`. The new Users data is posted using a fetch API call to the `/signup` URL which sends the data to the server.If this is a success a message alerts the User. In the server side code there are checks to ensure none of the data is blank. An access code is given to a User that is successfully logged in. There is a mongoose schema for Users which details the data to be stored to the database. For each user a first name ,last name, email , password(hashed) is stored in MongoDB along with a unique identifier which MongoDB assigns.

```
//verify email and save
User.find({
  email: email
}, (err, previousUsers) => {
  if (err) {
    return res.send({
      success: false,
      message: 'Error:Server error'
    });
  } else if (previousUsers.length > 0) {
    return res.send({
      success: false,
      message: 'account taken'
    });
  }
}
```

Figure 5.2: Ensuring emails are unique.

```

fetch('/api/account/signup', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    firstName: signUpFirstName,
    lastName: signUpLastName,
    email: signUpEmail,
    password: signUpPassword,
  })
}).then(res => res.json())

.then(json => {
  if (!json.success) {
    alert('Email already taken. Please try a different email.');
```

Figure 5.3: Fetch method for registering Users.

Log-in: When a user enters their details to log-in , their email is used to see if it is stored in the database and password is hashed and compared to hashed password stored in mongoDB if these match the User is given an access token, a user session is created and stored in mongodb.

Password The password is encrypted using bcrypt and you can see below how it is stored in MongoDB.

password \$2a\$08\$mmFAHuGbmJoreFnF/Ls2i.tf/QcBXhDgxSVK8Ar.ly... String

Figure 5.4: Password encryption.

Log-out: When the log-out button is clicked, the specific usersession is found in the mongoDB database using a mongoose query `UserSession.findOneAndUpdate` . The query is used to update the `isDeleted` attribute to true and the token is set to null.

5.2.2 Search Bar

For my Search component I used react-search-input package. I re-used the `get/lessons` request to get all lessons in the database. I imported `SearchFilter` and create filter from this. I created a filtered search which then is applied to the javascript map function so that when a user enters characters into the search bar it checks these against title, description, resources, class-level and subject ensuring they are all search-able. I used the `onChange` React function to call the `searchUpdated` method each time there is a change to the input in the search box by updating the state of the variable `searchTerm` each time input is changed. Search Bar is not case sensitive.

```
const filteredSearch = this.state.lessons.filter(createFilter(this.state.searchTerm, KEYS_TO_FILTERS));
```

Figure 5.5: Creating Filter.

5.2.3 Navigation Bar

The Navigation Bar was created using bootstrap and React router . Using the Link tag I specified which route to render when a user clicks on it.Each navigation element is a list item so it can be easily added to or edited. The index.js controls the rendering of each component to which URL as shown below.

```
return (
  <div >
    <nav style={navStyle} className = "navbar navbar-default">
      <div className="navbar-header">
        <Link style={titleStyle} className="navbar-brand" to="/">Teachers Toolbox</Link>
      </div>

      <div className = "collapse navbar-collapse" id="nav-collapse">
        <ul className="nav navbar-nav" >

          <li><Link to="/contact" > Contact </Link></li>
          <li><Link to="/create"> Create </Link></li>
          <li> <Link to="/Login"> Login </Link></li>
          <li><Link to="/Register"> Register </Link></li>
          <li><Link to="/MyAccount"> My Account </Link></li>
          <li><Link to="/search"> Search </Link></li>

        </ul>
      </div>
    </nav>
  </div>
)
```

Figure 5.6: Navigation Bar.

Rendering: The Navigation bar creates the links to the URL's for each component but it is in index.js where the components are rendered. Each component is imported into index.js and the pages are specified using the component attribute. React Router renders these components and outputs them to the app div in index.html file using document.getElementById('app') as seen in the code snip below.

```
render((
  <Router>
    <App>
      <div>
        <Switch>
          <Route exact path="/" component={Home}/>
          <Route path="/Login" component={Login}/>
          <Route path="/create" component={CreatLesson}/>
          <Route path="/contact" component={Contact}/>
          <Route path="/Register" component={Register}/>
          <Route path="/MyAccount" component={IndexItem}/>
          <Route path="/search" component={SearchNew}/>
          <Route path="/edit/:id" component={EditItem} />
          <Route component={NotFound}/>
        </Switch>
      </div>
    </App>
  </Router> ), document.getElementById('app'));
```

Figure 5.7: Index.js - React Routing.

Layout : Index.html contains all the script files and links to certain resources required for the application. This will then load all of these for each page as all components are rendered in the app div in this file. Index.html also contains the footer which appears at the end of end page rendered which is styled using inline styling. However, the Layout of each component is controlled by the App component . The App component contains the navigation bar and the EmailBox which appear on each page. Since it is in the App.js it saves code as it is not necessary to import these components individually to each page.

5.2.4 Creating a Lesson Plan.

A mongoose schema is created for creating a lesson plan which includes a title, description, resources, class-level , subject and file. Then this schema is used in LessonPlan.js (Server-side) to create an api post request which will save the data to mongoDB and redirect the user to My account page. On the Client side in components/CreateLesson/CreatLesson.js , this is where the code for the create lesson component is kept. The user input is saved as the state for each variable and it is this input that is sent. The onChange function ensures that each time a User changes the input data that the state is immediately updated. When a User clicks upload the Fetch api post request sends the data to be saved in the data base to the server side request which saves the data to MongoDB.

File upload:

The name of the file path is stored in MongoDB but not the entire path for security reasons. The actual file is stored in the upload folder .The file is uploaded using middleware called multer file upload which works with express. In the code snippet below the file size is restricted for safety reasons , also the file name is set. I chose to use the time uploaded as a unique identifier in case two files uploaded have the same name.

```
const multerConf = {
  storage: multer.diskStorage({
    destination: function(req, file, next) {
      next(null, './upload');
    },
    limits: { fileSize: 1000000 },
    filename: function(req, file, next) {
      next(null, file.fieldname + '-' + Date.now() + path.basename(file.originalname));
    }
  })
};
```

Figure 5.8: Multer file upload.

5.2.5 Sending a message.

A mongoose schema was created(Contact.js) which includes name ,email , mobile(int) and message fields.

Client side : Contact form was created using bootstrap. The onChange method is used again in this component to ensure that the variables which were created to store the input data are updated when a User changes their input. When the send message is clicked the onSubmitForm method is called and this contains the Fetch post method which sends the data to the server side.

Server side: The server side code includes validation checks to ensures no fields are empty and that the

mobile number is an integer. If the data passes the validation a new contact is created and saved to the database.

5.2.6 Signing up to email subscription

A mongoose schema was created (emailSignup.js) which contains firstname and email fields both of type String.

Client side : The Email subscription contains 3 components, Emailbox, EmailField and EmailName. EmailField contains the logic for the email box. The onChange method checks the email to see if it is valid using the email-validator package when the email is correct the email box changes from red to blue. onChange also updates the state of the users input. NameField contains the code for updating the state of the name field. The EmailField component and the NameField component are imported into the EmailBox component and that is the one that is rendering in index.js. EmailField contains the bootstrap code for the email box and the api request (using superagent) which posts the data to the server when Send is clicked.

Server side : The data is validated to ensure it is not null. A mongoose query is used to check if the email is already stored in the database. If it is not in the database a new email sign up user is created and saved to the MongoDB.

5.2.7 Delete / Edit buttons

The edit and delete buttons use the unique identifier created by MongoDB to find specific Lesson plans to either delete / edit them. These functionalities are used in the My account page. However, the logic is separated into the tableRow component and the API requests are in the Item Service class. It is good practise to keep the code clean by separating these, especially for larger applications.

Delete : When the delete button is clicked the object ID for the lesson calls the delete function in the ItemService class. Axios get request used to send the data(HTTP request) to the server side code where it uses the ID to find the specific lesson plan using the mongoose query findByIdAndRemove. The page is then reloaded.

Edit : When the Edit button is clicked a new component (EditItem.js) is rendered. EditItem allows you to edit the title of the Lesson Plan. The User enters the new title into the input box and the onChange function sets the state of the new title. The handle Submit is called when the update button is clicked and this calls the update method in ItemService with the new title and the unique objectID for the lesson that was requested to be updated. The axios post request is used to send the new title and the unique ID to the server side. The Lesson is found using mongoose query FindById and the new data is saved to the MongoDB using the mongoose .save function.

5.2.8 Security

Users have to log in to see and use certain functions. To ensure this I have used a token system. An access token is assigned when a user logs in. The token is saved as a cookie using the res.cookie() method in express.js.

On the client side when a user clicks on the navigation to the create , search , contact or My Account page their token is verified using the method below.If there is no token present the page is not rendered and a message is rendered instead informing them to log in to view this page. When the token is present the page is rendered. Get from storage method - used to check if key and object present.

```
// ...  
res.cookie('Authorization', 'Bearer ' + user.access_token);  
return res.send();
```

Figure 5.9: Saving token in cookie.

```
componentDidMount() {  
  
  const obj = getFromStorage('the_main_app');  
  if(obj && obj.token){  
    const {token} = obj;  
    //verifying token  
    fetch('/api/account/verify?token=' + token)  
      .then(res => res.json())  
      .then(json =>{  
        if(json.success){  
          this.setState({  
            token,  
            isLoading:false  
          });  
        }else{  
          this.setState({  
            isLoading:false,  
          });  
        }  
      });  
  }else{  
    this.setState({  
      isLoading:false,  
    });  
  }  
}
```

Figure 5.10: Code for verifying tokens.Client side.

System Test Design

6.1 Testing Overview

Throughout my project I found myself simultaneously coding and testing. Every step of my project involved testing as I went along to ensure everything was working together. I have chosen to use black and white box testing. I had to research and decide which was the best testing frameworks and methods to choose for my React application. I chose to use Jest to run the test and enzyme testing framework to create the tests. I am also using postman to test my API's. I was lucky to convince two colleagues of mine to test the website out and give feedback. Jest was created by Facebook to test Javascript code and also React applications. Jest is used to run the tests while enzyme created by airbnb is a Javascript testing tool specifically for testing React components. Jest works well with enzyme so that is why I have chosen to use them together. [8] [9]

6.2 White Box Testing

6.2.1 API testing

For testing my API calls I used Postman a Google Chrome app for interacting with HTTP APIs. In my project I have many requests eg: GET, POST, DELETE and PUT. Each API I created on the back-end was tested before I moved on to coding the connecting front-end. This helped me to detect any issues with the back-end. I also used Robomongo to see if the data had been sent to the database when using POST requests.

See below an example of a few of my API tests.

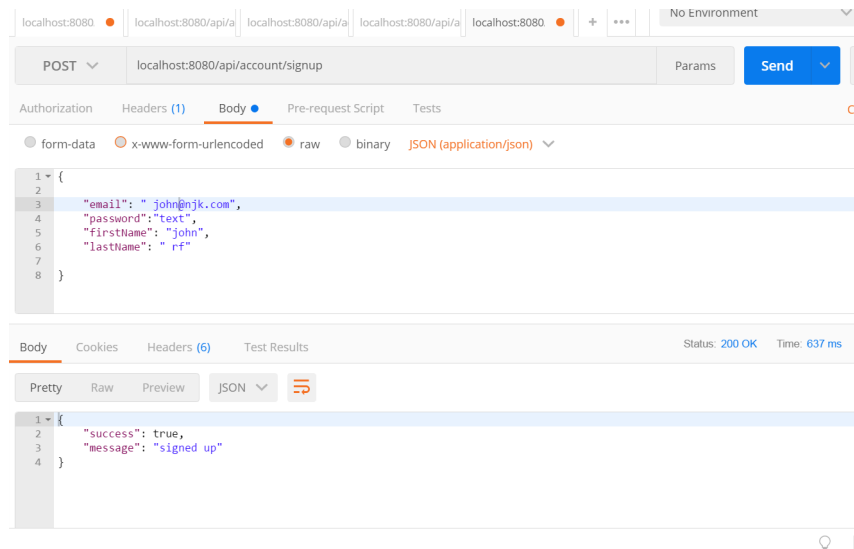


Figure 6.1: Testing Sign up -POST API

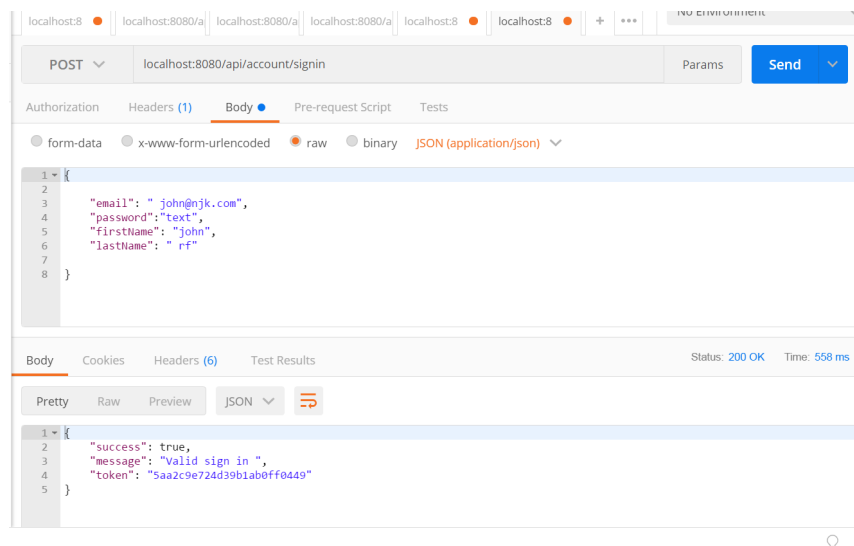


Figure 6.2: Testing Sign in API

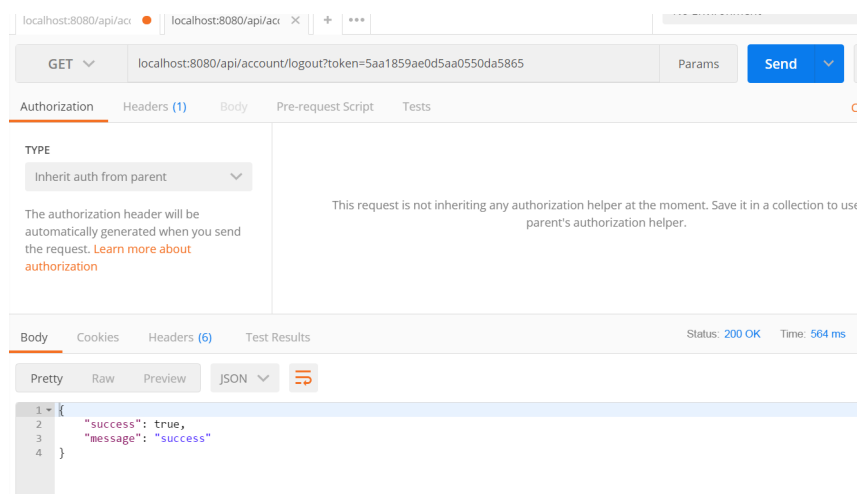


Figure 6.3: Testing Logout API using data sent to Robomongo

6.2.2 Test Framework- Enzyme

Enzyme is a Javascript library for testing React components. Enzyme needs an adapter to work with different versions of React and require certain peer dependencies. For my application I used react-adapter-15. Jest is installed to run the enzyme tests. config/setupTests was created to keep the configuration code. [9]

6.2.3 Enzyme Testing :

In the image below you can see an example of the Enzyme testing I implemented for my Login page. This tests to see if the component renders successfully and to see if it renders and email and password input. The results shown in figure 6.5 also include the snapshot testing results which is explained in the next section.

```
import React from 'react';
import {shallow} from 'enzyme';
import Login from './Login';

describe('Login Test', () => {

  test('Rendering Login page', () => {
    const wrapper = shallow(
      <Login/>
    );

    expect(wrapper).toMatchSnapshot();
  });

  it('renders a email input', () => {
    expect(shallow(<Login />).find('#inputUsername').length).toEqual(1)
  });

  it('renders a password input', () => {
    expect(shallow(<Login />).find('#password').length).toEqual(1)
  });
});
```

Figure 6.4: Enzyme testing.

```
PASS client\app\components\Forms\EmailBox.test.js
PASS src\App.test.js
PASS client\app\components\Login\Login.test.js
PASS client\app\components\Home\Home.test.js
PASS src\UnitTest.test.js
PASS client\app\components\CreateLesson\CreateLesson.test.js
PASS client\app\components\NavBar.test.js
PASS client\app\components\Contact\Contact.test.js
```

Figure 6.5: Enzyme passed tests.

6.2.4 Snapshot Testing :

Snapshots are used to ensure that your UI does not change unexpectedly. A snapshot is taken of a component when the test is run. This snapshot is like a textual screenshot and is compared to the new screenshot when the test is re-run. The snapshot test will fail if the screenshots do not match. I created a snapshot for my contact page component and below you can see the screenshot that was taken. I also changed the code to see if the test would fail and have shown the result in figure 6.7. Snapshot testing is not test driven development and is designed to be written after the component has been created. [8]

```
Snapshot Summary
  > 1 snapshot written in 1 test suite.

Test Suites: 8 passed, 8 total
Tests:       10 passed, 10 total
Snapshots:   1 added, 5 passed, 6 total
Time:        10.91s
Ran all test suites.
```

Figure 6.6: Successful Snapshot test

```
PASS client\app\components\Login\Login.test.js
FAIL client\app\components\Contact\Contact.test.js (5.361s)
  ● Contact Test > Rendering Contact page

    expect(value).toMatchSnapshot()

    Received value does not match stored snapshot 1.

    - Snapshot
    + Received
```

Figure 6.7: Here I changed the code in contact to fail the test.

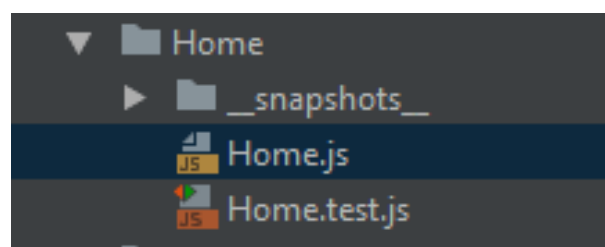


Figure 6.8: Snapshots are stored in the folder of each component.

```

import React from 'react';
import {shallow} from 'enzyme';
import renderer from 'react-test-renderer';
import Login from './Login';

describe('Login Test', () => {

  test('Rendering Login page', () => {
    const wrapper = shallow(
      <Login/>
    );

    expect(wrapper).toMatchSnapshot();

  });

  it('renders a email input', () => {
    expect(shallow(<Login />).find('#inputUsername').length).toEqual(1)
  });
  it('renders a password input', () => {
    expect(shallow(<Login />).find('#password').length).toEqual(1)
  });
});

```

Figure 6.9: Example test code.

```

Login.test.js x Contact.test.js x Contact.test.js.snap x storage.js x
1 // Jest Snapshot v1, https://goo.gl/fbAQLP
2
3 exports['Contact Test Rendering Contact page 1'] = `
4   ShallowWrapper {
5     "length": 1,
6     Symbol(enzyme.__root__): [Circular],
7     Symbol(enzyme.__unrendered__): <Contact />,
8     Symbol(enzyme.__renderer__): Object {
9       "batchedUpdates": [Function],
10      "getNode": [Function],
11      "render": [Function],
12      "simulateEvent": [Function],
13      "unmount": [Function],
14    },
15     Symbol(enzyme.__node__): Object {
16       "instance": null,
17       "key": undefined,
18       "nodeType": "host",
19       "props": Object {
20         "children": <div
21           className="container-fluid"
22           style={
23             Object {
24               "backgroundColor": "#839CF7",
25               "marginLeft": "-250",
26               "opacity": "0.9",
27               "padding": "2em",
28             }
29           }
30         >
31         <form
32           style={
33             Object {
34               "marginLeft": "350",
35               "marginTop": "-50",
36               "padding": "2em"

```

Figure 6.10: Snip of snapshot screen shot of contact component

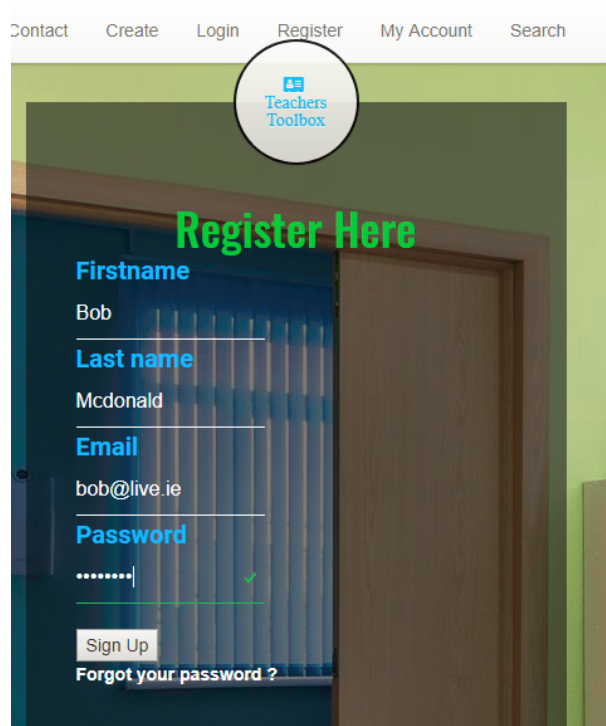
6.3 Black box Testing

6.3.1 Detailed Test Cases

For the blackbox testing I went through all components of my application and you can see the results below. I also was fortunate to have two colleagues of mine try out the application in its prototype state.

Test case 1 - Register User

Here I am testing if a user can register when all details are filled in correctly and also when all fields are not filled in.



The screenshot shows a web application interface with a navigation bar at the top containing links: Contact, Create, Login, Register, My Account, and Search. A circular logo for 'Teachers Toolbox' is positioned in the center of the navigation bar. Below the navigation bar, a large green banner with the text 'Register Here' in bold green font is displayed. Underneath the banner, a registration form is shown with the following fields: 'Firstname' (filled with 'Bob'), 'Last name' (filled with 'Mcdonald'), 'Email' (filled with 'bob@live.ie'), and 'Password' (filled with '.....'). A green checkmark is visible next to the password field. At the bottom of the form, there is a 'Sign Up' button and a link that says 'Forgot your password ?'.

Figure 6.11: Register form.

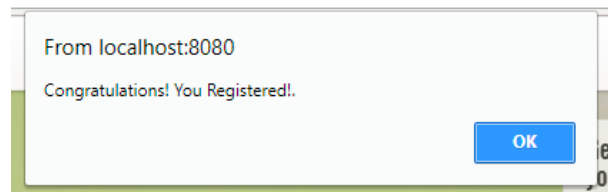


Figure 6.12: Message appears when user is registered.

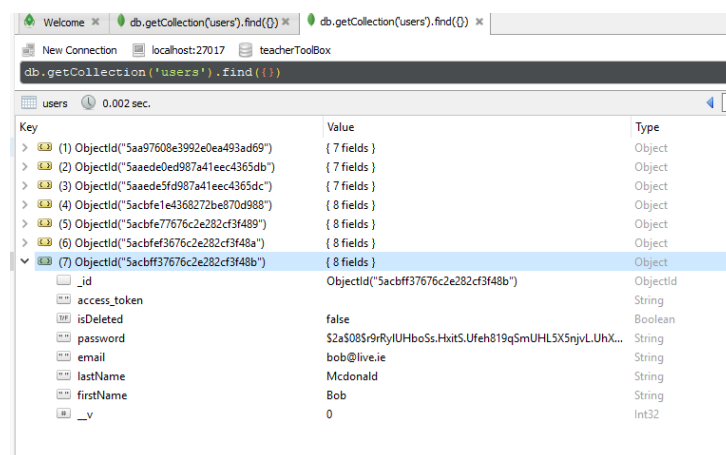


Figure 6.13: User is stored in MongoDB with password encryption.

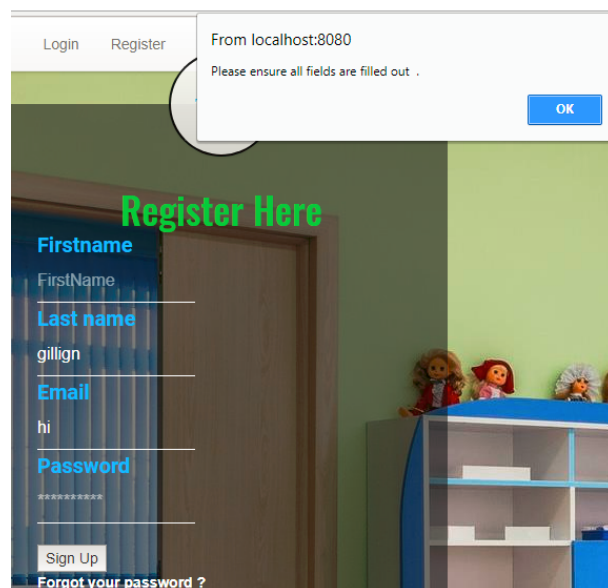


Figure 6.14: Alert when all fields not filled out when Registering.

Test case 1 : PASSED

Test case 2 - User Login

Here I am logging in with a registered Users details. When the correct details are entered the red X's change to green ticks.

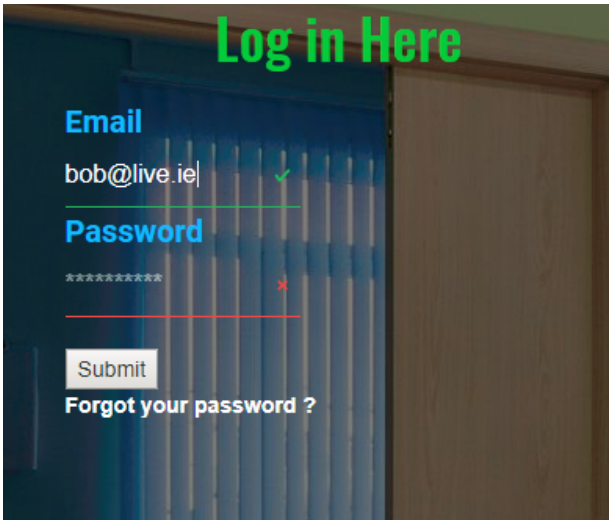


Figure 6.15: Login-colour change.

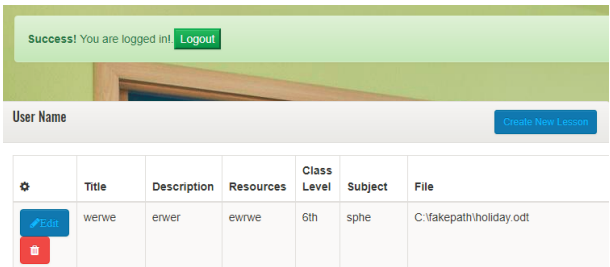


Figure 6.16: Logged in .

Test case 2 : PASSED

Test case 3 - Sending contact Form

Testing contact form. Message box can be re-sized by dragging the bottom-right corner to enlarge.

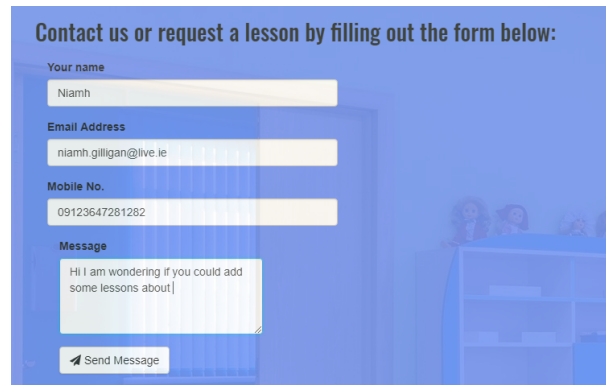


Figure 6.17: Test contact form.

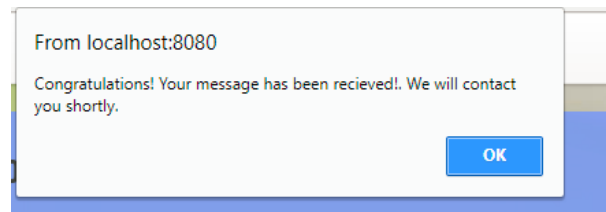


Figure 6.18: Contact message shown when contact form submitted.

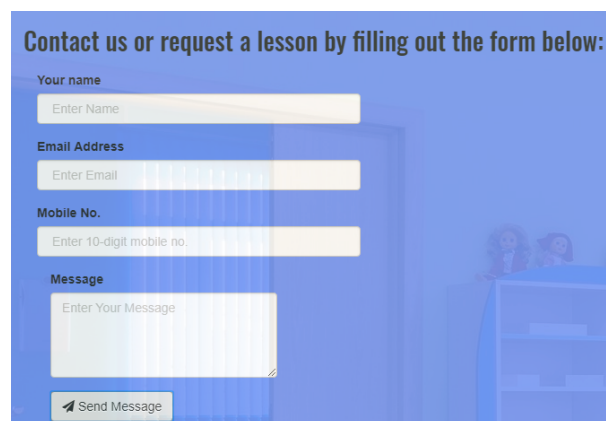


Figure 6.19: Contact form is cleared after message is sent.

Test case 3 : PASSED

Test case 4 - Uploading Lesson Plan

Add Lesson Plan

Title:

Three Little Pigs

Description:

Huff and puff

Resources:

sticks, straw and brick

Class

Junior Infants

Subject:

English

Upload

Upload a file

Choose file

thegreatcod...venture.PNG

Figure 6.20: Testing Lesson Upload.

From localhost:8080

Congratulations! You uploaded a Lesson!.

OK

Figure 6.21: Message after upload clicked.

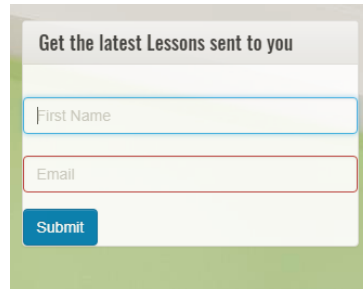
<div><div>Edit</div><div></div></div>	Three Little Pigs	Huff and puff	sticks, straw and bricks	junior	english	C:\fakepath\thegreatcodeadventure.PNG
---------------------------------------	-------------------	---------------	--------------------------	--------	---------	---------------------------------------

Figure 6.22: After upload user is redirected to My Account Page.

Test case 4 : PASSED

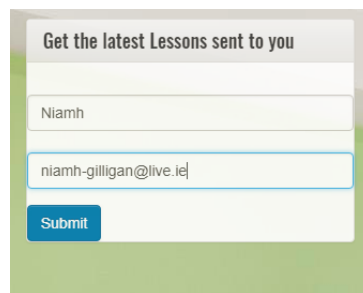
Test case 5 - Sending email Form

Test : Sending email and First name in sign up form.



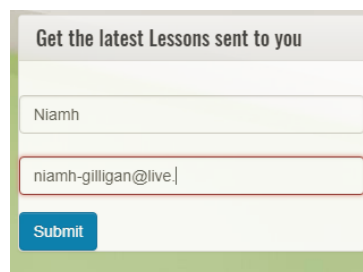
A screenshot of a web form titled "Get the latest Lessons sent to you". It contains two input fields: "First Name" and "Email". Both fields are empty. Below the fields is a blue "Submit" button. The form has a light green background.

Figure 6.23: Empty email Form .



A screenshot of the same web form as Figure 6.23, but with the "First Name" field filled with "Niamh" and the "Email" field filled with "niamh-gilligan@live.ie". The "Submit" button is still blue. The form has a light green background.

Figure 6.24: Email Form when correct email and name has been entered(red box turns to blue) .



A screenshot of the same web form as Figure 6.24, but with the "Email" field filled with "niamh-gilligan@live." (missing the top-level domain). The "First Name" field is still "Niamh". The "Submit" button is still blue. The form has a light green background.

Figure 6.25: Email Form when incorrect email and name has been entered .(box remains red)

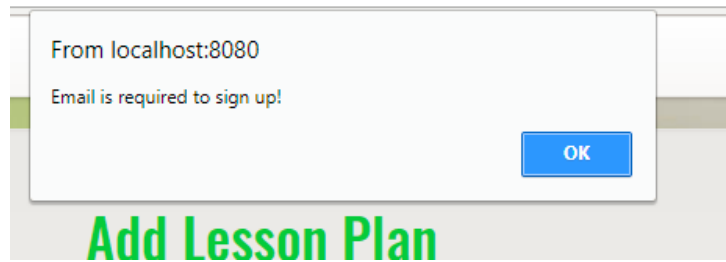


Figure 6.26: Message that appears if incorrect email is entered .

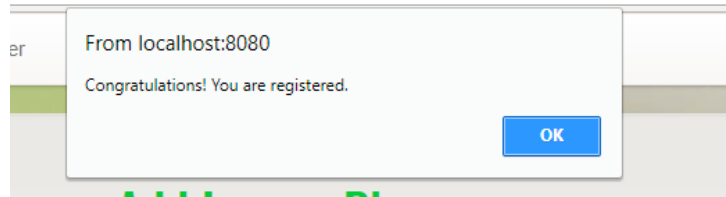


Figure 6.27: Message that appears when correct details entered.

(3) ObjectId("5aca9615dd7d3f20c4f2b1ac")	{ 4 fields }	Object
_id	ObjectId("5aca9615dd7d3f20c4f2b1ac")	ObjectId
email	niamh-gilligan@live.ie	String
firstName	Niamh	String
_v	0	Int32

Figure 6.28: Details stored in MongoDB database shown in Robomongo.

Test case 5 : PASSED

Test case 6 - Delete and Edit Title

Test : Deleting Lesson Plan.



	Three Little Pigs	I huff and I huff and I blow your house down.	Twigs, bricks, straw	junior	english	C:\fakepath\thegreatcodeadventure.PNG
						

Figure 6.29: File which I am testing edit and delete functions .

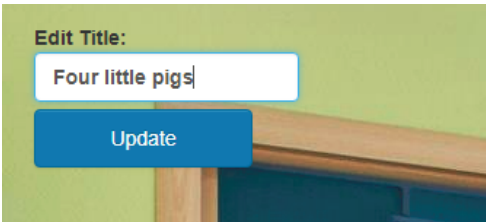


Figure 6.30: Edit title has been clicked and User redirected to edit Title .



	Four little pigs	I huff and I huff and I blow your house down.	Twigs, bricks, straw	junior	english	C:\fakepath\thegreatcodeadventure.PNG
						

Figure 6.31: After page is refreshed we can see the title has been changed .

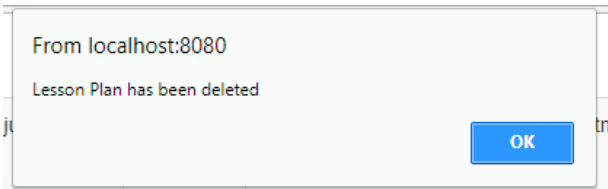


Figure 6.32: When delete button is clicked this message appears to inform user .

Test case 6 : PASSED

Test case 7 - Search Bar Test

I tested my search bar to see if I can search individually by title , description , resources , class level and subject.

Search					
					Q billy
⚙	Title	Description	Resources	Class Level	Subject
📄	Billy Goats Gruff	scary story	bridge, three goats	4th	drama

Figure 6.33: Search by title: Entered "billy".

Search					
					Q Huff
⚙	Title	Description	Resources	Class Level	Subject
📄	Three Little Pigs	Huff and puff	sticks, straw and bricks	junior	english

Figure 6.34: Search by description: Entered "huff".

Search					
					Q 5th
⚙	Title	Description	Resources	Class Level	Subject
📄	hey niamh	hows it going	fred	5th	music
📄	fish fish fish fishy	git	fred	5th	music

Figure 6.35: Search by class level: Entered "5th".

Search					
					Q straw
⚙	Title	Description	Resources	Class Level	Subject
📄	new title yesss	Three little pigs	straw house	4th	geography
📄	Three Little Pigs	Huff and puff	sticks, straw and bricks	junior	english

Figure 6.36: Search by resources: Entered "straw".

Test case 7 : PASSED

Test case 8 - Link Testing

First of all I tested all links on my Navigation Bar to ensure each link presented the correct component. Also when the name Teachers Toolbox is clicked on the navigation bar it redirects to the Homepage.

Contact : PASSED

Create: PASSED

Login: PASSED

Register: PASSED

MyAccount: PASSED

Search: PASSED

Homepage :PASSED

Next I focused on the links from certain components leading to other pages.



Figure 6.37: Link to sign up clicked .



Figure 6.38: Register page is rendered after link is clicked.



Figure 6.39: Create Lesson Button redirects to create lesson page.

Test case 8 : PASSED

6.3.2 User Evaluation

User 1

"The idea of sharing the lesson plans is great if lots of Teachers got involved I think it would be very useful. I think it is easy to use. I would like to have a specific SEN section if that were possible."

Teacher aged 49

User 2

"I like the design of the website and how simple it is to use. I think it would be great to have a link to resources along with the lesson plans."

Teacher aged 27

Bibliography

- [1] SmartBear website. <https://smartbear.com/news/news-releases/smartbear-software-establishes-european-headquarte/>. Accessed: Feb/March and April 2018.
- [2] React and Flux web development for beginners. <http://www.udemy.com>. Accessed: Feb/March 2018.
- [3] Node with React. <http://www.udemy.com>. Accessed: Feb/March and April 2018.
- [4] React , webpack , MongoDB , express, bootstrap ,babel and node tutorials. NetNinja , The new boston , Keith the coder ,Lynda and Traversy media. . www.youtube.com. Accessed: Feb/March and April 2018.
- [5] Mongo DB- The modern application stack. <https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack>. Accessed: 2018-13-03.
- [6] Webpack Documentation. <https://webpack.js.org/>. Accessed: Feb/March and April 2018.
- [7] React Documentation. <https://www.reactjs.org>. Accessed: Feb/March and April 2018.
- [8] Jest Documentation. <https://facebook.github.io/jest/docs/>. Accessed: Feb/March and April 2018.
- [9] Enzyme Documentation. <http://airbnb.io/enzyme/>. Accessed: Feb/March and April 2018.

.1 Appendix

.1.1 Learning Journals

Week 1 :

This week I was assigned my supervisor for my project. We met and discussed the specific technology stack I would be using with my company for my internship during the Summer and possible project ideas. Having previously visited SmartBear, I am aware that the technologies I will be focusing on are Javascript, React , Reflux and LESS. I have planned to meet with Ciaran from SmartBear next week to discuss my project idea and define the functionality and scope required.

Also this week I contacted teaching colleagues of mine and pitched the project idea to them. The aim of this is to get any feedback or any specific functionalities that could be helpful to their teaching/ something different to the websites currently running.

The main task this week is choosing a project and defining its scope. The challenges I face are discovering how best to use the required technology stack and implement these throughout my project. I am hoping that meeting with SmartBear next week will give me a better insight into how I can implement and utilise the best parts of these technologies.

Week 2 :

This week my challenge was to familiarise myself with the technologies proposed for my project as we have not covered the majority of them . Another challenge I faced this week was choosing which technologies would be most suitable for the project I am undertaking. I have chosen to use MongoDB , React Js, Javascript, html, css, Less, mlab, JSON , Express and Bcrypt. I have been exploring the functionality of these technologies and hope they will give the best results for my project. I have also completed the draft for my Requirements Document which I will email to my supervisor next week for feedback. I plan to set up the technologies needed for the initial phase of my project next week and have a clear plan identified.

Week 3 :

1. What tasks have you undertaken in your project this week?
 - submitted Requirement Document and met with Owen to discuss.
 - creating front-end views for register/login page.
 - begun coding schema and code for lesson plan upload page.
 - including react functions within my application
 - creating javascript functions
 - using handlebars
2. What challenges have you encountered or in any way prevented you from progressing those tasks?
 - connecting app to mongo database , difficulty setting up local mongoDB database so I connected to nuig monog database on danu7.
 - using vue.js, react and learning how to use it in my project.
3. What are you going to do next week to resolve those tasks?
 - learn more about how to get my data from my app to post it into the mongo database. Hopefully get this working for next week.
 - continue learning new javascript functions and react components to add functionality to my application

- continue to work on front-end view, creating view page for users when they log in.

Week 4 :

1. What tasks have you undertaken in your project this week

-file uploader coded and created. limited file size and file type using javascript function.

-bootstrap ,html, and javascript for create page.

- connecting front and back api's to post lesson plan information to the mongodb.

-view for users page when they login.(needs functionality)

- Met with my internship company SmartBear.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

-This week I had trouble sending the data from the form to the mongo database but alot of tutorials helped and improved my knowledge in this area so I could fix it.

- I met with my company SmartBear and they have advised me to study React and Reflux for the remainder of this month to ensure a good grasp of this before I implement this into my project. They have given me specific tasks to cover over the next few weeks which will expand my knowledge of these technologies and help me in implementing them correctly into my project work.

3. What are you going to do next week to resolve those tasks?

- I will begin the material and tasks SmartBear have set out for me.

Week 5 : 1. What tasks have you undertaken in your project this week?

-This week I have immersed myself in React tutorials. I have learned how to set up an app using react and also with webpack which I spent time learning to be able to use it with my app . I learned a lot about how react is structured and its advantages over different technologies. I also looked into reflux and redux which I hope to be using in my react app next week.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

-My unfamiliarity with React and the webpack technology meant that I spent alot of time this week on the basics of these and understanding their structure and functionality.

3. What are you going to do next week to resolve those tasks?

-Next week I aim to continue working on my react app and completing some more tasks to further my knowledge of react and hopefully include some reflux and redux elements also.

Week 6 :

1. What tasks have you undertaken in your project this week?

This week I focused on routing in react which will be vital to my project.I also researched project folder structure and which is the best approach to take.I have been making small react projects to understand to basic functionality.SmartBear also sent me on some material to cover based on react and flux. I started one of the tutorials they recommended but have about 3 more hours to complete this.In the material they sent I discovered a really good timeline for learning react which I have attached.I am currently at the build a few things section of this. I have also previously looked into webpack and set up a react app with webpack.So using this as a guide next week I will complete the course sent to me and look into ES6.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

This week I had set backs with some of my projects due to depreciated methods (even from recent tutorials)

3. What are you going to do next week to resolve those tasks?

- Reading of updated documentation is helpful for finding out which methods have depreciated.
- tutorial to finish next week.
- ES6 -learn basics.



Figure .40: React Timeline

Week 7 :

1. What tasks have you undertaken in your project this week?

This week due to the snow closing college for 2 days I took advantage and signed up for a React/Flux course on Udemy. I have completed the majority of this and have learned the following.

- How to create a React Skeleton which can be used for many projects.
- How to create re-usable React components eg: re-usable Nav-bar.
- How to break down items into components.
- React Router and forms.
- Flux /Reflux and how to use these in a React App.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

- Certain elements were outdated so I had some changes and problems to solve by reading documentation on React and Flux.

3. What are you going to do next week to resolve those tasks?

- Next week I will hopefully progress with my website and start getting the basic layout and fundamentals created.

Week 8 :

Completed this week :

- Set up Mern stack for project .
- Set up Routing and Sign in and sign up pages.
- Tested api's using postman.
- got bootstrap to work in my app correctly.
- Api's work! front and back end working together finally.

- I met with SmartBear this week to discuss my project and they were wondering whether I should focus on improving functionality or focus on adding a new technology to my project which I will need for my internship and which will benefit me in my future career.

Challenges :

Getting all technologies to work together.

I also had trouble with my api's but used a new technology called superagent to help me with this. Documentation for this was very good and saves time on the front end.

React functions sometimes trip me up and cause issues as there seems to be alot of updates and modifications to the syntax.

Next week :

- Working on apis for upload lesson page and users homepage.
- Focus on breaking some components up into smaller re-usable components and cleaner code.

Week 9 :

1. What tasks have you undertaken in your project this week?

This week I :

- made a search bar
- made lesson plan page and figured out front and backend.
- made contact page and got front and backend working together also fixed a routing issue I had.
- connected the login so that when someone logs in it will bring them to their user homepage.
- connected blog sign up box to backend.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

- backend delete is working .I have had issues with the front end . I have tried superagent , axios and fetch.
- Having issues with my edit button.
- I need to figure out how a user will be able to enter something into the search bar and what it will show. Alot of thought needs to go into how I can uniquely retrieve each individual lesson plan.

3. What are you going to do next week to resolve those tasks?

Next week my focus is on my searchbar and figuring out how to retrieve the data from the database. I would also love to get my edit and download buttons functioning and fix my delete button.

Week 10 :

1. What tasks have you undertaken in your project this week?

- This week I added functionality to my search bar, it took awhile but I got there in the end.
- I am working on edit and delete functionality . I have figured out the backend and I am very close with the delete button but it just is not working correctly. I have managed to get the individual ID I just need to figure a way to pass it to the method.
- This week I also put thought into the structure of my code and how I could break down certain components so that they are more re-usable.
- I met with my supervisor and we also discussed that I will implement unit tests over the coming weeks.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

Challenges this week were the search bar and how to ensure a user could search by any of the attributes. This was difficult for me as there are many different ways of accomplishing this. I chose to implement the functionality in the search component. I am having a lot of difficulty with my edit and delete buttons but will hope to have one last look next week and see if I can make them work and if not I will have to leave it to one side and focus on more important functionality.

3. What are you going to do next week to resolve those tasks?

Next weeks focus will be on completing edit and delete buttons, implementing upload and download buttons and perhaps start looking into how to implement unit tests in React.

Week 11 :

1. What tasks have you undertaken in your project this week?

- This week I managed to get my delete and edit button working . A user can delete any lesson plan by clicking the delete button and also they can change the title of their lesson plan.
- This week I also got to work on my upload and download buttons , The upload is works well and I can download a single lesson plan , however I need to make the download button work with all the lesson plans.
- I also began looking into how to do unit tests with react which I hope to implement . From my research I think it will be best for me to use the JEST testing framework which I've discovered works well with react.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

- This week I had trouble with the upload and download buttons . It was complicated to get my head around these and work out how to upload them with a unique identifier which I chose as the time and date it was uploaded to prevent any confusion between lesson plans.
- I also have the functionality to download a specific lesson plan but need to work on this to make it more dynamic.

3. What are you going to do next week to resolve those tasks?

Next week I will look more into how I can get the download button to download a specific chosen lesson. I will also attempt some small unit tests .I will look at how I can use the database to just show a

specific Users lesson plans and figure out how best to implement that feature. **Week 12 :**

1. What tasks have you undertaken in your project this week?

This week I have been testing my application using Enzyme Framework and Jest to run the tests. I have begun the write up of my project also. I managed to figure out a way to add security to my web application with React so that some of the pages cannot be viewed without signing in .This is a functionality which I spent a lot of time on trying different ways but finally found one that works well with my application. This week has also been filled with cleaning up my code and taking out any unnecessary code.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

I spent some time trying to figure out the MyAccount page to just show the data that the User has uploaded but to no avail so far.

3. What are you going to do next week to resolve those tasks?

Next week I am going to focus on finishing the write up phase of my project , adding some more tests and attempting to fix the my account page.

.1.2 Requirements Document