Building a React app

Not liking a framework or library doesn't mean that you don't know how to use it. I've build a sample app and waht to document the process, the potholes I dealt with and how I solved them.

I admit that I'm a beginner. There might be easier ways to deal with with some of the issues I encountered and there might be documentation that explains how to solve the issues, I wasn't able to find them.

What we will build

As a community we collect information that we might want to share with the community. Rather than put them in emails that might get lost along the way we can build an app that will take the informatin and then display it in a list.

The application has the following requirements:

- Be able to enter data into a form
- Be able to edit the data in an existing item
- List all the items in the database
- Use MongoDB to store the items

Building the backend with Express and MongoDB

Before we dive into the React specific code we'll create a small API server using React. This server will connect to the MongoDB database and abstract all database connections and JSON rendering from the React postion of the application.

First we create a server directory and move into that directory

```
mkdir server && cd server
```

Next, we initialize an NPM repository using default values.

```
npm init --yes
```

Once we've initalized the project we can install the packages that we need to run the server.

- **Express** provides the framework that we'll use to create the server
- **Mongoose** allows us to create the models that we'll use with our database. It is not required but it will certainly make life easier
- **cors** enables cors requests for the server
- **nodemon** takes care of keeping our application up and running

```
npm install express mongoose cors nodemon
```

Now that we have have everything installed we can write code. The first part is to create our Mongoose Schema that describes the structure of the database.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

let Review = new Schema({
    review_title: {
      type: String,
      required: true
    },
    review_url: {
      type: String,
      required: true
    },
    review_type: {
      type: String
    },
    review_description: {
      type: String
    },
    review_sugested_by: {
      type: String
    },
```

```
});

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

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const cors = require('cors');
const mongoose = require('mongoose');
const reviewRoutes = express.Router();
const PORT = 3010;

let Review = require('./review.model');

app.use(cors());
app.use(bodyParser.json());

app.use('/reviews', reviewRoutes);
```

```
mongoose.connect('path/to/mongodb/reviews', { useNewUrlParser: true });
const connection = mongoose.connection;

connection.once('open', function() {
    console.log("MongoDB database connection established successfully");
})
```

```
reviewRoutes.route('/').get(function(req, res) {
    Review.find((err, reviews) => {
        if (err) {
            console.log(err);
        } else {
            return res.json(reviews);
        }
    });
});
```

```
reviewRoutes.route('/:id').get(function(req, res) {
    let id = req.params.id;
    Review.findById(id, function(err, review) {
        res.json(review);
    });
});
```

```
reviewRoutes.route('/update/:id').post(function(req, res) {
    Review.findById(req.params.id, function(err, review) {
        if (!review) {
            res.status(404).send("data not found");
        } else {
          review.review_title = req.body.review_title;
          review.review_url = req.body.review_url;
          review.review_type = req.body.review_type;
          review.review_description = req.body.review_description;
          review.review_sugested_by = req.body.review_sugested_by;

         review.save().then(review => {
              res.json('Review updated!');
          })
          .catch(err => {
              res.status(400).send("Update not possible");
          });
        }
    });
});
```

```
reviewRoutes.route('/add').post(function(req, res) {
    let review = new Review(req.body);
    review.save()
        .then(review => {
            res.status(200).json({'review': 'review added successfully'})
        })
        .catch(err => {
            res.status(400).send('adding new review failed');
        });
```

```
});
```

```
app.use('/reviews', reviewRoutes);

app.listen(PORT, function() {
    console.log("Server is running on Port: " + PORT);});
```

Using create-react-app

I've chose to create the react portion of the application using create-react-app. As someone who is not completely familiar with the platform I'm ok using their opinionated setup to begin with and then, if necessary, I can eject from the tool and customize the project as needed.

The problem with this approach is that it's a one-way ticket. Once you eject from create-react-app you can't go back to it short of having a Git branch from before you ejected or some other form of backup at the point before ejecting.

```
npx create-react-app my-app
cd my-app
npm start
```

NPX Executes the specified either from a local `node_modules/.bin`, or from a central cache, installing any packages needed in order to run the command, in this case, `create-react-app`.

Once everything is installed we can switch to the directory and run `npm start` to run the default code the application provides. This code is for development and is not optimized for production.

```
my-app
├── README.md
├── node_modules
├── package.json
```

```
├── .gitignore
├── public
|   ├── favicon.ico
|   ├── index.html
|   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
```

Once we have this code running without errors we can move on to create our own components. We'll talk more about the different commands and version when we talk about development versus production builds

Basic CRU, not D

For this project I've chosen not to implement a delete functionality. I will hold that for future consideration.

> The three items I've chosen to consider are: List (read), Create and Update

List Reviews (Read)

```
import React, { Component } from 'react';
import { Link } from 'react-router-dom';
import axios from 'axios';

const Review = props => (
  <details style={{margin: 20}}>
      <summary>{props.review.review_title}</summary>

      <p>URL for resource: <a href={props.review.review_url}>{props.revie
```

```
      <p>Type of content: {props.review.review_type}</p>

      <p>Sugested by: {props.review.review_sugested_by}</p>

      <p>Why should we review this item as a community:</p>
      <section className="description">{props.review.review_description}</
      <p><Link to={"/edit/"+props.review._id}>Edit Item</Link></p>
  </details>

)

export default class ReviewsList extends Component {

    constructor(props) {
      super(props);
      this.state = {reviews: []};
    }

    componentDidMount() {
      axios.get('http://localhost:3010/reviews/')
      .then(response => {
        this.setState({ reviews: response.data });
      })
      .catch(function (error){
        console.log(error);
      })
    }

    reviewList() {
      return this.state.reviews.map(function(currentReview, i){
        return <Review review={currentReview} key={i} />;
      })
    }

    render() {
      return (
        <div>
```

```
          <h3>Items To Review</h3>
          { this.reviewList() }
        </div>
      )
    }
}
```

Create Review

```
import React, {Component} from 'react';
import axios from 'axios';

export default class CreateR'axios'xtends Component {
  constructor(props) {
    super(props);

    this.state = {
      review_title: '',
      review_url: '',
      review_type: '',
      review_description: '',
      review_sugested_by''''
    }

    this.onChangeReviewTitle = this.onChangeReviewTitle.bind(this);
    this.onChangeReviewUrl = this.onChangeReviewUrl.bind(this);
    this.onChangeReviewType = this.onChangeReviewType.bind(this);
    this.onChangeReviewDescription = this.onChangeReviewDescription.bind(t
    this.onChangeRev''wSugestedBy = this.onChangeReviewSugestedBy.bind(th
    this.onSubmit = this.onSubmit.bind(this);
  }

  onChangeReviewTitle(e) {
    this.setState({
      review_title: e.target.value
    });
  }
```

```
onChangeReviewUrl(e) {
  this.setState({
    review_url: e.target.value
  });
}

onChangeReviewType(e) {
  this.setState({
    review_type: e.target.value
  });
}

onChangeReviewDescription(e) {
  this.setState({
    review_description: e.target.value
  });
}

onChangeReviewSugestedBy(e) {
  this.setState({
    review_sugested_by: e.target.value
  });
}

onSubmit(e) {
  e.preventDefault();

  console.log(`Form submitted:`);
  console.log(`Review Title: ${this.state.review_title}`);
  console.log(`Review URL: ${this.state.review_url}`);
  console.log(`Type: ${this.state.review_type}`);
  console.log(`Review Description: ${this.state.review_description}`);
  console.log(`sugested by: ${this.state.review_sugested_by}`);

  const newReview = {
    review_title: this.state.review_title,
    review_url: this.state.review_url,
```

```
    review_type: this.state.review_type,
    review_description: this.state.review_description,
    review_sugested_by: this.state.review_sugested_by,
  };

  axios.post('http://localhost:3010/reviews/add', newReview)
      .then(res => console.log(res.data));

  this.setState({
    review_title: '',
    review_url: '',
    review_type: '',
    review_description: '',
    review_sugested_by: '',
  })

  this.props.history.push('/');
}



render() {
  return (
    <div>
      <form onSubmit={this.onSubmit}>
      <h2>Add Item to Review</h2>
        <div className='grid-container'>
          <div className='grid-x grid-padding-x'>
            <div className='medium-6 cell'>
              <label>Name of item to review
                <input  type="text"
                        placeholder="Item Title"
                        value={this.state.review_title}
                        onChange={this.onChangeReviewTitle}/>
              </label>
            </div>
            <div className='medium-6 cell'>
              <label>URL for item
```

```
              <input  type="text"
                      placeholder="URL of item to review"
                      value={this.state.review_url}
                      onChange={this.onChangeReviewUrl}/>
            </label>
          </div>

          <div className='medium-6 cell'>
            <label>Type of resource
              <input  type="text"
                      placeholder="Item Type"
                      value={this.state.review_type}
                      onChange={this.onChangeReviewType}/>
            </label>
          </div>

          <div className='medium-12 cell'>
           'http://localhost:3010/reviews/add's a group?
              <textarea   placeholder="None"
                          rows='5'
                          value={this.state.review_description}
                          onChange={this.onChangeReviewDescription}></
            </label>
          </div>

          <div className='medium-12 cell'>
            <label>Sugested By</label>
              <input  type="text"
                      placeholder="sugested by"
                      defaultValue={this.state.review_sugested_by}
                      onChange={this.onChangeReviewSugestedBy}/>
          </div>
        </div>
        <input  type='submit'
                className='button'
                value='Create Item to Review'
                onClick={this.onSubmit}/>
      </div>
```

```
        </form>
      </div>
    )
  }
}
```

Update Review

```
import React, {Component} from 'react';
import axios from 'axios';

export default class EditReview extends Component {
  constructor(props) {
    super(props);

    this.state = {
      review_title: '',
      review_url: '',
      review_type: '',
      review_description: '',
      review_sugested_by: '',
    }

    this.onChangeReviewTitle = this.onChangeReviewTitle.bind(this);
    this.onChangeReviewUrl = this.onChangeReviewUrl.bind(this);
    this.onChangeReviewType = this.onChangeReviewType.bind(this);
    this.onChangeReviewDescription = this.onChangeReviewDescription.bind(t
    this.onChangeReviewSugestedBy = this.onChangeReviewSugestedBy.bind(th
    this.onSubmit = this.onSubmit.bind(this);
  }
```

```
  onChangeReviewTitle(e) {
    this.setState({review_title: e.target.value});
  }

  onChangeReviewUrl(e) {
```

```
    this.setState({review_url: e.target.value});
  }

  onChangeReviewType(e) {
    this.setState({review_type: e.target.value});
  }

  onChangeReviewDescription(e) {
    this.setState({review_description: e.target.value});
  }

  onChangeReviewSugestedBy(e) {
    this.setState({review_sugested_by: e.target.value});
  }

  onSubmit(e) {
    e.preventDefault();

    console.log(`Form submitted:`);
    console.log(`Review Title: ${this.state.review_title}`);
    console.log(`Review URL: ${this.state.review_url}`);
    console.log(`Type: ${this.state.review_type}`);
    console.log(`Review Description: ${this.state.review_description}`);
    console.log(`sugested by: ${this.state.review_sugested_by}`);

    const newReview = {
      review_title: this.state.review_title,
      review_url: this.state.review_url,
      review_type: this.state.review_url,
      review_description: this.state.review_description,
      review_sugested_by: this.state.review_sugested_by,
    };

    axios.post('http://localhost:3010/reviews/update/' +this.props.match.p
      .then(res => console.log(res.data));

      this.setState({
        review_title: '',
```

```
        review_url: '',
        review_type: '',
        review_description: '',
        review_sugested_by: '',
      })

      this.props.history.push('/');
  }
'http://localhost:3010/reviews/update/'
```

```
  componentDidMount() {
    axios.get('http://localhost:3010/reviews/' + this.props.match.params.
    .then(response => {
      this.setState({
        review_title: response.data.review_title,
        review_url: response.data.review_url,
        review_type: response.data.review_url,
        review_description: response.data.review_description,
        review_sugested_by: response.data.review_sugested_by,
      });
    })
    .catch(function (error){
      console.log(error);
    })
  }
```

```
  render() {
    return (
      <div>
        <form onSubmit={this.onSubmit}>
        <h2>Edit Review Item</h2>
          <div className='grid-container'>
            <div className='grid-x grid-padding-x'>
              <div className='medium-6 cell'>
                <label>Name of item to review
                  <input  type="text"
                    placeholder="Item Title"
```

```
          value={this.state.review_title}
          onChange={this.onChangeReviewTitle}/>
      </label>
    </div>
    <div className='medium-6 cell'>
      <label>URL for item
        <input    type='text'
          placeholder = 'URL of item to re view'
          value = {this.state.review_url}
          onChange = {this.onChangeReviewUrl} />
      </label >
    </div>

    <div className='medium-6 cell'>
      <label>Type of resource
        <input  type="text"
          placeholder="Item Type"
          value={this.state.review_type}
          onChange={this.onChangeReviewType}/>
      </label>
    </div>

    <div className = 'medium-12 cell'>
      <label>Why should we review this as a group ?
        <textarea   placeholder = 'None'
        rows = '5'
        value = {this.state.review_description}
        onChange = {this.onChangeReviewDescription} > </textarea
      </label>
    </div>

    <div className='medium-12 cell'>
      <label>Sugested By</label>
        <input  type="text"
        placeholder="sugested by"
        defaultValue={this.state.review_sugested_by}
        onChange={this.onChangeReviewSugestedBy}/>
    </div>
```

```
            </div>

            <input  type = 'submit'
                    className = 'button'
                    value = 'Edit Review Item'
                    onClick={this.onSubmit}/>
          </div>
        </form>
      </div>
    )
  }
}
```

Putting it all together

```
import React, { Component } from 'react';
import { BrowserRouter as Router, Route, Link } from "react-router-dom";
import 'foundation-sites/dist/css/foundation.min.css'

import CreateReview from "./components/create-review";
import EditReview from "./components/edit-review";
import ReviewsList from "./components/reviews-list";

class App extends Component {
  render() {
    return (
      <Router>
        <div className="top-bar foundation-5-top-bar">
          <div className="top-bar-title">
            <Link to="/"><strong>Items to review</strong></Link>
          </div>
          <div className="right">
            <Link to="/create">Create Review</Link>
          </div>
        </div>
```

```
        <Route path="/" exact component={ReviewsList} />
        <Route path="/edit/:id" component={EditReview} />
        <Route path="/create" component={CreateReview} />
      </Router>
    );
  }
}

export default App;
```

The final piece is `index.js`

```
import React from 'react';
import ReactDOM from 'react-dom';
import './in'react-dom'mport App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />,'./App'nt.getElementById('root'));

'root'// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

SEO :`react-helmet`

`react-helmet` from the NFL (See the helmet reference) provides a way to inject content into the

```
import { Helmet } from 'react-helmet';
```

```
<Helmet>
  <meta charSet='utf-8' />
  <title>My Title</title>
  <link rel='canonical' href='http://mysite.com/example' />
</Helmet>
```

SSR: `react-static`

The idea behind Server Side Rendering an application is to make it load faster and be a better friend with search engine crawlers (although this is less of an issue now that the Googlebot crawler moved to a current version of Chrome)

Development versus production build

Publishing

Future requirements

Adding user accounts

Implementing Delete Functionality

Links and Resources

- Basic Crud Project
  - https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-1/
  - https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-2/
  - https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-3/
  - https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-4/
- Static build
  - https://medium.com/superhighfives/an-almost-static-stack-6df0a2791319
  - https://github.com/nfl/react-helmet
  - https://www.npmjs.com/package/react-snapshot
- User accounts
  - https://www.sitepoint.com/build-react-app-user-

authentication-15-minutes/