



# mApp My Fish

Project Engineering

Year 4

Joseph Fox

**G00361752**

Bachelor of Engineering (Honours) in Software and  
Electronic Engineering

Galway-Mayo Institute of Technology

2020/2021

## Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

\_\_\_\_\_ Joey Fox \_\_\_\_\_

## Acknowledgements

I would like to thank my supervisor Brian O'Shea for helping me throughout the year. His extensive experience in full stack programming and generosity with his time was of major benefit to my project. I would also like to thank Paul Lennon for his help throughout the year.

## Table of Contents

1	Summary.....	5
2	Introduction.....	7
3	Project Architecture .....	8
4	Project Plan Gannt Chart .....	9
5	Development Platform and Tools .....	10
5.1	Visual Studio Code .....	10
5.2	React Native .....	11
5.3	Expo .....	11
5.4	Node.js.....	11
5.5	MongoDB.....	11
6	Key API calls .....	12
6.1	Signup.....	12
6.2	Login .....	14
6.3	React Map Api .....	15
7	Ethics.....	17
8	Conclusion .....	17
9	References .....	18
10	Appendix .....	18
10.1	Links .....	18
10.2	Poster.....	19

# 1 Summary

This app provides anglers with a platform to record and store information about their experience fishing on Lough Corrib. The app is known as MyFish and includes a geo map with a default location of Lough Corrib on which you can pinpoint where you caught fish by clicking on the map and adding a marker. Additional data can be logged as if using a diary.

This data includes:

- Method of fishing used.
- What bait the fish was caught with.
- Weather conditions (Temperature, overcast/clear skies, wind direction and type of wave)
- Location i.e. Where the fish was caught.
- Species of fish caught/size.
- Time of year (date).

You can also take pictures through the app or chose a picture from the existing gallery on your mobile.

The app consists of a login screen, signup screen, home page and a map screen where you can add markers of where fish were caught or seen. On the home page you can also go to your gallery where you can take a picture or chose a picture from your library.

This project was motivated by my wish to be more systematic in my fishing. Catching a fish is a relatively rare event. A good day fishing would see three fish caught over eight hours and it is not unusual to catch nothing at all. There are a lot of variables on the lake: weather conditions, light, wind, bait used, date, time of day, location, water temperature and many others. I felt that storing and analysing these variables and ultimately linking them with 'success' would improve my fishing results.

This project was started by establishing general user requirements. Then, I researched suitable languages and technologies. Following this, I wrote the code for the front end of my app. When the front end of my app was complete, I researched the back end of my project and incorporated Node.js and mongo db.

My project is an expo app written in JavaScript. The backend of my project consists of a server and a mongo database where all users and fishing details are stored. The app communicates with MongoDB using a Node.js server.

By developing this app, I have created a place where anglers can efficiently store their data. This allows anglers to quickly and easy make logs of their fishing day.

I think my project could benefit anglers hugely. Market research showed that there was nothing like it on the market. The app could be further developed to advise beginner anglers where to fish and what method and bait to use given the prevailing conditions. The main screen is shown in Figure 1 below.

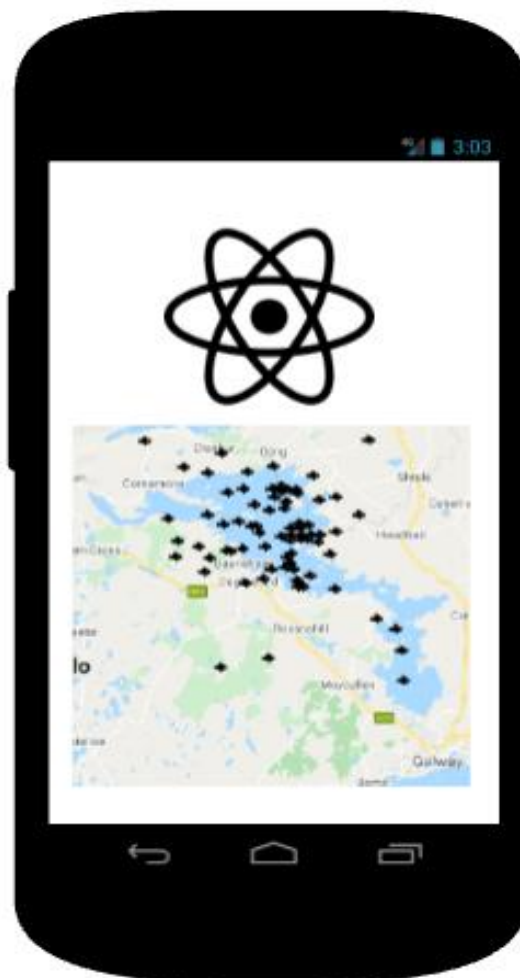


Figure 1: Main Screen

## 2 Introduction

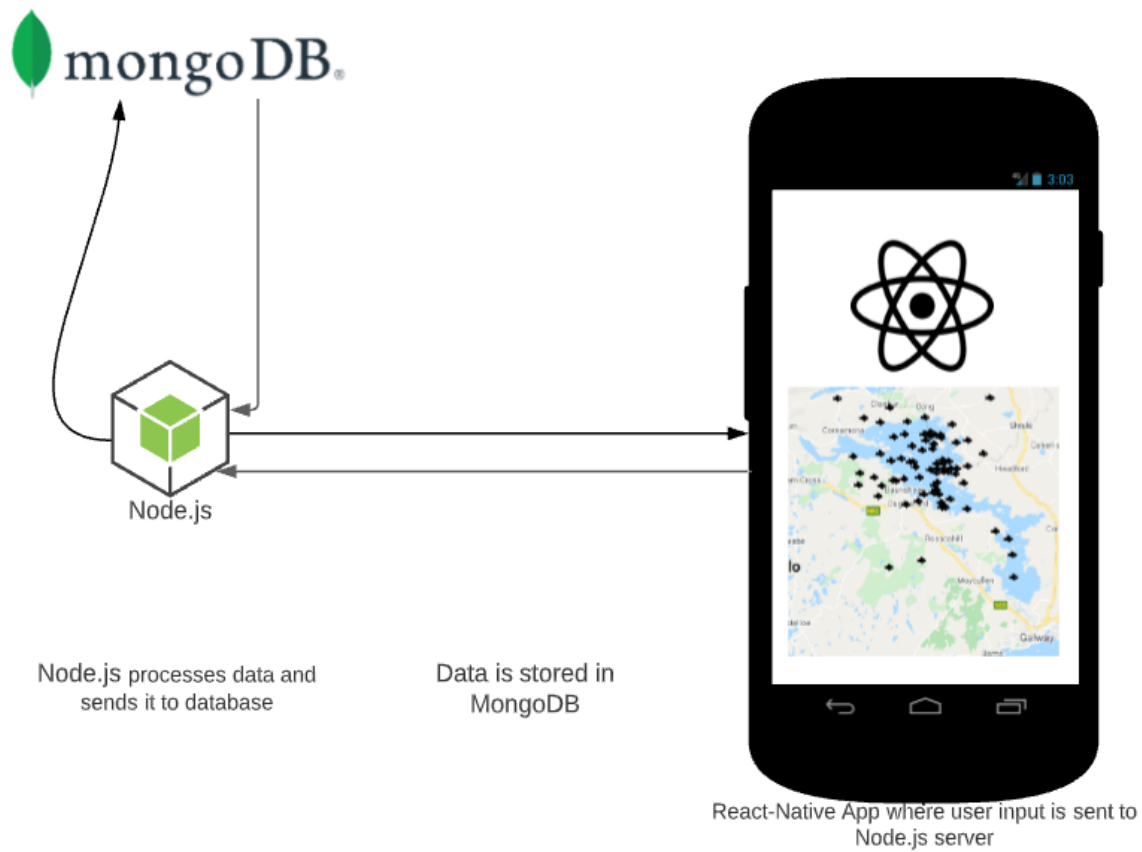
Inspired by my time fishing with my father and grandfather on the big lakes of Ireland, I decided to make an app that would help other fishermen store data about their fishing day.

Traditionally fishermen would have kept a diary with details of where they caught fish and with what bait was used. They would also have kept details such as weather conditions, wind direction, temperature, fishing method and so on. The app I have created removes the need to bring around a notepad or diary or to remember everything by memory and opens the possibility to analyse the data and make recommendations and suggestions to improve decision making around choices regarding fishing location and bait or flies to be used.

When the app is launched, you are directed to a login screen where you are prompted to enter your email address and password. A third option at the bottom of the screen reads “no account yet? Signup here”, which will redirect you to the sign-up screen. When signed up, your details will be saved to mongo DB provided that that email hasn’t already been used. When logged in you can navigate to the home screen where you further navigate to a Google map set to Lough Corrib or to a gallery screen where you can take a picture or choose a picture from your existing photos on your phone. On the map you can add a marker which is a fish icon. This icon will represent where you caught a fish. You then add details which are saved in the database.

### 3 Project Architecture

The basic architecture schematic of my project is shown in Figure 2 below. The user inputs details through the app. The user input is sent to a Node.js server where it is processed and saved to Mongo DB. Data can be pulled from the database and displayed.



**Figure 2: Architecture Schematic**

Initially I was unsure on how to implement a database to my project. I researched different methods and databases such as firebase before deciding on Mongo db. When I first tried to implement Mongo dB, I tried to connect my app directly to the database using a Realm app API. After struggling to integrate the realm app Api, in conjunction with my supervisor, I decided to use Node.js to connect my app to Mongo db.



## 4 Project Plan Gantt Chart

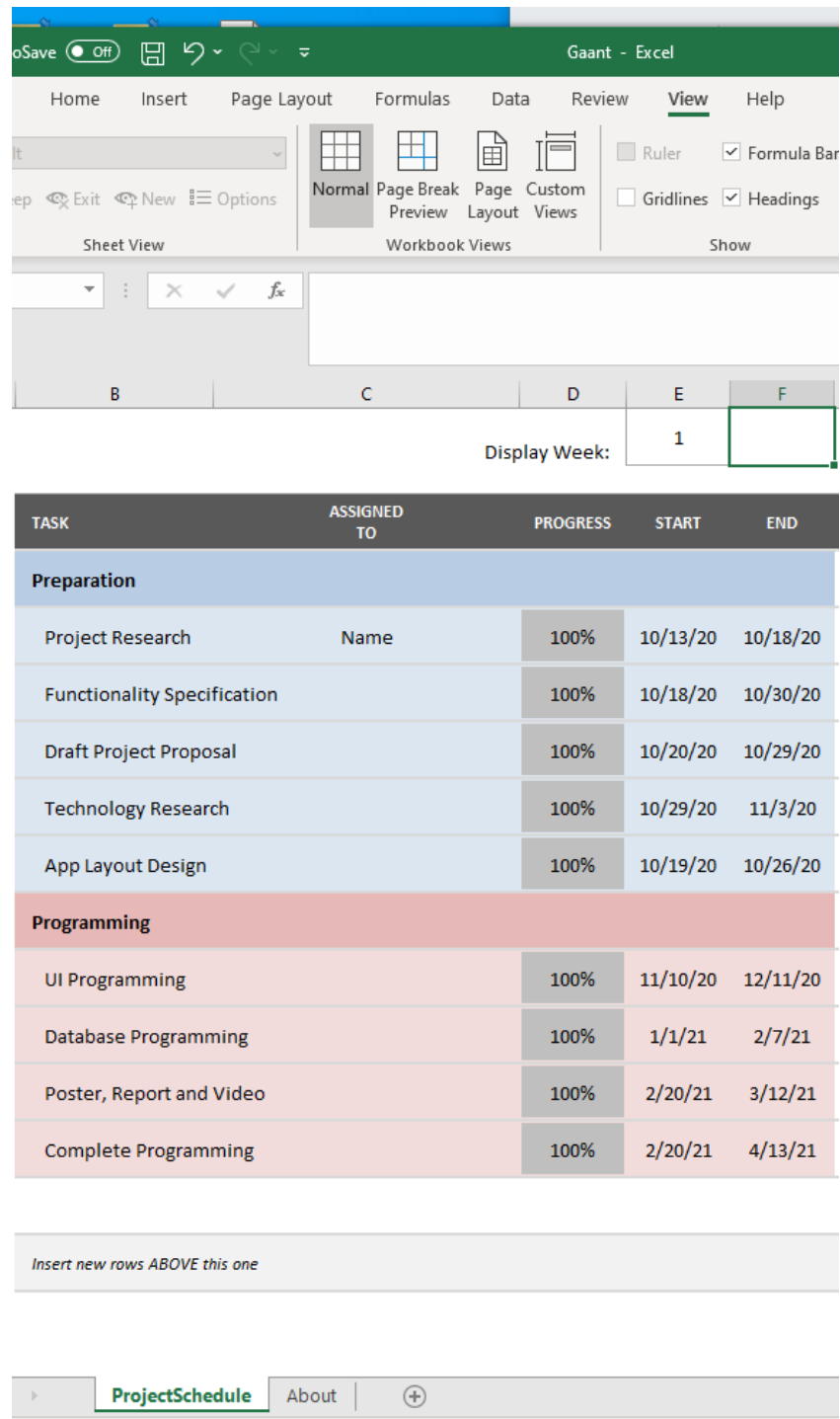


Figure 3: Screenshot of Project Plan Gantt Chart

See Poster in Appendix.

## 5 Development Platform and Tools

Each of the technologies used in the development are introduced below.

### 5.1 Visual Studio Code

Visual Studio Code is a freeware source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette. [1] I used this IDE to write my code for the front end of the app and the server. A screenshot is shown in Figure 4 below.

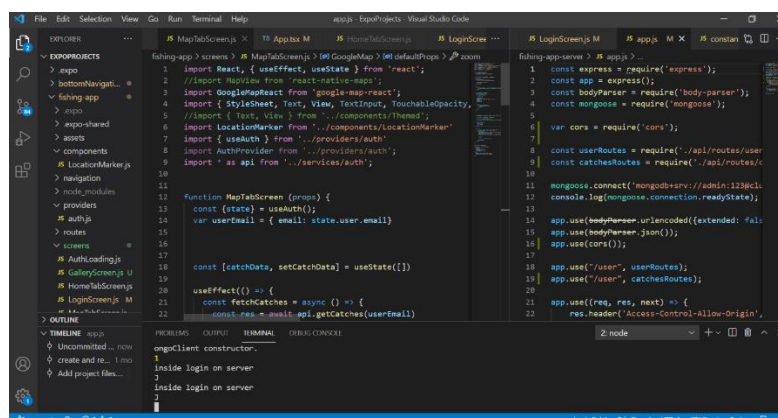


Figure 4: Visual Studio IDE screenshot

## 5.2 React Native

React Native is an open-source mobile application framework created by Facebook, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web, Windows and UWP by enabling developers to use React's framework along with native platform capabilities.[2]

## 5.3 Expo

Expo is a framework and a platform for universal React applications. It is a set of tools and services built around React Native and native platforms that help you develop, build, deploy, and quickly iterate on iOS, Android, and web apps from the same JavaScript/TypeScript codebase.[3]

## 5.4 Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts. Node.js has an event-driven architecture capable of asynchronous I/O.[4]

## 5.5 MongoDB

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.[5] In my project I use MongoDB to store user login and signup data as well as co-ordinates of catches details of the catch.

## 6 Key API calls

Some of the key API calls in the app are introduced below.

### 6.1 Signup

The async function below calls the register api from services / auth.js.

```
async function onSubmit(state){  
  let response = await api.register(state);  
  
  Alert.alert('Registration Successful',  
    response.message,  
  )  
}
```

Figure 5: Register API 1

The async function below awaits a call. c.REGISTER is called from constants where the URLs for the server are. The URLs in constants are the link between the app and the server.

```
export async function register(data){  
  try{  
    let res = await axios.post(c.REGISTER, data);  
  
    return res.data;  
  }catch (e) {  
    throw handler(e)  
  }  
}
```

Figure 6: Register API 2

In Figure 7 below, User.find is getting the email address that was entered by the user and checking if the email address has already been used. If the email address exists in the database already an error message will be returned. If not the password will be hashed and a user will be created and saved in the database.

```
router.post('/signup', (req, res, next) => {
  User.find({email: req.body.email})
    .exec()
    .then(user => {
      console.log("here")
      if (user.length >= 1){
        return res.status(409).json({
          message: 'An account with this email address already exists'
        });
      } else {
        bcrypt.hash(req.body.password, 10, (err, hash) =>{
          if (err){
            return res.status(500).json({
              error: err
            });
          } else {
            const user = new User({
              _id: new mongoose.Types.ObjectId(),
              email: req.body.email,
              password: hash
            });
            user
              .save()
              .then(result => {
                console.log()
                res.status(201).json({
                  message: 'User created'
                });
              })
              .catch(err => {
                console.log(err);
                res.status(500).json({
                  error: err
                });
              });
          }
        });
      }
    })
    .catch(err => {
      console.log(err);
      res.status(500).json({
        error: err
      });
    });
});
```

Figure 7: Register API server code

## 6.2 Login

The login api is being called from login api from services / auth.js. a token is passed back from the login api along with the username. These are stored in reponse which are passed to handleLogin where they are stored in an async storage. Login status can be checked using useAuth. If there is a username in the response it means the login was successful and we can navigate to home page.

```
async function onSubmit(state){
  try {
    let response = await api.login(state);
    await handleLogin(response);

    let username = (response.user.username !== null);
    if (username) navigate('Home');
  } catch (error) {
    console.log(error)
  }
}
```

Figure 8: Login API 1

Handle login in Figure 9 is where the token and the user data is stored. It also sets the Axios Authorization header and dispatches the user data to the reducer to be saved. [6]

```
// Handle Login
const handleLogin = async (data) => {
  try{
    //STORE DATA
    let {user, message, token} = data;
    let data_ = [[USER_KEY, JSON.stringify(user)], [TOKEN_KEY, token]];
    await AsyncStorage.multiSet(data_);

    //AXIOS AUTHORIZATION HEADER
    axios.defaults.headers.common["Authorization"] = `Bearer ${data.token}`;

    //DISPATCH TO REDUCER
    dispatch({type: LOGGED_IN, user:data.user});
  }catch (error) {
    throw new Error(error);
  }
};
```

Figure 9: Login API 2

### 6.3 React Map Api

The google map is passed the catch data from the database specific to what user is logged in. Catch data is stored in markers when you click on the map. The default location below is centered to the lough Corrib as in Figure 10.

```
const GoogleMap = ({ catchData, userEmail, center, zoom }) => {

  const [selected, selectMarker] = useState([])

  const markers = catchData.map( ev => {
    return <LocationMarker lat={ev.location.lat} lng= {ev.location.lng} customClickEvent={()
      =>{selectMarker(ev);console.log(ev)}}></LocationMarker>
  })

  const defaultProps = {
    center: {
      lat: 53.4964,
      lng: -9.2764
    },
    zoom: 11
  };
};
```

Figure 10: React Map API

Below in Figure 11 is a method getCatches which calls the API in services / auth.js. The useeffect is used here to keep making calls to the API so that the map refreshes when there is a new marker added.

```
const [catchData, setCatchData] = useState([])

useEffect(() => {
  const fetchCatches = async () => {
    const res = await api.getCatches(userEmail)

    setCatchData(res.catches)
  }
  fetchCatches()
},)
```

Figure 11: FetchCatches Map API call

In Figure 12 services / auth.js in the get catches the URL for the for the server are found through constants.js. The code below is making the call to the server where the information is being pulled from the database.

```
export async function getCatches(data){
  try{
    let res = await axios.post(c.CATCHES, data);
    return res.data;
  }catch (e) {
    throw handler(e);
  }
}
```

Figure 12: Map API

Below in Figure 13 is where the catch information from the user is being put into the database through the server.

```
export async function addCatch(data){
  try{
    console.log(data)
    let res = await axios.put(c.ADDCATCH, data);
    return res.data;
  }catch(e){
    throw handler(e);
  }
}
```

Figure 13: Catch Information Call



## 7 Ethics

Each API call and technology used is referenced.

## 8 Conclusion

Overall, I consider my project a success. The project has a functioning web and expo app that can be deployed on both iPhone and Android phones. Once logged in you can add a marker to a specific coordinate as intended. The camera and gallery are also functioning as intended. The database is saving all user inputs specific to each user. The app is user friendly and because nothing like it exists on the market, I think it would be a success if launched on the app store.

The project has a great deal of scope for further developments, including adding AI algorithms which look for most successful fishing methods, places, baits and based on these factors tells the user where to fish, what method to use and what bait to use.

## 9 References

- [1] [https://en.wikipedia.org/wiki/Visual\\_Studio\\_Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)
- [2] [https://en.wikipedia.org/wiki/React\\_Native](https://en.wikipedia.org/wiki/React_Native)
- [3] <https://docs.expo.io/>
- [4] <https://en.wikipedia.org/wiki/Node.js>
- [5] <https://en.wikipedia.org/wiki/MongoDB>
- [6] <https://betterprogramming.pub/how-to-add-authentication-to-your-react-native-app-with-react-hooks-and-react-context-api-46f57aedbbd>

## 10 Appendix

### 10.1 Links

Link to github: <https://github.com/JoeyFox9>

Code for the project I am presenting: <https://github.com/JoeyFox9/fishing-app2>

Server: <https://github.com/JoeyFox9/fishing-app-server2>

Initial project I used (was forced to change project): <https://github.com/JoeyFox9/fishing-app>

Weekly logs: in with app files.

Two-minute video: <http://youtu.be/TKY5ufOdiL4?hd=1>

Seven-minute video presentation: <http://youtu.be/nlc0d-08Ll8?hd=1>

## 10.2 Poster

### mApp My Fish

By Joseph Fox

#### INTRODUCTION


Inspired by my time fishing with my father and grandfather on the big lakes of Ireland, I decided to make an app that would help the likes of my father and grandfather store data about their fishing day. Traditionally fishermen like my grandfather would have kept a diary with details of where they caught fish and with what they caught the fish with. They would also have details such as weather conditions, wind direction, temperature, fishing method and so on. The app I have created removes the need to bring around a notepad or diary or to remember everything by memory.

#### SKILLS & TECHNOLOGIES

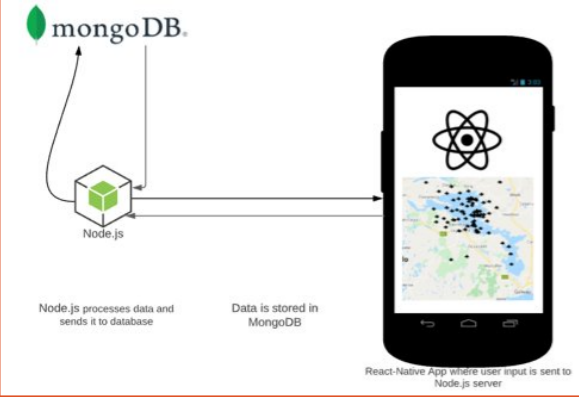
My project is an expo app written in JavaScript using the react native framework. The backend of my project consists of a Node.js server and a mongo database where all users and fishing details are stored.

- JavaScript programming
- Full stack development

Bachelor of  
Engineering (Honours) in Software  
and Electronic Engineering  
Galway-Mayo Institute of  
Technology



### Project Architecture



#### FEATURES

mApp My Fish provides anglers with a platform to store all their fishing data. The app includes:

- Geo Map with a default location of the Lough Corrib.
- A Gallery Screen in the app will allow users to take pictures. Users can also access pictures that have already been taken on their phones.
- Signup Screen where users enter their email address and a password of your choice which is secured using bcrypt.
- Log in using the details you created in the signup screen.

#### CONCLUSION

Overall, I consider my project a success. The project has a functioning web and expo app that can be deployed on both iPhone and Android phones. Once logged in you can add a marker to a specific coordinate as intended. The camera and gallery functionality are also functioning as intended. The database is saving all user inputs specific to each user. The app is incredibly user friendly and because nothing like it exists on the market, I think it would be a success if launched on the app store.

Figure 14: Poster Presentation