

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: Daniel Gilbert
Student ID: X13516403
Supervisor: Adriana Chris

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: DANIEL GILBERT _____

Date: __13/05/2018_____

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

What constitutes plagiarism or cheating?

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a particular piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

Penalties for Plagiarism

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced

- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and
- the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.

National College of Ireland

BSc in Computing

2017/2018

Daniel Gilbert

X13516403

Daniel.Gilbert@student.ncirl.ie

Total Fishing Web Application

Technical Report



Executive Summary	7
Introduction	8
Background	8
Aims	9
Technologies	9
System	11
Requirements	11
Functional requirements	11
Non-Functional Requirements	20
Performance/Response time requirement	20
Security requirement	20
Data requirements	20
Data requirements	21
Database Design	21
User requirements	24
Environmental requirements	24
Usability requirements	24
Functional Requirements	24
Non-Functional Requirements	27
Design and Architecture	29
Implementation	30
Back-End Development	30
Graphical User Interface (GUI) Layout	37
Testing	46
Think Aloud Test	46
Think aloud test results analysis	47
Heuristic Evaluation	48
Conclusions	51
Further development or research	51
References	51
Appendix	53
Project Proposal	53

Executive Summary

Fishing is one of the biggest recreational activities in the world, and although the older generation enjoys that little technology is needed to fish, that does not mean it should be so far behind in IT compared to other hobbies. I have grown up fishing, and I am included in the new generation of which believe technology and community could advance how people fish.

The older generation believes that fishing spots should be kept secret, so they can enjoy fishing there for as long as possible, as there is a big problem in Ireland with poaching for fish. I believe this application will provide more awareness and highlight the safety of the fish stocks in our countries.

Total Fishing will target Freshwater anglers, generally from the age group of 10-40. The application will provide a community to all fishermen where they can share all sorts of information about their catches. For inexperienced users, there will be a section of the app that will use an image recognition system to detect what species of fish they have caught.

There will be a section that all users will get use from, which will be the best fishing day predictor. This page will use a weighted score model that was based off multiple datasets containing over 100 values. The user will select a location on the map where they would like to fish; this will automatically retrieve a 5-day forecast of weather information on that current location. This data will be stored in arrays and scored accordingly.

1 Introduction

Fishing is a favourite hobby of mine, I have been fishing for many years, and one thing I noticed in the fishing community is a lack of technology and community. I believe that some of the older fishing generations like to keep things secret and may not like the use of technology currently, but a new generation of people are starting to fish that are more technology friendly. I believe that this could be a big gap in the market to take advantage of, as fishing is one of the biggest recreational sports in the world. I believe that a community where users can share their catches along with the location, weather information, date and method used. With this information the weighted score model will get very smart over time as the user's information is logged everytime they catch, a dataset can be generated from the database and can be used to generate a linear regression graph to highlight at what value of the particular data the fish is caught at. The image recognition system will be based off a dataset of images which have been ran through a training model with 8000 training steps. The result of the species will only be displayed if the accuracy of the image recognition system is over 60%.

1.1 Background

During my time of fishing, I have come to notice that there were certain patterns in weather conditions that would help me catch fish and some weather conditions that didn't allow me to catch fish. So, I started to log conditions manually of when I caught fish, where what weather conditions, how I caught it etc. I came to notice that some fish were easily caught when the atmospheric pressure was low, and when the light was low. That is when I came up with this application idea somewhere online that I could store data and find trends in fish. It would be much easier having structured data than taking down information in a notepad. I discussed this idea with some fishing colleagues, and they agreed that there were certain conditions that made it easier to catch fish. This was when I came up with the idea of creating a community to gather a mass of data and recognise

patterns. The patterns would be recognised by running the dataset through a python analysis tool which could generate the linear regression between the variables. Upon recognition of the patterns, a weighted score model would be built updated and compared to a forecast thus producing what the best day to go fishing would be. This would be a great way to save time for all fishermen who would like to go out with a better chance of catching.

1.2 Aims

- The application should be able to create a fishing community where users can share pictures and other details about the catches.
- Users should be able to interact with other users.
- The application will be aimed at freshwater fishing only.
- A user should be able to use an image recognition artificial intelligent algorithm to name the species of the fish.
- The application will use a weighted score model to recommend users the best time/day to go fishing, the best method to use and the best fish to fish for. The application should be capable of becoming smarter over time as it will progress from each user's input. This input will consist of the date, time, temperature, atmospheric pressure, wind, rain and clouds.

1.3 Technologies

- Node.js will be used as the backend and also used to generate the front end of the web app. Node.js is an open-source JavaScript run-time environment for executing JavaScript server side. This will help connect with MongoDB and feed data to the web page using 'Express', which is a node.js framework.
- MongoDB/Mongoose will be used to store all the information from the application. MongoDB is a NoSQL database. It works well with node.js and can be implemented via node package manager. Mongoose is the

node package associated with connecting and communicating with MongoDB. Mongoose provides a vast amount of documentation on how to set up, and create methods such as find a user.

- Python is a high-level programming language, Ran server side. This language will be the easiest to implement the Artificial intelligent algorithm; it will be done through the use of TensorFlow which is an open-source software library for machine learning across a range of tasks. It is a symbolic math library and used as a system for building and training neural networks to detect and decipher patterns and correlations, analogous to human learning and reasoning.
- ExpressJS (express) is a web framework for node.js. This helps to create the front end, and to route around the application, it can also help with requesting and responding with information to the user.
- HandleBarsJS is a view templating engine to be used with express. With this templating engine, it will be easy to display variable and objects sent from the node.js server to the front end.
- Tensorflow is a machine learning framework created by Google. It will be used to create the image recognition system for my application.
- Flask is a minimalistic framework for python. It was perfect to create an API that implemented methods to run tensorflow functions from the node.js server.
- Amazon Web Services(AWS) is a cloud service platform that offers computational power with their servers. This was needed to process the image recognition methods.
- Heroku is a cloud service similar to AWS but without the computational power. It is free to host small scaled applications which fit perfectly for my node.js application.

2 System

2.1 Requirements

2.1.1 Functional requirements

2.1.2 Requirement 1: User Registration

Description & Priority

The scope of this use case is to describe how a user can register to the application.

Use Case

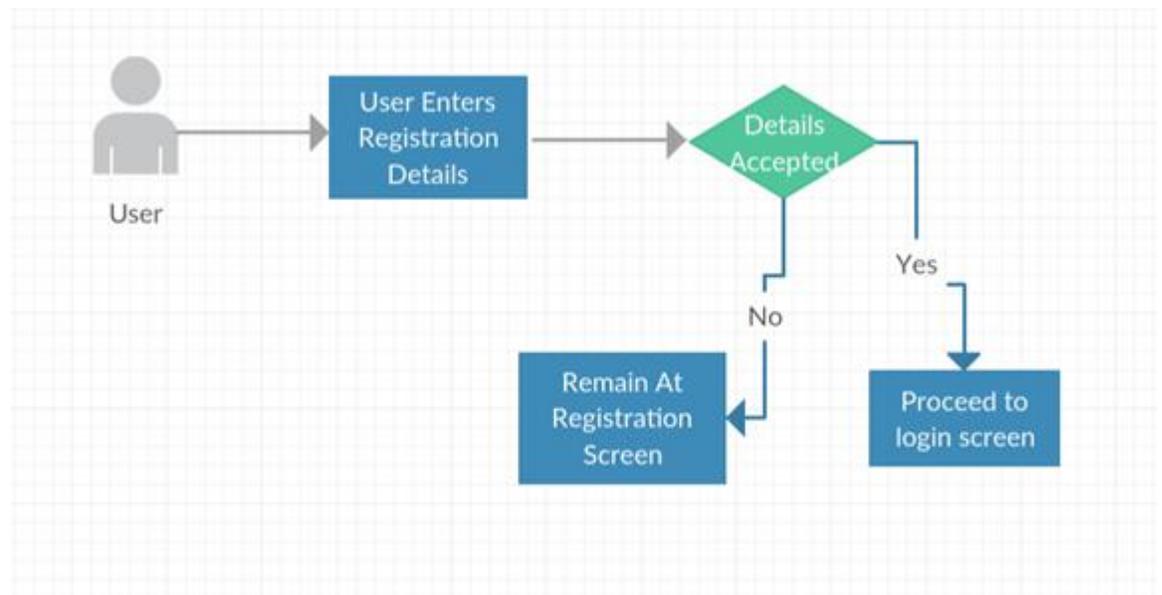
Scope

The scope of this use case is to describe how a user can register to the application.

Description

This use case describes the process of registering from a user's point of view

Use Case Diagram



Flow Description

Precondition

No precondition

Activation

This use case starts when a new user visits the website

Main flow

1. The user enters in the correct information
2. The user clicks the register button
3. The registration has been successful
4. The user is brought to the login page

Alternate flow

1. The user enters incorrect information to the form
2. The user clicks the register button
3. The registration has been unsuccessful
4. The user remains at the registration page

Termination

The system stores user information to the database and user is directed to login screen

Post condition

The user can now login with their details.

2.1.2.1 Requirement 2: User Login

Description & Priority

The Login Use Case is essential to the application's. All functionality requires a user to be logged into their account.

Use Case

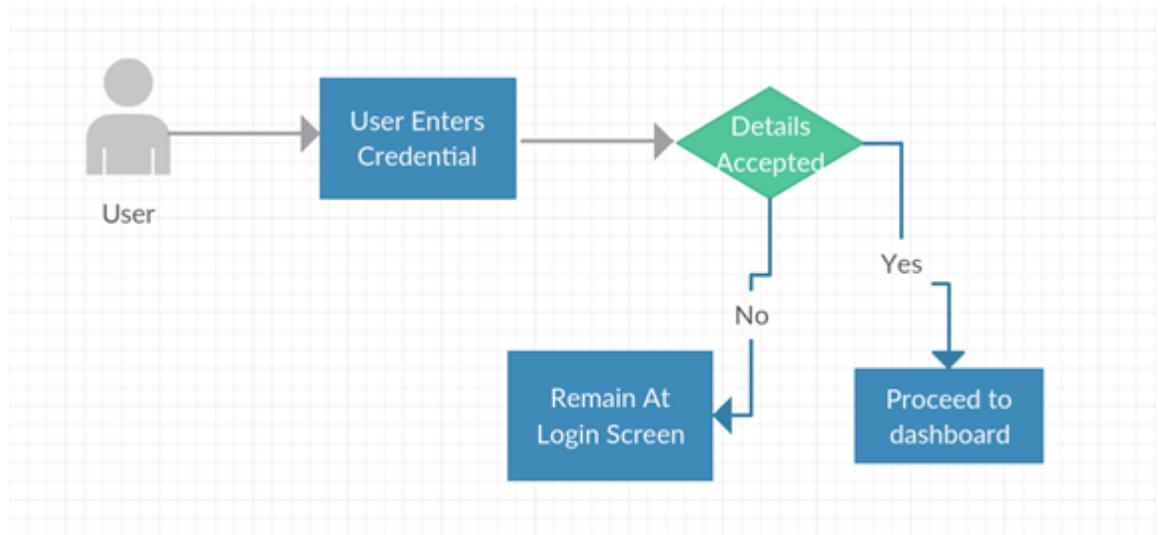
Scope

The scope of this use case is to describe how a user can login to the application.

Description

This use case describes the process of login from a user's point of view

Use Case Diagram



Flow Description

Precondition

Requirement 1: User Registration

Activation

This use case starts when a user visits the website or has just been registered

Main flow

1. The user enters correct username
2. The user enters correct password
3. The user clicks the login button
4. The login has been successful
5. The user is brought to the dashboard

Alternate flow

1. The user enters incorrect information to the form
2. The user clicks the login button
3. The login has been unsuccessful
4. The user remains at the login page

Termination

The system matches the correct username and password and the user is brought to the dashboard

Post condition

The user can now use all functionality of the application.

2.1.2.2 Requirement 3: User Post

Description & Priority

The posting Use Case is an essential function to the application.

Use Case

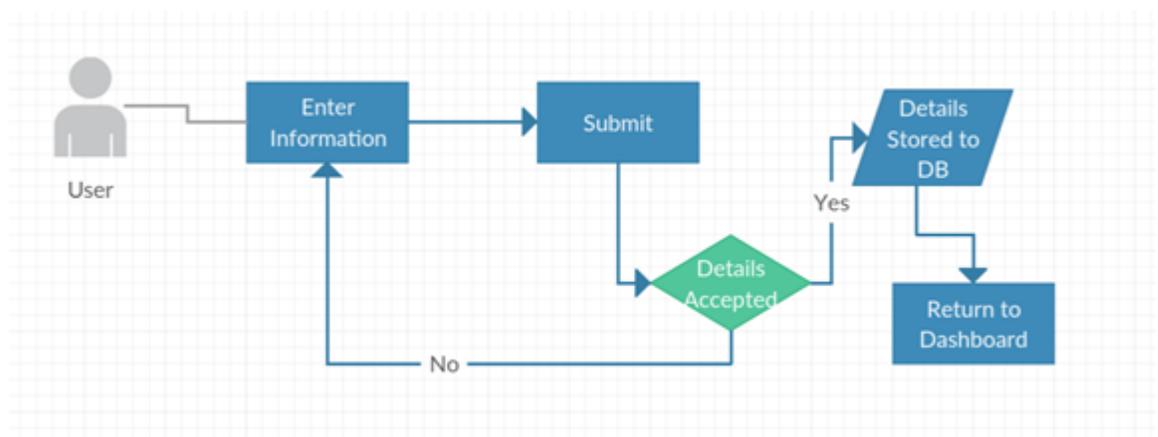
Scope

The scope of this use case is to describe how a user can post a catch to the application.

Description

This use case describes the process of posting from a user's point of view.

Use Case Diagram



Flow Description

Precondition

Requirement 2: User Login

Activation

This use case starts when a user has logged in and selects the create post button.

Main flow

1. The user enters information about fish
2. The user uploads picture of fish
3. The user enters location of catch
4. The user enters weather information
5. The user selects whether it is to be kept private or public.
6. The user hits submit button.
7. Information is saved to database and user is redirected to home screen where the information shall appear

Alternate flow

1. The user does not enter all the required information
2. The user clicks the submit button
3. The submission has been unsuccessful
4. The user remains at this page

Termination

The system successfully submits the users post and he is brought back to the dashboard where his new information will be shown.

Post condition

The user has now logged information of his catch.

2.1.2.3 Requirement 4: Image Recognition

Description & Priority

Image recognition will be an important part of the application for beginning anglers

Use Case

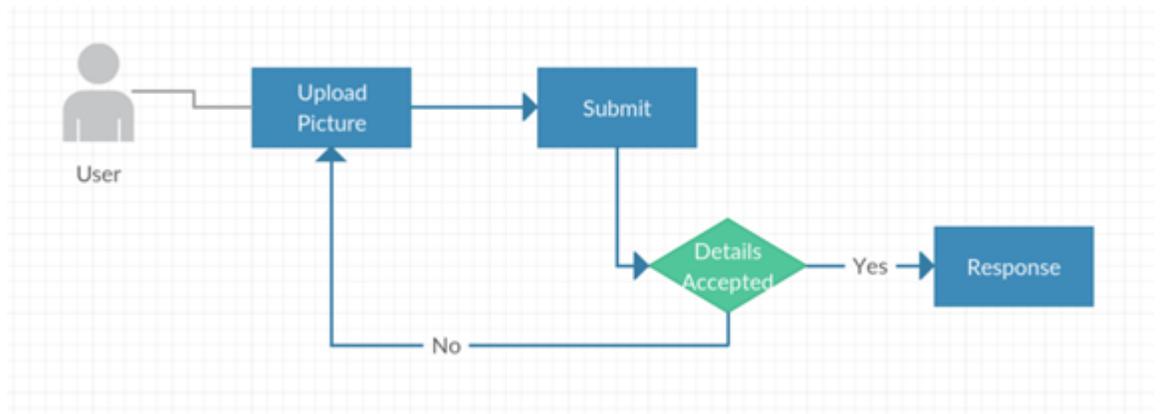
Scope

The scope of this use case is to describe how a user can use image recognition to identify the species of a fish.

Description

This use case describes the process of using the image recognition function.

Use Case Diagram



Flow Description

Precondition

Requirement 2: User Login

Activation

This use case starts when a user has logged in and select the Specie Recognition tab

Main flow

1. The user uploads a picture of fish.
2. The user hits submit button.
3. The system responds with the species of the fish.

Alternate flow

1. The user uploads a picture that does not fit requirements.
2. The user clicks the submit button
3. The submission has been unsuccessful
4. The user is asked to follow the requirements of the picture.

Termination

The system successfully submits the users image and gets a response of the correct species of fish.

Post condition

The user now knows the species of fish

2.1.2.4 Requirement 5: Best Fishing Date

Description & Priority

Pattern recognition will be an essential part of the application the all users.

Use Case

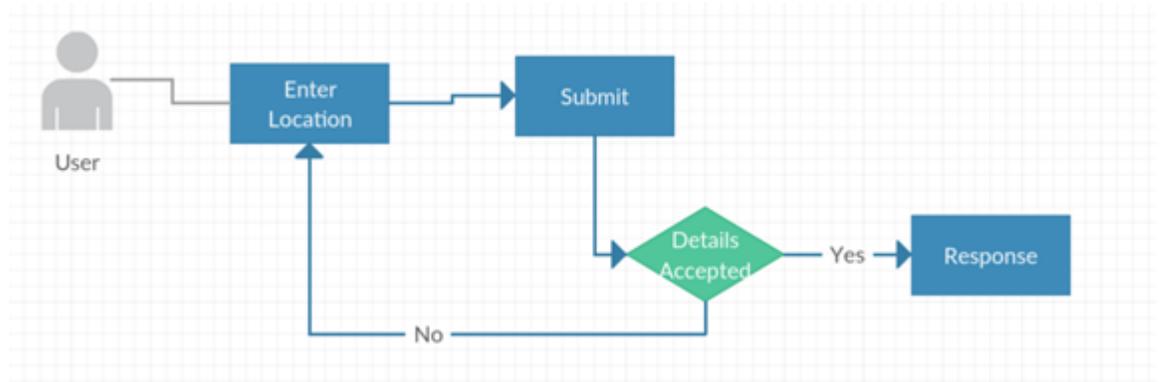
Scope

The scope of this use case is to describe how a user can use pattern recognition to find the best day to go fishing.

Description

This use case describes the process of using the pattern recognition function.

Use Case Diagram



Flow Description

Precondition

Requirement 2: User Login

Activation

This use case starts when a user has logged in and selects the Best Fishing Date tab.

Main flow

1. The user selects a location
2. The user hits submit button
3. The system responds with forecast data and best day.

Alternate flow

1. The user selects a location
2. The user hits submit button
3. The system responds with error message such as could not find location.

Termination

The system successfully submits, and the user is told what the best day is to go fishing along with all other information needed.

Post condition

The user knows when to go fishing.

2.1.2.5 Requirement 6: User Dashboard

Description & Priority

The user dashboard will be essential for users to navigate through the web application.

Use Case

Scope

The scope of this use case is to describe the User Dashboard

Description

This use case describes the process of landing on the user dashboard

Use Case Diagram

Flow Description

Precondition

Requirement: User Login

Activation

This use case starts when a new user visits the website

Main flow

1. The user has logged in successfully.
2. The user lands at the dashboard.

Alternate flow

1. The user enters incorrect login information

2. The user remains at login screen

Termination

The dashboard loads correctly.

Post condition

The user will now be able to navigate through the web application with ease.

2.1.3 Non-Functional Requirements

2.1.3.1 Performance/Response time requirement

This requirement defines the performance of the system. To ensure that the user is happy with the system the image recognition algorithm must be trained to the greatest extent possible. This will create a quicker response from the server. The processing power of the ec2 instance could also affect the loading times.

2.1.3.2 Security requirement

This requirement defines the security requirements. User information will need to be encrypted when passing from the node.js server to our mongo database for users to trust the system. Read permissions will also have to be created if a user wants to keep his catch's private.

2.1.3.3 Data requirements

Data from users will be sent from the web page to the node.js server which will then send and store the information in the MongoDB database. All data will be sent and received as JSON.

2.1.4 Data requirements

2.1.4.1 Database Design

For this project, I have chosen to use MongoDB a NoSQL database. I will host the on mlab.com. This database will contain a decent amount of information as it will contain users login information and users posts which can contain high-quality images.

Database Details:

Database provider:	mLab cloud hosted MongoDB
Name of the database:	Total_Fishing
Location of database:	<a href="mongodb://<dangil>:<12345qwerty>@ds119160.mlab.com:19160/total-fishing">mongodb://<dangil>:<12345qwerty>@ds119160.mlab.com:19160/total-fishing
Database style:	NoSQL
Data Format:	JSON

MongoDB is a NoSQL database as such all data is stored as JSON object documents. There are no tables, columns, rows or records like SQL databases. There can be some advantages to NoSQL databases such as faster searching, easier to upgrade, Object-oriented language that is easy to understand.

I have created two main schema models for my database a User and a Post

User Schema:

```
var UserSchema = mongoose.Schema({
  username: {
    type: String,
    index: true
  },
  password: {
    type: String
```

```

    },
    email: {
      type: String
    },
    name: {
      type: String
    }
  }
});

```

An example of the JSON stored in database for users:

```

{
  "_id": {
    "$oid": "5af43d2a2bf1440014729f97"
  },
  "name": "Admin",
  "email": "admin@admin.com",
  "username": "Admin",
  "password": "$2a$10$wgZzwM67lMYX3QyPvR.qYeqPY9CLRgAe.SS1gT0SgY9UULiKHtWEa",
  "__v": 0
}

```

As you can see each user is stored with a unique id which is automatically generated for us thanks to mongoose. The rest of the object is pretty ordinary with the name, email and username being stored in strings and the password being stored as a hash by using the bcrypt node package.

Post Schema:

```

var PostSchema = mongoose.Schema({
  creator: {
    type: String
  },
  species: {
    type: String
  },
  weight: {
    type: Number
  },
  method: {
    type: String
  },
  description: {
    type: String
  }
});

```

```

    },
    img_name: {
      type: String
    },
    location: {
      type: String
    },
    privacy: {
      type: Boolean
    },
    likes: {
      type: Number
    },
    comments: [{
      body: String,
      user: String,
      date: Date
    }]
  });

```

an example of the JSON is stored in the database:

```

{
  "_id": {
    "$oid": "5af392c34eb050442068a345"
  },
  "creator": "DanGil96",
  "specie": "Roach",
  "weight": 15,
  "method": "Bait",
  "description": "Caught on spinnerbait",
  "img_name": "219e0e85818fba6e2deadde56c93292f.jpg",
  "location": "Ballynakill, Mornington, Co. Westmeath, Ireland",
  "privacy": true,
  "likes": 0,
  "comments": [
    {
      "_id": {
        "$oid": "5af45fca5501a20014e1e7e7"
      },
      "body": "Nice catch man",
      "user": "Admin"
    }
  ],
  "__v": 0
}

```

The post schema is somewhat more complex than the user schema. The creator value is the username of the user which is grabbed upon submitting the form. The img_name is the new name of the image after it is stored in the database, this makes it easy to retrieve the images for the front end. As we can see comments is stored as an array of objects with two strings inside. The unique id is used to find and update/delete comments.

2.1.5 User requirements

- Users should be able to register through any browser.
- Usability, users should be able to navigate through the website easily on any device.
- Users should be able to post to the community or decide to keep it private.
- Users should be able to identify the species of a fish by using the image recognition functionality.
- Users will have access to a simple maps API to import their location
- Users will have access to a simple weather API to import current weather from the location.
- Users should be able to use the predict what day will be best to go on their next fishing trip.

2.1.6 Environmental requirements

- Internet Access: This web application requires internet to access the system. The image recognition requires fast upload speed to have good loading times.

2.1.7 Usability requirements

2.1.7.1 Functional Requirements

Requirement	View Dashboard
Number	1
Description	Users should be able to view their dashboard with all users posts up to date
Rationale	Users should be able to have a dashboard with the most up to date "news feed" to achieve a community.
Success criteria	The data gathered should be summarized and displayed in a well-structured manner, should provide vital information on the user's posts.
Level of Importance	5 High, this is a basic requirement

Requirement	Post Catch
Number	2
Description	Users should be able to post a catch to the dashboard or their private dashboard.
Rationale	Users should be able to post to a dashboard to view the catches at another time.

Success criteria	A user should find it very easy and quick to post. They should also be able to add an optional description to their post.
Level of Importance	5 High, this is a basic requirement

Requirement	Image Recognition
Number	3
Description	The user can post an image to the image recognition system and receive a response.
Rationale	Users should be able to verify the species of a fish using the image recognition system.
Success criteria	A user should find it very easy and quick to use the image recognition system. It should have a high accuracy rate and display information about the species.
Level of Importance	5 High, this is a basic requirement

Requirement	Best Fishing Date
Number	4
Description	The user can find out what day is best to go fishing.

Rationale	Users should be able to use the best fishing date section to plan a day to go fishing.
Success criteria	A user should find it very easy and quick to use the best fishing date system. It should display the 5-day forecast in an easy to view, well-summarized manner.
Level of Importance	5 High, this is a basic requirement

Requirement	Error Handling
Number	5
Description	Users should be notified of errors instead of being brought a 500 or 404 page.
Rationale	Errors in-app should be user-friendly.
Success criteria	A user should be redirected to a page with a flash error message if an error is received in the application. It should describe the error and how to fix it if possible.
Level of Importance	5 High, this is a basic requirement

2.1.8 Non-Functional Requirements

Performance/Response time

The system should be designed with user experience in mind, page loading and response time should be optimal saving Users from unnecessary frustration and

when wait times are expected they should be clearly and correctly communicated to users.

Responsiveness Requirement

The system should be designed with responsiveness as the main priority. Users should find this system easy to achieve their task when using their mobile devices.

Security Requirement

The system should be designed with access controls measures in place, as users should not be able to edit/delete other users posts, or comments.

Concurrency Requirement

This system should incorporate concurrency as an essential requirement. Users should be able to create posts or post comments at the same time without being hindered by errors. This will prevent users from being in deadlock when performing multiple actions.

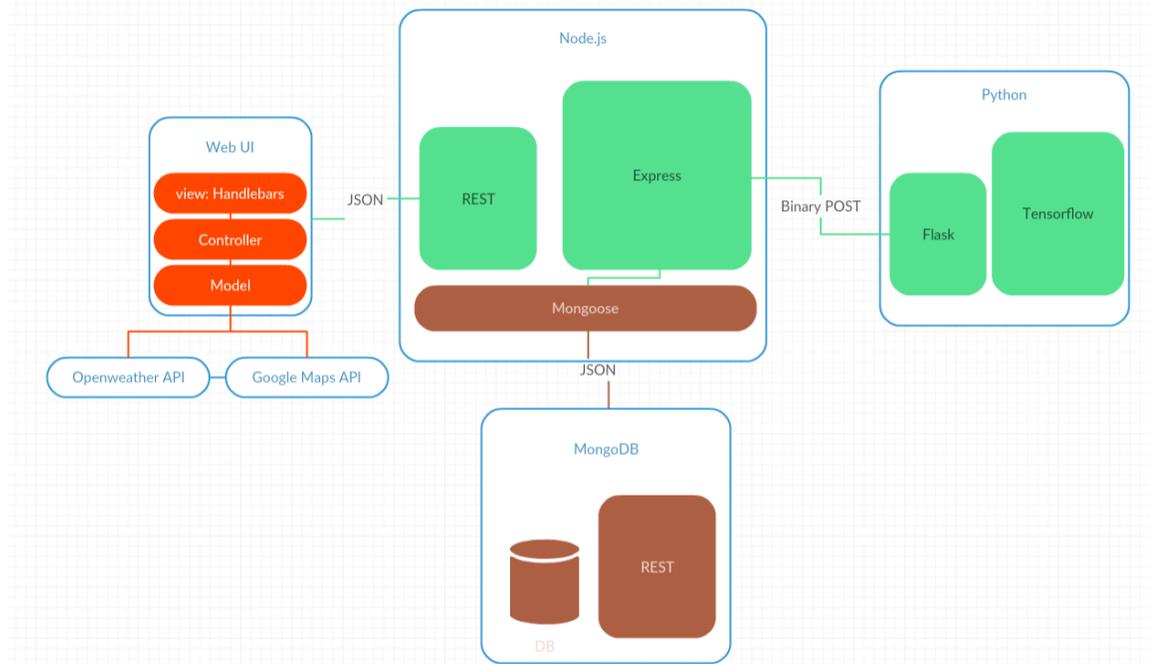
Scalability

The system should be scalable, as the number of users and posts increases the system should scale with it to ensure fast loading times.

Internationalization

Thankfully because this is a web application and most current browsers have an automatic translate function built in. Although this application will be targeting users in Ireland, there will still be a chance of a user that does not speak English.

2.2 Design and Architecture



Web UI implements the MVC model. I choose to use handlebars as my view templating engine as it was the most relatable engine for me. Other engines like jade and pug do not resemble HTML enough for me. Handlebars can be sent variables from the backend. This makes it easy to check if a user is logged in or not and what to display if they are. I created a helper method for handlebars to check if statements as it was not implemented in the original functionality. This helped me check if a post was set to private or if the logged on user was the creator of the post etc. I used ajax to communicate with the Openweathermap API and javascript for Google maps API.

Node.js was chosen as it is a fairly new and up and coming technology which I had experience with from my work placement. Node.js has a very big development community with millions of node packages available to install. Express was the framework I used with node.js which made it easy to route through the application and display the frontend while communicating with it.

MongoDB was used as the database as it is well adapted for use with node.js with the node package mongoose. Mongoose contains documentation on how to

set up the schemas and model methods which were very helpful for me. It is also a NoSQL database which has some advantages such as easier to upgrade and object-oriented programming which is easier to use.

At the time Tensorflow was mainly only implemented for use with Python although some time through development TensorFlowjs was released which was able to perform actions on the client side which would have been interesting to see the results. I used tensorflow to train my data which contained over 80 pictures of each species. I then created an API and implemented tensorflow to be able to call it from my node.js server. This API was created using Flask minimalist framework.

2.3 Implementation

The web application was developed using Node.js, Handlebarsjs, MongoDB, Javascript, JQuery, CSS and Python. Python was used to train the image recognition system and create the API which is pure backend. Node.js acts as a backend but produces the front end as well.

2.3.1 Back-End Development

Node.js:

Firstly I will go through the core of the server. The app.js which makes everything tick. Firstly we import all of the node packages that we need. We then create a connection to the database initialize a stream to retrieve images and create a storage engine:

```
//MongoDB Connection
mongourl = 'mongodb://dangil:12345qwerty@ds119160.mlab.com:19160/total-fishing';
mongoose.connect(mongourl);
var db = mongoose.connection;
// Create mongo connection
const conn = mongoose.createConnection(mongourl);
// Init gfs
let gfs;
```

```

// Init stream
conn.once('open', () => {
  gfs = Grid(conn.db, mongoose.mongo);
  gfs.collection('uploads');
})

// Create storage engine
const storage = new GridFsStorage({
  url: mongourl,
  file: (req, file) => {
    return new Promise((resolve, reject) => {
      crypto.randomBytes(16, (err, buf) => {
        if (err) {
          return reject(err);
        }
        const filename = buf.toString('hex') + path.extname(file.originalname);
        const fileInfo = {
          filename: filename,
          bucketName: 'uploads'
        };
        resolve(fileInfo);
      });
    });
  }
});
const upload = multer({
  storage
});
const type = upload.single('file')

```

Express makes it very easy for application to receive any type of request. For an example I will show you a section of the image recognition request. I will explain the code throughout the comments

```

// Upload Image and send binary post request to image classification API
app.post('/upload', type, function(req, res, next) {
  //in the req we receive the file and information about it from the user from the
  //user. We save the name of the file to a variable
  var file = req.file.filename;

  //findOne a predefined method from grid-fs. This will search for the filename
  //matching and create a readstream to pipe it to tensorflow flask api url.
  gfs.files.findOne({ filename: req.file.filename}, function (err, file) {
    console.log(file);
    if (err) {
      console.log(err);
    }
  });
});

```

```

}
var readstream = gfs.createReadStream(file.filename);
readstream.pipe(request
  .post("http://35.164.132.68/classify")
  .on('response', function(response) {

//On the response we set some headers and we save the response data to a string and
we parse the string to a json object.

    response.setEncoding("UTF-8");
    response.on('data', function(data) {
      console.log(data);
      var jsonString = data.toString('utf8');

//If the string contains the word tench in it then continue
//change accuracy to get a percentage
//It will then check if the accuracy is higher than 60% if so render the tench page
//else reload the image recognition page with an error message

      if (jsonString.includes("tench")) {
        var json = JSON.parse(jsonString);
        var accuracy = json.predictions.tench;
        accuracy = Math.floor(accuracy* 100)
        accuracy = accuracy.toString().replace("0.", "");
        if (parseInt(accuracy)>60) {
          res.render('tench', {
            accuracy: accuracy
          });
        }else {
          req.flash('error_msg', 'Could recognise specie of fish, please ensure
that it is a clear picture of the fish.');
```

To allow a user to create a post, there had to be a post model created which contained the contents that would be saved to our database. With this we had to create methods to create posts, create comments, delete posts, delete comments and to update likes.

First we define the schema of the post:

```

// Post Schema
var PostSchema = mongoose.Schema({
  creator: {
```

```

    type: String
  },
  specie: {
    type: String
  },
  weight: {
    type: Number
  },
  method: {
    type: String
  },
  description: {
    type: String
  },
  img_name: {
    type: String
  },
  location: {
    type: String
  },
  privacy: {
    type: Boolean
  },
  likes: {
    type: Number
  },
  comments: [{
    body: String,
    user: String,
    date: Date
  }]
});

```

Then we must export the model to be used in other files and create our methods that will be needed :

```

module.exports.createPost = function(newPost, callback) {
  newPost.save(callback);
}

module.exports.findAndUpdateLikes = function(_id, likes, callback) {
  var query = {
    _id: _id
  };
  var numlike = parseInt(likes);

```

```

numlike = numlike + 1;
Post.findOneAndUpdate(query, {
  $set: {
    likes: numlike
  }
}, callback)
}
module.exports.findAndUpdateComments = function(_id, comment, user, callback) {
  var query = {
    _id: _id
  };
  var date = new Date();
  var comments = {
    "body": comment,
    "user": user,
    "date" : date
  };
  Post.findOneAndUpdate(query, {
    $push: {
      comments: comments
    }
  }, callback);
}
module.exports.findAndDeleteComment = function(_id, postid, callback) {

  Post.update({
    _id: postid
  }, {
    "$pull": {
      "comments": {
        "_id": _id
      }
    }
  }, callback);
}
module.exports.findAndDeletePost = function(_id, callback) {
  var query = {
    _id: _id
  };
  Post.findOne(query).remove(callback);
}

```

We can then call one of these exported functions upon request. For example when the app receives a post request from the front end containing `/comment/:id` we can call the function.

```

// Post a comment
router.post('/comment/:_id', ensureAuthenticated, function(req, res, done) {
  var _id = req.params._id;
  var comment = req.body.comment.toString();
  var user = req.user.username;
  console.log(user);
  console.log(_id);
}

```

```

//Gather all needed info above and save to variables then call function passing these
//variables and get response
Post.findAndUpdateComments(_id, comment, user, function(err, call) {
  if (err) {
    req.flash('error_msg', 'Something went wrong!');
    res.redirect("/");
  }

  console.log(call);
  req.flash('success_msg', 'You have posted a comment!');
  res.redirect("/");
});
});

```

When implementing the API, I chose flask a minimalistic framework as I did not see much point in choosing Django a heavy feature framework. Tensorflow gives premade files classify an image based on your trained_graph.pb and trained_labels.pb these two files are produced after we running the training file. The training file is a shell file which executes the retrain.py and sets some values such as how_many_training_steps, directory of image dataset etc:

```

python retrain.py\
  --bottleneck_dir=tf_files/bottlenecks \
  --how_many_training_steps=8000 \
  --model_dir=inception \
  --summaries_dir=tf_files/training_summaries/basic \
  --final_tensor_name=final_result \
  --output_graph=tf_files/retrained_graph.pb \
  --output_labels=tf_files/retrained_labels.txt \
  --image_dir=training_dataset \

```

When we have created our two files retrained_graph and retrained_labels we can start to create the API, we must build a basic flask API and import some of Tensorflows classify methods to run the image recognition on the given image file. I chose to handle a binary post, this way I could simply pipe a post request with the image from my node.js server. This was done with the request node package. Firstly I set the response to output one prediction as that is all we need. I also set the path to the two files which I had to edit when my files where on the ubuntu ec2 instance.

```

FLAGS = tf.app.flags.FLAGS

tf.app.flags.DEFINE_string(

```

```

'model_dir', './',
"""Path to retrained_graph.pb, """
"""retrained_labels.txt""")

tf.app.flags.DEFINE_integer('num_top_predictions', 1,
                            """Display this many predictions.""")

```

Then we can handle the post request. We can grab a lot of this code from tensorflow and change it to suit our method. So firstly we must create a graph from the saved retrained_graph.pb

```

def create_graph():
    with tf.gfile.GFile(os.path.join(
        FLAGS.model_dir, 'retrained_graph.pb'), 'rb') as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())
        _ = tf.import_graph_def(graph_def, name='')

```

We then must load up the node_lookup which will convert the node ID to a readable string which are in our retrained_labels.txt. After this, we can run tensorflow inference on the image and print out the result in a JSON format and handle if there is an error.

```

@app.route("/classify", methods=["POST"])
def classify():

    create_graph()
    print("Model loaded")

    node_lookup = NodeLookup()
    print("Node lookup loaded")

    predictions = dict(run_inference_on_image(request.data))
    print(predictions)
    return jsonify(predictions=predictions)

@app.errorhandler(404)
def not_found(error):
    return make_response(jsonify({'error': 'Not found'}), 404)

```

Run Inference on the image:

```

def run_inference_on_image(image_data):
    """Runs inference on an image.
    Args:
        image_data: Image data.
    Returns:
        Nothing
    """
    # Set the tensorflow session to not use GPU this was to run on an ec2 instance
    config = tf.ConfigProto(device_count = {'GPU': 0})
    sess = tf.Session(config=config)
    print("Tensorflow session ready")
    node_lookup = NodeLookup()
    print("Node lookup loaded")
    # Runs the softmax tensor by feeding the image_data as input to the graph.
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')
    predictions = sess.run(softmax_tensor, {'DecodeJpeg/contents:0': image_data})
    predictions = np.squeeze(predictions)

    # sort predictions to make sure the highest result is outputted
    top_k = predictions.argsort()[-FLAGS.num_top_predictions:][::-1]

    # map to the friendly names and return the tuples
    return [(node_lookup.id_to_string(node_id), float(predictions[node_id])) for
            node_id in top_k]

```

To run the app on the ec2 and get a response from it externally I needed to run the app on the localhost port 80.

```
app.run(host='0.0.0.0' port=80 debug=true)
```

I then started a tmux window on the ec2 so the application would continue to run when I closed the ssh connection. I achieved this by running this command.

```
tmux new -s mywindow
```

2.4 Graphical User Interface (GUI) Layout

Mobile Layout:

- Login

13° 37839.herokuapp.com 2 14:31

Total Fishing

Account Login

Username

Password

Submit

© Total Fishing, Inc.

- Register

13° 37839.herokuapp.com 2 14:32

Total Fishing

Register

Name

Username

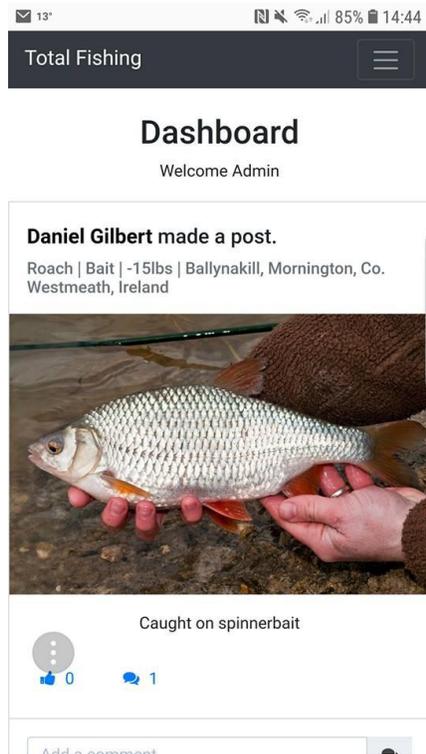
Email

Password

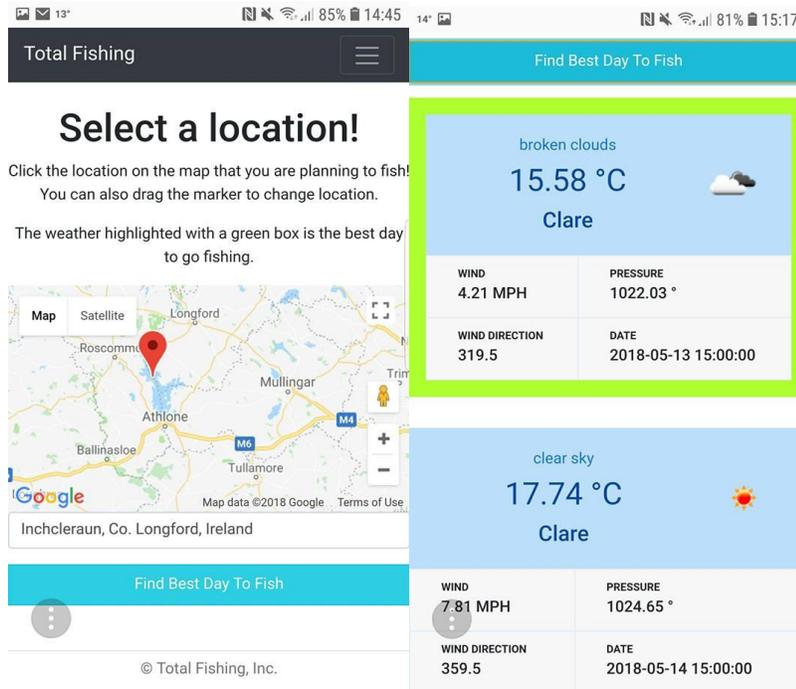
Confirm Password

Submit

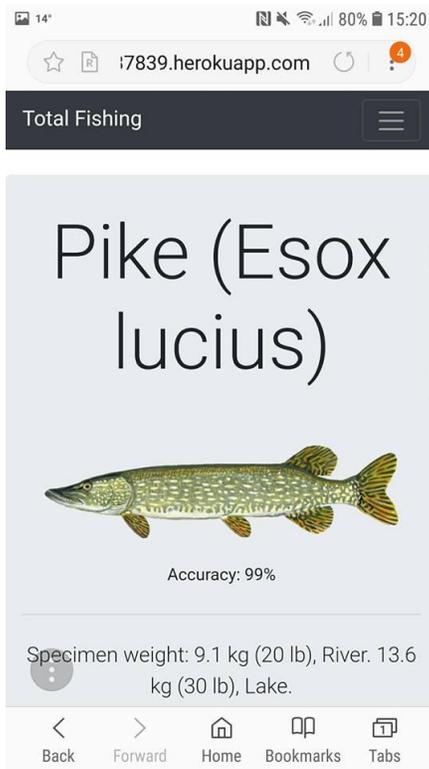
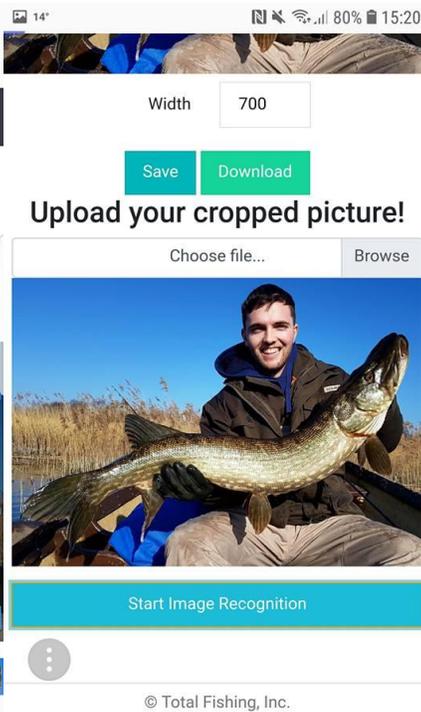
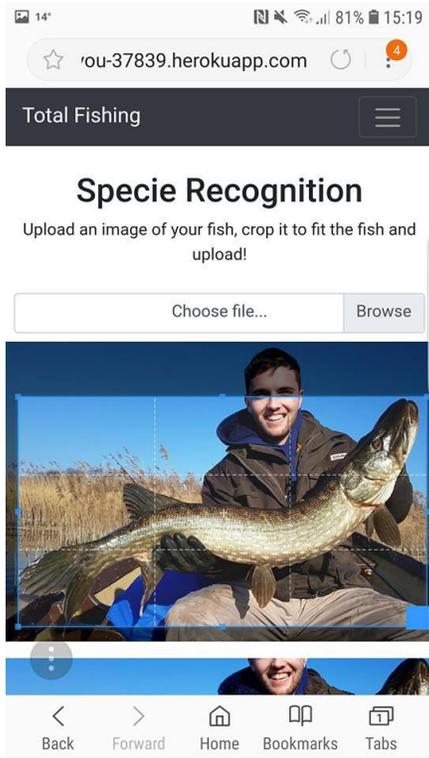
- Dashboard



- Best Fishing Date



- Image Recognition



- Post Catch

A screenshot of a mobile browser displaying the 'Post Catch' form on the Total Fishing website. The browser's address bar shows 'rou-37839.herokuapp.com'. The page title is 'Total Fishing'. The form is titled 'Information on your catch!' and includes the following fields: 'Specie of fish' (a dropdown menu with 'Pike' selected), 'Fish Weight' (a text input field with the placeholder 'Enter fish weight in lbs'), 'Method Of Catch' (a dropdown menu with 'Lures' selected), and 'Description (optional)' (a large text area). At the bottom, there is a file upload section with a 'Choose file...' button and a 'Browse' button. The mobile navigation bar at the very bottom includes icons for Back, Forward, Home, Bookmarks, and Tabs.

Desktop Layout:

- Login

A screenshot of a desktop browser displaying the login page on the Total Fishing website. The page has a dark header with 'Total Fishing' and links for 'Login' and 'Register'. A pink error message at the top states 'You are not logged in'. The main heading is 'Account Login'. Below this, there are two input fields: 'Username' with the value 'DanGil96' and 'Password' with masked characters '.....'. A grey 'Submit' button is located at the bottom of the form.

© Total Fishing, Inc.

- Register

Total Fishing [Login](#) [Register](#)

Register

Name

Username

Email

Password

Confirm Password

© Total Fishing, Inc.

- Dashboard

Dashboard

Welcome DanGil96

 **Daniel Gilbert** made a post.
Roach | Bait | -15lbs | Ballynakill, Mornington, Co. Westmeath, Ireland



Caught on spinnerbait

 0  1

 **Admin**

- Best Fishing Date

Total Fishing Dashboard Private Catches Best Fishing Date Fish Recognition Post Catch Logout

Select a location!

Click the location on the map that you are planning to fish! You can also drag the marker to change location.

The weather highlighted with a green box is the best day to go fishing.

Ulicksmountain, Co. Galway, Ireland

Find Best Day To Fish

Find Best Day To Fish

light rain 14.18 °C Galway		clear sky 15.15 °C Galway	
WIND 4.8 MPH	PRESSURE 1021.34 °	WIND 3.7 MPH	PRESSURE 1028.06 °
WIND DIRECTION 206.502	DATE 2018-05-13 18:00:00	WIND DIRECTION 242.501	DATE 2018-05-14 18:00:00

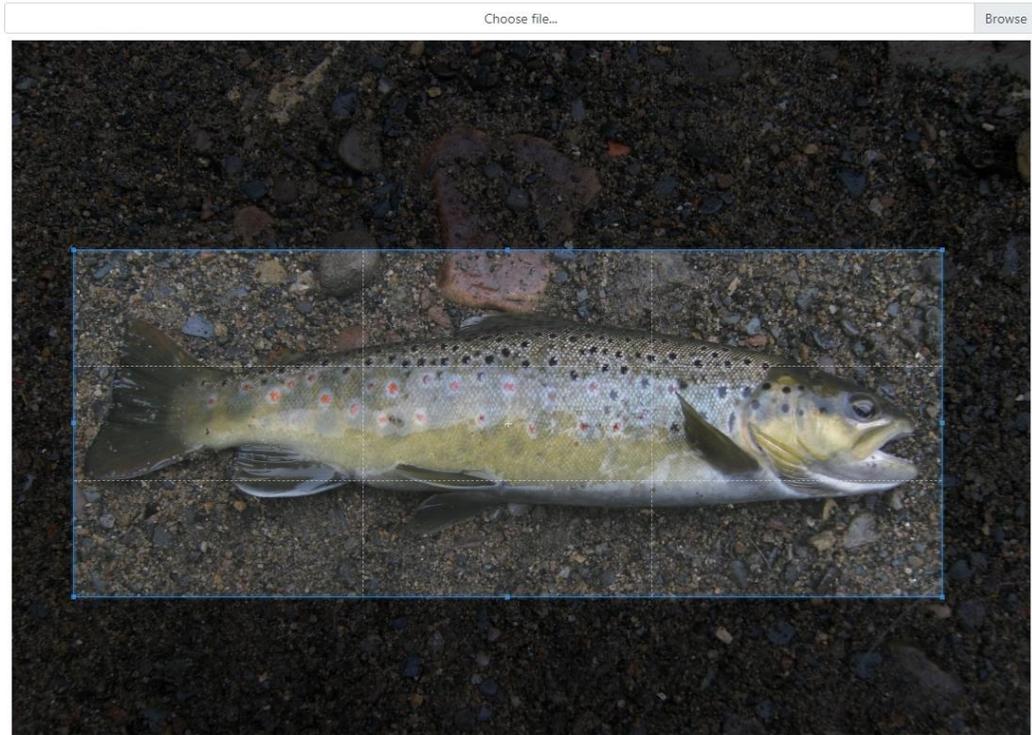
overcast clouds 10.36 °C Galway		clear sky 14.18 °C Galway	
WIND 8.17 MPH	PRESSURE 1035.46 °	WIND 4.66 MPH	PRESSURE 1038.04 °
WIND DIRECTION 356.501	DATE 2018-05-15 18:00:00	WIND DIRECTION 8.50314	DATE 2018-05-16 18:00:00

broken clouds 14.21 °C Galway	
WIND 2.34 MPH	PRESSURE 1033.38 °
WIND DIRECTION 264.501	DATE 2018-05-17 18:00:00

- Image Recognition

Specie Recognition

Upload an image of your fish, crop it to fit the fish and upload!





Width

[Save](#) [Download](#)

Upload your cropped picture!

[Browse](#)



[Start Image Recognition](#)

Total Fishing [Dashboard](#) [Private Catches](#) [Best Fishing Date](#) [Fish Recognition](#) [Post Catch](#) [Logout](#)

Brown Trout (Salmo Trutta)



Accuracy: 99%

Specimen weight: 2.268 kg (5 lb).

Record weight: 26.2lb Lough Ennell, 1894.

The brown trout (*Salmo trutta*) or Breac Donn (Gaelic) is a native Irish species and the most widely distributed freshwater fish in Ireland. It thrives in waters of all types, from small mountain streams to broad limestone rivers and loughs. It's main requirements are clean water and gravel in which to spawn. Ireland has 16,000 km of main river channel and 10,000 km of tributary which are unspoilt and relatively unpolluted. They provide ideal habitat for trout. In addition, there are in excess of 500,000 acres (200,000 ha) of loughs.

Most brown trout fisheries open between February 15th and March 17th. Most close on September 30th with some exceptions which close on various dates between September 15th and October 12th. Clubs may have their own regulations on opening and closing dates.

The majority of waters are owned either privately or by the State. Many are leased to angling clubs or associations. A fishing permit issued by the owner or the lessee gives the permit holder the right to fish for set periods ranging from a day to a season. In the Republic of Ireland, permits for the best waters generally cost between €10 and €20 a day. Many of the big western loughs offer free fishing for brown trout. Some fisheries such as Loughs Mask, Corrib, Conn, Derg, Carra, Ree, Cullin, and Arrow do not

- Post Catch

Information on your catch!

Specie of fish
Pike

Fish Weight
Enter fish weight in lbs

Method Of Catch
Lures

Description (optional)

Choose file... Browse

Location of Catch

A map showing the location of the catch. The map covers Ireland, the Isle of Man, and parts of northern England. Major cities like Dublin, Galway, Limerick, and Manchester are labeled. The map includes road networks and geographical features like national parks.

2.5 Testing

I carried out extensive testing throughout the project and believe that it is at a stage now where error handling is well laid out throughout the app, and usability design is good. To get to this stage, I carried out testing techniques such as Think Aloud Test and Heuristic Evaluation.

2.5.1 Think Aloud Test

The thinking aloud usability availability testing, involves asking test participants to use your system while continuously talking aloud, expressing their feelings or thoughts as they navigate through the systems interface completing a series of given tasks.

To carry out the think-aloud test I developed a series of tasks relating to the web application that the test participants were asked to carry out, the questions are outlined below. After the tests had been completed we asked the test

participants to fill out a System Usability Scale (SUS) form; the results can be found in the appendix.

Questions asked:

- Can you locate and register as a user
- Can you create a post as this user
- Can you add a comment to this post
- Can you delete your post
- Can you use the best fishing date in Dublin
- Can you use the image recognition system with provided picture

Think aloud test results analysis

The testing process went well, except for the issue that occurred with one of the participants expecting the add post button to be on the dashboard page and not in the navbar.

The majority of the test participants found the web application simple and intuitive to use; they also felt they would easily be capable of using the site without any assistance, which I feel is an excellent result as this is what I aimed for.

A breakdown of the System usability scale can be seen below; the SUS form can be seen as a general measurement of the usability of the site, a site with a score below 68 usually means there is a serious issue with the usability of the systems interface.

Test participants (TP) SUS scores:

TP1 - 100

TP2 - 95

TP3 - 84

TP4 - 82.2

With the following scores, we can see that there doesn't seem to be any serious issues with the usability of our website. There may be a small issue with the location of the Post button, but this can be reinvestigated to see if a possible solution exists that may help to improve the overall usability of the website.

2.5.2 Heuristic Evaluation

Heuristic evaluation is a discount method for quick, cheap and easy evaluation of a user interface. It requires a set of testers to examine and judge the interface according to the recognised usability principles. Heuristic evaluation is a prevalent method of testing as it requires minimal resources. Heuristic evaluation is known to find more than 90% of usability problems. We will have five individual testers to evaluate our website. They will follow these usability heuristics:

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users recognise, diagnose, and recover from errors
- Help and documentation

2.5.2.1 Visibility of system status

Visibility of System Status is concerned with the questions “Where am I now?” and “Where can I go next?”.

Our testers expressed that With the Titles of every page it was quite simple to see where they were although they would like if the navbar was highlighted on the current page.

2.5.2.2 Match between the system and the real world

The system should speak the users' language using words, phrases and concepts that are familiar to the user.

Our testers spoke about how the website felt comfortable to use because of the use of bootstrap.

2.5.2.3 User control and freedom

This principle talks about giving the user the freedom to navigate and perform actions. The freedom to undo any accidental actions.

The testers express that the alert to ensure that they want to delete their post/comment was a valuable insight to the page

2.5.2.4 Consistency and standards

Across the website, similar names and terms should be used to describe entities and concepts. If a symbol or name is used for something on one page, it should not be different on another page.

The testers ensured that there was no problem with consistency across the app and highlighted that the map marker system and image upload was the same on different pages.

2.5.2.5 Error prevention

The system should be designed in a way to limit the errors a user can see while using the website.

There was a small issue with the CSS loading in for one of the user; We believe this was to a cdn link that was down for a second because upon refresh the CSS loaded incorrectly. Other than that there were no errors.

2.5.2.6 Recognition rather than recall

Make sure objects, actions, and options are highly visible. Website visitors should not have to remember information between different parts of their dialogue with the site.

None of our testers expressed any issues with remembering what different elements did, or meant.

2.5.2.7 Flexibility and ease of use

Flexibility and ease of use are the optimisations of a system to benefit the needs of a novice and an advanced user. There should be operations to speed up interactions with the system.

They express the app is simple to use and does not need any operations to speed up interactions with the system. They highlighted how fast the image recognition system was to respond

2.5.2.8 Aesthetic and minimalist design

Extraneous information on a page is a distraction and a slow-down. The system should only have information that is needed.

The testers were happy with our design on the site.

2.5.2.9 Help users recognise, diagnose and recover from errors

If errors appear, the problem should be precisely indicated, and a solution should be provided with it if possible.

Testers ensured that error handling on the site was brilliant, instead of being brought to a 404 page they were redirected home with an error message on top of the screen highlighting what to do next time, so it does not happen.

2.5.2.10 Help and documentation

Although our testers were able to easily navigate through our page without the need of help or documentation. They expressed it would not hurt to have an FAQ section or Contact us page.

3 Conclusions

I enjoyed working on this application. It is different to work on something that you enjoy as a hobby in life. Although I do feel there are some advantages and disadvantages to the workings of the project. There was very little information on hosting an API to interact with tensorflow, and there was even less information on what was needed to host it. I feel like the use of node.js and MongoDB were a huge success in the application. Node.js and express made it very easy to create all the views and Handlebars made it easy to implement checks in the HTML. MongoDB and Mongoose were perfect, and I found all the documentation I needed on the Mongoose site.

In the end, the flask API worked out very well, and I was surprised with the speed of the AWS ec2 instance as it gives responses much faster than expected. The accuracy of the image recognition was impressive. If I had more time, I would add more species to the image recognition, and I would create a python application to generate a weighted scoring model instead of creating it with the use of graphs from the dataset.

4 Further development or research

With more resources, the species list for the image recognition is endless, with a powerful enough set of GPUs very accurate graph could be trained. Also with a larger dataset of images per fish species would increase the accuracy even more. If a python application was created to rebuild a weighted score model every month based on users input, I believe that it would lead to a very successful predictor.

5 References

Abadi, M. (2016). *TensorFlow: A System for Large-Scale Machine Learning*.
[online] Savannah. Available at:

<https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf>
[Accessed 3 Mar. 2018].

Cleary, F. (2015). *Running a Flask app on AWS EC2*. [online] Datasciencebytes.com. Available at:
<https://www.datasciencebytes.com/bytes/2015/02/24/running-a-flask-app-on-aws-ec2/> [Accessed 1 Apr. 2018].

Docs.mongodb.com. (2018). *MongoDB Documentation*. [online] Available at:
<https://docs.mongodb.com/> [Accessed 2 Mar. 2018].

Holland, J. (2010). *Adaptation in natural and artificial systems*. Cambridge, Mass. [u.a.]: MIT Press.

Ireland, I. (2018). *Fishing in Ireland. An angler's guide to the bestfishing in Ireland..* [online] Fishinginireland.info. Available at:
<http://fishinginireland.info/> [Accessed 9 Mar. 2018].

Mongoosejs.com. (2018). *Mongoose v5.1.0: Schemas*. [online] Available at:
<http://mongoosejs.com/docs/guide.html> [Accessed 14 Apr. 2018].

Siraj Raval (2018). *Tensorflow*. [podcast] Available at:
<https://www.youtube.com/channel/UCWN3xxRkmTPmbKwht9FuE5A>
[Accessed 8 Feb. 2018].

TensorFlow. (2018). *API Documentation | TensorFlow*. [online] Available at:
https://www.tensorflow.org/api_docs/ [Accessed 5 Mar. 2018].

Udeogu, I. (2017). *How to create an image recognition web-service — Part I*. [online] Medium. Available at: <https://medium.com/@innarticle/how-to-create-image-recognition-web-service-23554bc967bb> [Accessed 10 Mar. 2018].

WIKSTRÖM, J. (2015). *Evaluating supervised machine learning algorithms to predict recreational fishing success*. [online] Stockholm. Available at:

<https://www.diva-portal.org/smash/get/diva2:851477/FULLTEXT01.pdf>
[Accessed 4 May 2018].

6 Appendix

6.1 Project Proposal

3 Objectives

- The application should be able to create a fishing community where users can share pictures and other details about the catches.
- The application will be aimed at freshwater fishing only.
- A user should be able to use a picture recognition AI to name the species of the fish.
- The application will use a deep neural network to recommend users the best time/day to go fishing, the best method to use and the best fish to fish for. The application should become smarter over time as it will progress from each users input. This input will consist of the date, time, temperature, atmospheric pressure. It will use a weather API to grab the weather info and try to see some patterns that appear with catching certain fish.

4 Background

One of my best hobbies is fishing, I have been fishing for many years and one thing I noticed in the fishing community is a lack of technology and community. I believe that some of the older fishing generation like to keep things secret and may not like the use of technology currently, but a new generation of people are starting to fish that are more technology friendly. I believe that this could be a big gap in the market to take, as fishing is one of the biggest hobbies in the world. I believe that community where users can share there catches along with the

location, weather information, date and method used. With this information the application will get very smart over time as the deep neural network should continue to learn after everyone's input. For example, a user catches a fish in December and logs the weather info with. The AI should be able to recommend a date to go fishing based on future weather forecast. Over time the AI will recognise patterns in the weather to catch ratio and it should start to generate trends such as; When atmospheric pressure is below 1000 pike seem to feed heavily. From this we can see that temperature and rainfall don't affect these fishes feeding and the AI should be able to pick up on these things from all the users information.

5 Technical Approach

An android application must be created which will contain a database to store all the user's information. A deep neural network will be created using tensorflow which is a python framework for machine intelligence. This should be able to grab data from the database and run it through the AI to generate trends or patterns. The application should also have direct contact with the AI, as if the user would like to use the image recognition system to see what fish species they have caught. There will be an API created that will retrieve weather information which will communicate with the Deep Neural Network to plot against the future weather forecasts.

6 Special resources required

Based on some of my researched to run a big dataset through a deep neural network requires a strong GPU. I believe that the image recognition system will require a strong GPU as it won't learn as the app progresses with more information it should know the species from get go. This will require me to run a large dataset of images through it, and test it extensively for errors.

7 **Project Plan**

Gantt chart using Microsoft Project with details on implementation steps and timelines

8 **Technical Details**

Implementation language and principal libraries

- Android Studios
- Node.js
- MongoDB
- Swagger API
- Python
- Tensorflow

9 **Evaluation**

The image recognition system will be tested with real images of the fish species I will be defining.

I will use mocha to automate testing in some areas such as login, registration, and social posting.

The fishing prediction AI will be tested initially but will require real user data to improve over time.