Statflow Weather HUB

Garret Tonra

Alan Heanue

Ian Burke

B.Sc.(Hons) in Software Development

April 17, 2018

Final Year Project

Advised by: Kevin O Brien

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

About this project

Abstract For a final year project we were looking to create an industry standard application that would help people look at today's weather and to modify old data into graphs. As a team we want to design a free web application where anyone can visit to look at today's weather and data taken from met Eireann but instead implement data visualization by plotting the data into a graph, or some visual representation that's easy on the eye. We have created a full stack application with a client and server bases pulling data from multiple databases and APIs. The project will be build using a python framework called Flask but there will be a hint of neural networks to try and predict the weather or get very accurate account of old data and fire it into a Training set. There will be a google map API where we can visit any where in the world to see what the weather is like. This application will be a simple little hub and be simple to navigate through the menus.

Authors The Authors of this project are Garret Tonra, Ian Burke and Alan Heanue , Currently students studying Software Development in Galway-Mayo Institute of Technology.

Acknowledgements We would like to acknowledge and thank our supervisor Mr Kevin O Brien for the time and effort he put into helping us throughout this project. We would also like to thank all our lectures at Galway-Mayo Institute of Technology for equipping us with the skills and knowledge in the past 4 years that has enabled us to complete this project.

Chapter 1

Introduction

The rise of machine learning is a hot topic in industry at the moment. It is used commercially in data analytic , data mining and data visualization. As fourth year software development students we decided to look into this as we have a lot of interest in this area. In this section we have the introduction and a description on the project. We did look into a lot of different ideas to study and develop, although finding a source for statistics was difficult to obtain. The idea is to pull stats from a particular database or api and send it onto a dashboard embedded in a web application. The goal is to have multiple visualizations on the dashboard and to have a choice in what way you want to see the data. The user should be able to log on and be registered on a database once they have a user-name and password to log in anytime they need. When logged on they should see multiple data entry boxes. Also an empty dashboard where it should display multiple charts when the specific data is inputted into the web application. When the data is uploaded and showed on the visualization dashboard that search and result should be saved onto a database. When that is saved there should be a trend section where you can see that specific data that you have searched in a trend chart. Design will be a major factor we want to redesign the way you look at the weather data and live weather. A hub where it's easy to navigate and bring up data. We will also add a email request for further information where we will reply with an update.

The end result of this application should be a nice and easy on the eye web application. It should be quick in response to gather data and easy to navigate through the menus and be minimal in design so you can see the results instantly. The way we will approach this project will be described in the objectives below with the specific functionality detailed below.

1.0.1 Objectives of this project

The objectives of this project is to: To develop a web application that consists of the following:

- An Admin User capability
- Create a Web application
- To predict Trends Using Tensorflow
- To be Deployed on Heroku
- Store data on Database
- Save Trends on a Database
- Python scripts to connect Machine learning to Application
- Have Fully functional
- Recieve email and respond

1.0.2 Specifications

The objectives of this project is to

- Create New User
- Log in and out
- Input wanted data
- View a Dashboard
- Display Charts and stats
- Predict certain trends
- View the trends
- View older searches
- View past Trends
- Delete User
- Delete Trends

- Delete old searches
- Look At map

1.0.3 Database Specifications

The Database Should do the following:

- Store User profile
- Store User Searches
- Store Trends
- Relation between each user and data search

1.0.4 Server Specifications

- Secure routes
- Authentication
- Receive and send requests to and from database
- Receive responses from Application

1.0.5 Tensor-flow Specifications

- Apply neural Networks
- Create a Flow
- Take Stats and predict trend
- Test model and save
- Transfer data to a model

Chapter 2

Methodology

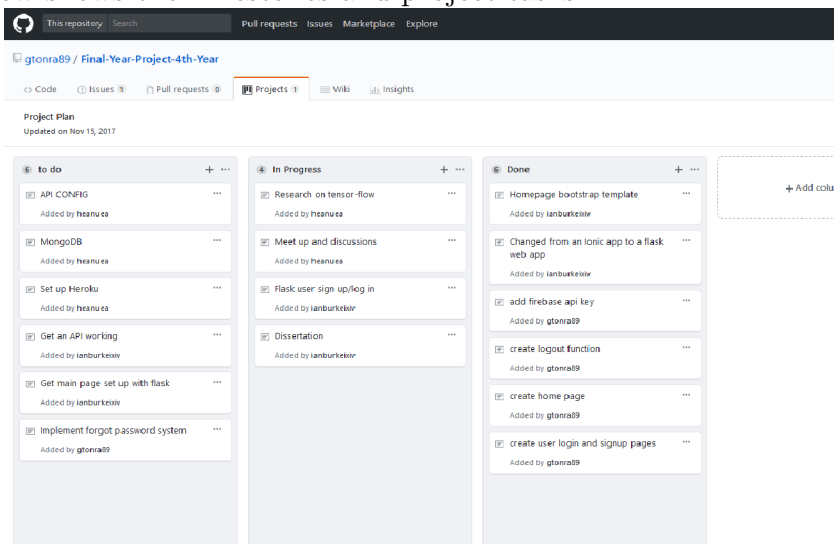
2.1 Approach and research

This section we will look at the approach we are taken to develop this project. We will discuss the methodology we adopted and describe how it was implemented into the research and development of the project. We will cover on how we approached the tasks and delegate tasks to each other. The point of this section is that a person who is reading this can have an insight on how we researched and worked together on the project and how we delivered the software to the final stages.

As we started this project without any real idea in what we were developing it looked like we were going to adapt the Ad Hoc approach. It did start that way as we drew multiple plans and did chop and change the project after deciding what were developing. But we took time and started to do meet ups (sprints), this approach fell into the methodology for this we adapted was scrum which is Agile model. Agile is more planning and project management a programmer's way to program in an agile way. Scrum is a management methodology is where we set goals weekly or daily and will have meet ups for discussion and overlook of the work we have done. The sprints are set up nearly every day now we do have multiple tools to work as a group. Google docs was set up for meetings and to write the dissertation. Facebook messenger was set to a group chat for messaging updates and github was used for the work (code). We would meet daily to see where we were at and discuss a goal or work to be carried out that day or week. We moved around certain jobs if needed and assigned different tasks if needed. We used github not just for the work but also for topics that were discussed in the meeting or in the chat groups. Here is a brief outline on the types of technologies used to work on our project:

2.1.1 GitHub

For the Project and version control we used GitHub. There are so many different project management tools available on the software market and they all do specific things. We choose GitHub because it offers a simple yet effective approach to project management. The Group was able to use GitHub to track Project tasks by creating milestones, attaching deadlines and creating issues and notes for each milestone hit or not. It also allows you to add, edit or delete deadlines and assign them to different members of the team. With GitHub Tracking work was essential to our development we sometimes had to backtrack on older repositories in order to fix a bug, with that we than were able to take record and store that as a milestone down below shows the milestones and project tasks.



2.1.2 Google Drive

Collaboration was one of the main points in this project and with students who were very busy we needed to use a very good tool to collaborate. With a good few tools out there we had a good choice. We will outline some of the collaboration tools we passed on and why. Trello is a popular web based project/task management tool it has a nice nice UX and interface and with a list item called cards are very on the eye not all of the members of the team were not keen on the idea so we passed on it. Slack was also mentioned but again none of the team had used it or familiar with the tool. We ended up staying with google drive as it was all we has used before. It was a good

amount of memory, 15GB of storage. We can edit documents and charts also there was a good clean cloud environment.

2.1.3 Sprint (monthly review)

For the duration of this project we did monthly meetings. We did meet most days but we agreed we would meet monthly and set milestones. We also had weekly meetings with our supervisor where he would guide us for time and if we needed any help. Below we outlined and took minutes of meet ups we did over course of the project.

2.1.4 Sprint (October)

The first sprint was used to get and decide on the architecture of the system and to research different technologies that could be used for each part. Each team member had a notebook and wrote down early concepts of the overall architecture and some technologies that might be used. The concepts were passed around and were put into a shortlist and voted on, but before we had discussed some of the technologies that be used and scratched off ones we were not in favour. Each sprint took an hour so we had time and we did research on some of ideas especially the tech behind it.

Sprint 2 We had to meet again in a week as we could not finalize on our idea we had a brief concept and a few technologies picked out but overall architecture was not set. We looked at some concepts and had a rough estimate on the deadline. We decided on a web app with a server and database. We also decided to add Machine learning and multiple APIs. At the review stage of the Sprint the technologies that we would be using had been chosen. More information on why each technologies used in the Project were chosen can be found under the technology review chapter.

2.1.5 Sprint (November)

The second sprint we had we did a number of tasks out and assigned them members of the team. We divided them equally and made a milestone on each task. One of the tasks in this sprint was to set up the version control which was Git and for documentation we did was set up Google docs look up at previous chapter for details on this. We agreed we would make only one branch for the git repository. Originally we were going to do multiple branches for each task but with a few members of the team were not comfortable with merging we decided this would be best. The project was divided out as follows :

- The design and user authentication was to be developed by Ian.
- The backend and polling to be done by Garrett.
- The Start of Dissertation and charts to be done by Alan.

The rest of the tasks were to monitored but held off until we got the start of project done. All members had worked on overall design and how it was laid out. The team had a few instances of Flask running to test certain APIs and Graphs.

2.1.6 Sprint (December)

This month was a challenging as for the end of semester projects and exams some of the coding and developing of the project was stalled. We did have sprints as there was much discussion on the databases we were going to include, we currently were using couchDB to poll data from an api and to put into a graph but Garret ran into issues regarding the data being pooled it crashed every few mins. So he decided to run with RethinkDb as he was familiar with this database and when tested it was smooth sailing. However the team did discuss that there was no future support for this and will this be an issue further ahead. After a brief discussion we decided to run with it as it was not the main database for the project. With this sprint as we were discussing databases we agreed we would work with mongoDB for the login for flask. The Layout of the dissertation was created and overall architecture was designed we had a presentation on this. New functionality and Api added also overall of project changed the data we were taking in was changed from sport to weather. The idea of Tensorflow being introduced was that we would include elements into the project. Story boarding and overall design was at a stage where the team were happy and some functionality was there. Most of the app was created in Python and using the Flask Framework were all the dependencies were installed. Again minor issues with rethinkDB but quickly resolved.

2.1.7 Sprint (January

This sprint was the first time we meet back after the Christmas holidays as we had not progressed as much as we wanted. The application was running and we were grabbing data from multiple api's and having charts up and running. The login was not yet completed and we had no css or bootstrap on the web application. We had discussed the research for the tensorflow models and how we can implement it within the application. Garret was

looking into a five day weather report and how that would work within the app. The different elements that we had discussed in the last sprint was working. There was still a lot to do in regards on which Database we were using, we were still using RethinkDb but still in the air weather or not we keep it. At the review of this sprint we decided we just keep the roles and just get them elements fully finished and will not add more elements to add til next sprint. We also decided to add the technology chapter to the dissertation and keep the minutes of this meeting. You can check the technology's we used in the technology review chapter.

2.1.8 Sprint (February)

The issue we had was that Json was giving us an error when we pool the data from the api. We had Garret working on that. The issue was when he pooled the data from the api we could not see that coming out to the graph. We had a discussion on how implement the dashboard in the homepage as we were wanted something to look at when we log on. We currently have multiple pages with different charts and maps, we have the log in working as Ian finished it that has bootstrapped added to it. Another topic in this sprint was how we going to deploy the Application we have a few options in the original plan is Heroku but we have been working with Docker lately in college we wanted to look at the options. With a few options we might need to review some of the two technologies.

2.1.9 Sprint (March)

In this sprint, the final touches were added on the application. We added a Interactive map which is attached to the openWeather api. Google maps was easy to implement with the api and the python scripts we were using. Five day forecast was included into the application with all the Irish cities and towns added to a database. Garrett had issues on getting that data which was giving him errors when he requested a city. With this page it does a five day forecast where you can see the forecast for the following five days. We created a drop down menu where you can select a town and than the screen would change to the five day forecast. The original storyboard that was planned was a dashboard but Alan was having issues getting that together. But had plotly graphs up and running with four more tabs put together the whole app was nearly complete. The highcharts was put together with the openweather forecast, this was on the historic data page with bootstrap added to it to get a nice effect in it. The team currently have five tabs set up and the application running locally but we had discussions over the

possibility of getting hosted and that was the aim for our final sprint. We had assigned Ian the task and research of the Heroku or AWS was mentioned. The weather prediction and neural networks was up the models created but implementation to the app was made. This was worked on by Alan and was just trying to embed the notebook into the application. As this was one of the last sprints a lot of cleaning up the application and stylizing, the team all agreed on the overall design and happy with overall design.

2.1.10 Sprint (April)

This was the final Sprint before the deadline it was all about trying to host the web application on Heroku for hosting it, We also had testing and validation to work on it before hand up. We did run on multiple machines locally and ran it on different operating systems we were not happy overall look so we decided to change the template of the overall project. Garret worked on that but all functionality of app was running smoothly we yet had to run and see how fast it performed for testing and validation. We did use a tool called postman to run a http requests to if all requests worked. Alan was not fully finished on the final weather models but had enough to satisfy us to move forward. In the following chapter we look at all the different technologies used within the application. All these sprint notes will be put up in a wiki on our github.

Chapter 3

Technology Review

This chapter will discuss the technologies that were utilised in the project in detail. It will provide the reader with examples of the research that was carried out into the different types of available technologies. This chapter also cover some of the reasons why the team picked that technology over alternatives. Check out the nice graphs in Figure **??**, and the nice diagram in Figure **??**.

3.0.1 Web Application

In This section the different technologies used in the web app will be discussed and the reasons why these technologies were picked over alternatives. Because Python was used the project was built with many tools and applications.

3.0.2 Flask

Flask is a micro-framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. This means that flask provides a developer with tools, libraries and technologies that allows for the development of a web application. It aims to keep the core simple but extensible. The web application can consist of web pages, a blog, a wiki or as big as a commercial website. Flask is lightweight meaning there are little dependency to update and watch for security bugs. By default, Flask does not include a database abstraction layer, form validation or anything else where different libraries already exist that handle those components. Instead, Flask supports extensions that can add application features as if they were implemented in Flask itself. Numerous extensions provide database integration, form validation, upload handling, authentication and much more.

3.0.3 The Advantages of Flask

- Flask provides simplicity, flexibility and fine-grained control. It is opinionated (it lets you decide how you want to implement things).
- Flask focuses on the experience and learning opportunities, or if you want more control about which components to use (such as what databases you want to use and how you want to interact with them).
- Flask implements a bare-minimum and leaves the bells and whistles to add-ons or to the developer

3.0.4 Templates(Flask)

Using templates we are able to set a basic layout for web pages and mention which element will change. This way a header is defined to keep consistency over all pages of the website and if a change is needed in the header, we will only have to update it in one place. Using a template engine will save a lot of time when creating an application but also when updating and maintaining it.

3.0.5 JinJa(Flask)

Jinja Template Jinja2 is a full featured template engine for Python. It has full unicode support, an optional integrated sandboxed execution environment, widely used and BSD licensed.

What is Jinja Jinja2 is one of the most used template engines for Python. It is inspired by Django's templating system but extends it with an expressive language that gives template authors a more powerful set of tools. On top of that it adds sandboxed execution and optional automatic escaping for applications where security is important. It is internally based on Unicode and runs on a wide range of Python versions from 2.4 to current versions including Python 3.

Why The Team Used Jinja It has a Sandboxed execution mode. What this means for Jinja is that Every aspect of the template execution will be monitored and explicitly white or black listed, whichever is desired by the user's system. This makes it possible for Jinja2 to execute untrusted and unknown templates easily. jinja2 has powerful automatic HTML escaping system for XSS cross site scripting prevention systems. Jinja2 uses template inheritance which makes it possible to use the same or a similar layout for all templates which can be very useful for similar Web Pages to be made into a SPA (Single Page Application). Jinja2 is Great for High performance. It

has “just in time compilation” to “Python bytecode”. Jinja2 will translate your template sources on first load into Python bytecode for best runtime performance. Jinja makes it very Easy to debug your code and templates with a debug system that has integration for the template compile and runtime errors built into the standard Python traceback system. Configurable syntax. For instance you can reconfigure Jinja2 to better fit output formats such as LaTeX or JavaScript.

3.0.6 JinJa’s API

It uses a central object called the template “Environment”. The Instances of this class are used to store the configuration and global objects for your web Templates,they are then used to load templates from the file systems or from where ever the location of the templates may be stored on the system. Even if you are creating templates from strings by using the constructor of Template class, an environment is created automatically for you, albeit a shared one. Most applications will create one environment object on application initialization and use that to load templates. In some cases however, it’s useful to have multiple environments side by side, if different configurations are in use.

3.0.7 JinJa vs Django

Jinja

Jinja2 is a templating engine which was inspired by django template language. Though Jinja2 use django inspired syntax heavily, they are quite different in philosophy and internal technical aspects. Jinja2 is considered much faster in performance and can do heavy lifting in contrast to django. from django 1.8 it is now possible to use jinja2 templates in django projects easily. So if you need performance and more flexibility over django template engine and need all the other goodies of django then you could use jinja2 with django.

Django

Django is a MVC based web framework. It tells you how to organize and write your application. Jinja (Jinja2) is a template library. This means that you use Jinja to write the structure of your webpages. Once written, these templates are then served and rendered by Django.

Just to be clear, Django is not a web server either. It’s just a web framework. Django might use any of the famous web servers. It’s similarly also

not a Database server. It is just a web framework in python. It lays down certain conventions to write web apps to make life easier for developer.

3.0.8 Pymongo(Flask)

To connect MongoDB to Flask, We had to use a library called Flask-PyMongo which acts as a bridge between Flask and PyMongo. This extension allowed us to use Flask's normal mechanisms to configure and connect to MongoDB. PyMongo is a Python distribution containing tools for working with MongoDB, and is the recommended way to work with MongoDB from Python.

3.0.9 Python Language

The Statflow Application code is mostly written in Python programming language, The team decide to pick python as it uses pseudo-code like syntax and provides dynamic typing and has an interpreter unlike other languages like Java and C++. Another reason why Statflow choose Python over other languages, such as Java, because it provides a lot easier to use serial communication code. As a team the decision to pick Python was not difficult the language is easy to understand and very good for forgiving errors, Furthermore Python still be able to run or compile a program until you hit the problematic part. The team did discover after using python it tends to be slower to but that tends to be from a lot of referencing.

3.0.10 MongoDB

MongoDB is one of the most popular open source NoSQL database that uses a document-oriented data model. It stores data in JSON (JavaScript Object Notation) like documents. It is non-relational database with dynamic schema. MongoDB is horizontally scalable and easy for both the developers and administrators to manage. MongoDB is written in C++ and is currently being used by some big companies like Google, Facebook, Cisco, Ebay and SAP. Related information is stored together for fast query access through the MongoDB query language. Fields can vary from document to document; there is no need to declare the structure of documents to the system – documents are self-describing. If a new field needs to be added to a document, then the field can be created without affecting all other documents in the collection, without updating a central system catalog, and without taking the system offline. Optionally, schema validation can be used to enforce data governance controls over each collection. MongoDB document data model maps naturally to objects in application code, making it simple for developers

to learn and use. Documents give you the ability to represent hierarchical relationships to store arrays and other more complex structures easily. Because documents can bring together related data that would otherwise be modeled across separate parent-child tables in a relational schema, MongoDB atomic single-document operations already provide transaction semantics that meet the data integrity needs of the majority of applications. One or more fields may be written in a single operation, including updates to multiple subdocuments and elements of an array. MongoDB ensures a complete isolation as a document is updated; any errors cause the operation to roll back so that clients receive a consistent view of the document.[1]

3.0.11 MySQL

MySQL is a popular open-source relational database management system (RDBMS) that is developed, distributed and supported by Oracle Corporation. Like other relational systems, MySQL stores data in tables and uses structured query language (SQL) for database access. In MySQL, you predefine your database schema based on your requirements and set up rules to govern the relationships between fields in your tables. Any changes in schema necessitates a migration procedure that can take the database offline or significantly reduce application performance.

3.0.12 MySQL vs NoSQL

NoSql databases are non-rational meaning they are table-less. This leads to easier management and thus provide a higher level of flexibility with newer data models compared to SQL rational databases. Relational database or RDBMS databases are vertically Scalable When load increase on RDBMS database then we scale database by increasing server hardware power ,need to by expensive and bigger servers and NoSQL databases are designed to expand horizontally and in Horizontal scaling means that you scale by adding more machines into your pool of resources. Maintaining high-end RDBMS systems is expensive and need trained manpower for database management but NoSQL databases require less management. it support many Features like automatic repair, easier data distribution, and simpler data models make administration and tuning requirements lesser in NoSQL. NoSQL databases are cheap and open source. NoSql database implementation is easy and typically uses cheap servers to manage the exploding data and transaction while RDBMS databases are expensive and it uses big servers and storage systems. So the storing and processing data cost per gigabyte in the case of NoSQL can be many times lesser than the cost of RDBMS. In SQL server's

data has to fit into tables anyhow. If your data doesn't fit into tables, then you need to design your database structure that will be complex and again difficult to handle.[2] NoSQL database is schema less so Data can be inserted in a NoSQL database without any predefined schema. So the format or data model can be changed any time, without application disruption.and change management is a big headache in SQL. NoSQL database support caching in system memory so it increase data output performance and SQL database where this has to be done using separate infrastructure. NoSQL database is Open Source and Open Source at its greatest strength but at the same time its greatest weakness because there are not many defined standards for NoSQL databases, so no two NoSQL databases are equal No Stored Procedures in mongodb (NoSql database). GUI mode tools to access the database is not flexibly available in market Too difficult for finding nosql experts because it is latest technology and NoSQL developer are in learning mode.[3]

3.0.13 Rethink-Database

For the Database Technology as a Team We discussed many numerous types of databases during our meetings at the start of the project we discussed about the more relational Type of DataBases like SQL, MY-SQL, T-SQL, PRE-SQL and POST-SQL

Compared to the newer type Databases being introduced and used more often anymore by companies like RethinkDB, MongoDB, CouchDB etc etc where there not as rigid and inflexible Relational Databases for the simple reason that there a Nosql type of databases. which means that they are not like the highly structured table structure of a relational database thats is rigidly defined and a record type structure.

Nosql DB's are a collection of complex documents with nested data formats and have varying records format eg not all Data inputted to the DB have to have the exact same information . Another reason for using Nosql database is the way we utilised the API from the Open Weather Map website.

The data we were requesting from the website has a couple of formats to return the data (CSV and JSON) format. Since RethinkDB deals with JSON documents it seemed like the natural choice and the quick adaptability of creating tables and databases was very useful .

As well as dealing with JSON Objects as standard,The RethinkDB Database System Can also be easily deployed as a platform as a service (PaaS) which is alway a great selling point for any application that need or wants to be deployed for future use on a easily scalable level.[4]

3.0.14 Bootstrap

Bootstrap is a free and open source front end web framework for designing websites and web applications. It contains HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions Most of the elements on this application web pages are implemented using the bootstrap framework. One of the significant advantages of using bootstrap is high capability with various browsers. It is a frequent problem when some web page elements work for one browser and don't for another or work in a different manner. Bootstrap reduces this risk to a minimum. It provides templates that help increase productivity and help reduce the developing time, which is very important.[5]

3.0.15 Open Weather Map API

Why Choose Open Weather Map against other API Sources As a Team/Project group we again had another informal meeting to discuss our use of Api's in our project to help the system work fact and effective as well as have active data that was reliable and cost effective for the system ..we discussed numerous systems as follows : AccuWeather,Aeris Weather,OpenWeatherMap WeatherBug AccuWeather : AccuWeather is one of the leading digital weather information providers. According to its website, AccuWeather provides weather forecasts for nearly 3 million locations worldwide, and over a billion people worldwide rely on AccuWeather every day. AerisWeather : Founded in 1996, AerisWeather's mission is to "be the most dependable weather source in the country." The company provides detailed and comprehensive weather information, including local weather forecasts, extended forecasts and weather maps. WeatherBug : The WeatherBug brand is developed by Earth Networks, a company operating vast global sensor networks that monitor weather, lightning, greenhouse gases, cameras and more to deliver real-time localized weather information, weather alerts and critical atmospheric intelligence. WeatherBug is one of the company's most popular services, providing millions of people with real-time weather forecasts and weather-related information.

3.0.16 OpenWeatherMap

OpenWeatherMap provides free weather data for more than 200,000 cities accessible programmatically via the OpenWeatherMap API. The philosophy of OpenWeatherMap is based on OpenStreetMap and Wikipedia in that the

company would like to "make weather information free and available for everybody." OpenWeatherMap is an online Website/Api service that provides weather data, including current weather data,historical data and Forecasting its also offers Agricultural Weather services for farmers and Agricultural Contractors. The OpenWeatherMap API can be used by developers to build third-party applications that utilize OpenWeatherMap weather data and functions such as current conditions, 5 day 3 hour forecasts by city ID or Geo Coordinates and also 16-day forecasts as well as that there are also historical data available to download as a json file. You can also get recent weather station data and Create weather map layers. The API is free to use; however, there are also paid plans available for developers that require higher levels of weather data and support. Open Weather Map is also a very detailed API with quite detailed examples on every api call available in the Website .

3.0.17 JSON

JSON[(JavaScript Object Notation) is a lightweight data-interchange format. It uses human-readable text to send data objects consisting of name/-value pairs. JSON is defined by ECMA-404[39] standards which describes the allowed syntax. This project uses JSON extensively to send data from both the Web and APIS to the Main Server and responses are returned from the Main Server in JSON format. The Plotly Graphs also returns responses to the Main Server in JSON format.

3.0.18 JQuery

jQuery is a cross platform set of javascript libraries designed to make the client side scripting of HTML a much easier process. jQuery takes a lot of tasks that require many lines of javascript lines and puts them into methods that the developer can call with a single line of code. It Asynchronous JavaScript and XML (AJAX) calls and DOM manipulation. Following are the features of jQuery: HTML/DOM manipulation, css manipulation HTML methods, effects and animations, AJAX +utilities[6]

3.0.19 CSS

CSS is used to dene how the HTML elements are to be laid out on the screen and, in general, how the user will see it. It's highly benecial as the CSS document can manage the layout of not just one page or just specic item on the page but every single page in the project. For example, a website made

up of 5 dierent pages has a button on each page. The CSS can manage the design of every button to be the same no matter what page it's on. CSS can be internal or external. Internal CSS means that the CSS will be inside the HTML document and the downside of this is that it will only aect the page containing the code rather than all the pages. External CSS on the other hand is saved as a separate le in the solution with the .css extension and it may be used by all pages in the project.[6]

3.0.20 Html

HTML is a standard language used for the creation of web pages and web applications. It describes the structure/architecture and overall appearance of the information on the Internet. Web browsers receive documents from webserversandrenderthemintomultimediawebpages. Keywords and tags are used to format and demonstrate the content of Web Page. The layout of HTML follows a tree structure called the Document Object Model (DOM) and is validated on page load.[6]

3.0.21 Tensorflow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.[7]

3.0.22 Plotly

Statflow uses plotly for interacting with some of the data that's gathered from MET Eireann. Plotly.py is an interactive, browser-based graphing library for Python. Built on top of plotly.js, plotly.py is a high-level, declarative charting library. Plotly.js ships with over 30 chart types, including scientific charts, 3D graphs, statistical charts,SVG maps charts and more. Plotly Graphs can be viewed in Jupyter notebooks, stadalone HTML files, or hosted online by plot.ly Plotly creates leading open source tools for composing, editing, and sharing interactive data visualization via the Web. Our collaboration servers (available in cloud or on premises) allow data scientists to showcase their

work, make graphs without coding, and collaborate with business analysts, designers, executives, and clients.

3.0.23 Heroku

Heroku is a container-based cloud Platform as a Service (PaaS). For this project, Heroku was used to deploy and run both the Main Server and the Messaging Server using a free Heroku Hobby account. Heroku allows the deployment of applications from the version control system, GitHub by associating a specified GitHub repository when creating new applications. This means that when changes in the source code of the servers are sent to GitHub, they are automatically updated, redeployed and run in Heroku. Heroku builds the application the first time the repository is added and every subsequent time that a new version of the source code is added to GitHub. Heroku uses the package.json file of the project to download any necessary dependencies. It also compiles the source code of the project and generates a slug which is the dependencies, compiled source code and language runtime all bundled together ready for execution. A Procfile is a text file that is required to be present in the root directory of the application. It tells Heroku what language the application is written in and what file should be called to start the application running. The Procfile for the Main Server is as follows: web: node server.js Heroku also offers add-ons which are components or services maintained by a third-party provider or by Heroku itself. Add-ons are installed into the application by Heroku and are used to handle parts of the application such as data storage. This project uses two add-ons to the Main Server application, Cloudinary and GrapheneDB both of which are discussed in more detail in other sections of this chapter.

3.0.24 Highcharts

What is HighCharts

Highcharts is a charting library that has been written in pure JavaScript It offers a easy way of adding many interactive charts to your projects and web applications. HighCharts was Designed from Scratch with mobile Devices in mind eg for use with Companies to show detailed charts to potential customers for your Company. Highcharts has designed everything from there multi-touch zooming to touch-friendly tooltips as well as Dynamic Chart Loading which Works great on mobile platforms as Well as Web Applications.

Why The Team Choose HighCharts ? There were any Chart Visualization libraries available to use on the Market but none had a s much success for our project as Highcharts mainly because it is now becoming one of the most used and in demand visualization libraries out there and most of all it's free to most customers as well as that you can have more functionality available to you if need via a paid option as well .. below are a few of the main reasons the Team Decided on Highcharts.

- Easy learning curve
- Multiple chart types available eg(pie, bar, line,etc)
- HighCharts has Responsive charts available
- HighCharts can Handle everything you throw at it aka its (Robust)
- HighCharts makes it easy to parse documentation for use
- They have an easy to customize color scheme and palettes for your charts
- Has a Built in Export Chart to image file as well as other formats
- A highchart can be expressed as a json object, lets call it configobj object. To create any different kind of chart, what you have to do is take a configobj object from one of HighCharts demos and change it according to your object Data needs to visualize what you want.
- This configobj object is a constant among various charts available in HighCharts api. if you wanted to change your Bar chart to a Pie chart, all you'll have to do is: configobj.chart.type='pie';
- if you want to create a dashboard and want to create a lot of similar looking charts but with different data. Highcharts API has something called as plotoptions. You can set the default look-n-feel setting for a particular type (i.e. bar/pie etc.) of chart in its plotoptions. If you want to change this default look and feel just before creating the

3.0.25 Other Technologies

Here are some of the other technologies we are using on Statflow which helped to make the development process more productive and effective.

Here is a list of technologies:

- Visual studio code and sublime these are lightweight code editors to write most of the code.
- Git bash and Cmdr terminals for running process sh.exe and to commit, push, pull and fetch operations for the workflow.
- Operating systems such as Windows 10 and Linux(mint) were used in the creation of this site.
- Browsers such as Firefox and chrome were used for testing the application.
- Plotly cloud editor for creating and filling out graphs.
- The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Chapter 4

System Design

The architecture of this project is based around a three tier of technologies. In this chapter we go through all the tiers and explain each step and design of web application. The App was built with Flask and this technology is roughly built on the MVC but also it is in conjunction with a front end framework. But FLask is Back end as well as other python frameworks like Django. The middle tier consists of the RESTFUL api and Heroku and mostly weather Api which we get our data from. The back end has multiple databases like RethinkDB and MongoDB we are using MongoDB for members and RethinkDB for the data. All the user Authentication details of our user login is stored in MongoDB Above is the architecture of the whole project in a diagram as you can see there is a lot of API calls. In this chapter we will go step by step on overall design of the project.

4.0.1 Databases

We chose to use two databases to store our data, that being MLab MongoDB and RethinkDB. Both of these databases are being hosted online on Heroku. MLab was founded in 2011 and provides a fully managed cloud database service that can host MongoDB databases. Heroku offers a free plan for MLab MongoDB with up to 500MB free storage with paid plans thereafter. The purpose of MongoDB is to store the users data eg. UserName and Password. There are a few collections in the database, each to store a different set of data for the web application. RethinkDb we are using for storing data for the app to pull from for the graphs.

RethinkDB Query Language : ReQL
ReQL is the RethinkDB query language similar enough to SQL.ReQL offers a very powerful and easy way to manipulate JSON documents. ReQL

is different from other NoSQL query languages As It's built on three key principles: ReQL embeds into your programming language. Queries are constructed by making function calls in the programming language you already know. You don't have to concatenate strings or construct specialized JSON objects to query the database. All ReQL queries are chainable. You begin with a table and incrementally chain transformers to the end of the query using the . operator. All queries execute on the server. While queries are constructed on the client in a familiar programming language, they execute entirely on the database server once you call the run command and pass it an active database connection.

Here is how RethinkDB is implemnted into the app

Here's some Of the packages for RethinkDB for python:

```
import rethinkdb as r
from rethinkdb.errors import RqlRuntimeError, RqlDriverError

#rethink config
RDB_HOST =  'localhost'
RDB_PORT = 28015
STATFLOW_DB = 'statflow
\end{minted}

#rethink config
RDB_HOST = 'localhost'
RDB_PORT = 28015
STATFLOW_DB = 'statflow'

# db setup; only run once
def dbSetup():
    connection = r.connect(host=RDB_HOST, port=RDB_PORT)
    try:
        r.db_create(STATFLOW_DB).run(connection)
        r.db(STATFLOW_DB).table_create('rainfall1').run(connection)
        r.db(STATFLOW_DB).table_create('citylist').run(connection)
        print('Database setup completed')
    except RqlRuntimeError:
        print('Database already exists.')

    finally:
        connection.close()
```

```
dbSetup()

# open connection before each request
@app.before_request
def before_request():
    try:
        g.rdb_conn = r.connect(host=RDB_HOST, port=RDB_PORT, db=STATFLO
    except RqlDriverError:
        abort(503, "Database connection could be established.")

# close the connection after each request
@app.teardown_request
def teardown_request(exception):
    try:
        g.rdb_conn.close()
    except AttributeError:
        pass
```

Parallelism

All ReQL queries are automatically parallelized on the RethinkDB server as much as possible. Whenever possible, query execution is split across CPU cores, servers in the cluster, and even multiple datacenters. If you have large, complicated queries that require multiple stages of processing, RethinkDB will automatically break them up into stages, execute each stage in parallel, and combine data to return a complete result.

optimization

While RethinkDB doesn't currently have a fully-featured query optimizer, ReQL is designed with one in mind. For example, the server has enough information to reorder the chain for efficiency, or to use alternative implementation plans to improve performance. This feature will be introduced into future versions of RethinkDB. RethinkDB has powerful Hadoop-style map-reduce tools, that integrate cleanly into the query language.

Here is some of the code how we implemented MongoDB

```
@app.route('/login', methods=['GET','POST'])
def login():
    error = None # error variable set to None as default.
    if request.method == 'POST':
        users = mongo.db.users # access the users collection in the da
        # Checking to see if username exists
        login_user = users.find_one({'name' : request.form['username']
```

```
        if login_user: # if it does exist
            # check to see if password hash is equal to password hash
            if check_password_hash(login_user['password'], request.form
                # if password is correct then activate user session us
                session['username'] = request.form['username']
                return redirect(url_for('main')) # redirect to main ro
            # if password does not match then...
            else:
                # error takes in a string message
                error = 'Incorrect username/password'
                #return 'wrong password' # Login failed
        else:
            error = 'Incorrect username/password'
    # return login page and also pass in error message
    return render_template('login.html', error=error)
```

4.0.2 User Authentication

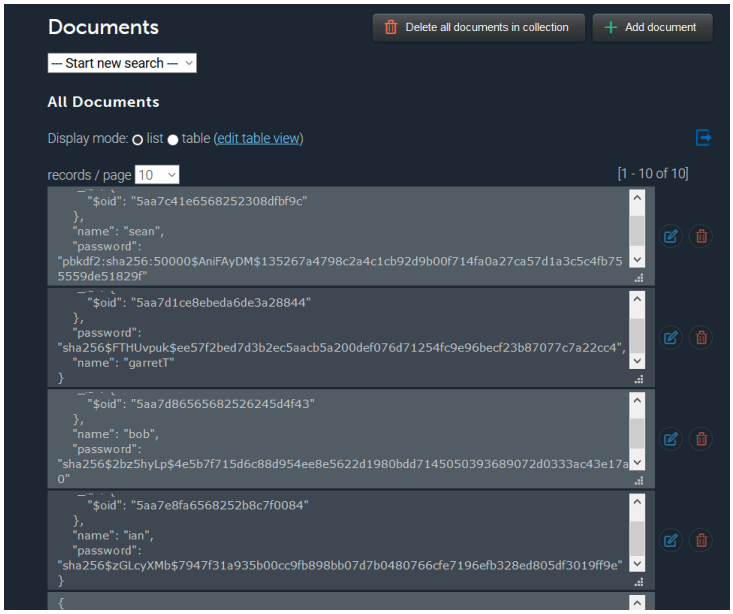
Instead of allowing full user access to all features of Statflow, we have restricted this by adding user authentication to the web app. To access all features, the user must register by creating a username and password. Once created, the user is then redirected to the login page to sign in. If the username or password does not match with the stored username and password in the Mongo database then the user is flashed an error message. Otherwise, when the user signs in successfully, a user session is activated to keep track of user status. When the user logs out, then the session is terminated and redirected back to the login page.

Register

```
users = mongo.db.users # Accessing our users collections
    # Checking to see if a username already exist in the users collect
    existing_user = users.find_one({'name' : request.form['username']}

       # if theres no existing username
       if existing_user is None:
        # Generating sha256 hash from user password
    hashpass = generate_password_hash(request.form['pass'],
    method='sha256', salt_length=8)
    # Add new user to users collection along with password
```

```
        users.insert({'name' : request.form['username'], 'password' : has
            # Once user is registered.. return login page for them to logi
                return redirect(url_for('login'))
            else:
                error = 'Username already exists'
```



login

```
        if login_user: # if it does exist
            # check to see if password hash is equal to password hash
            if check_password_hash(login_user['password'], request.form
                # if password is correct then activate user session us
                session['username'] = request.form['username']
                return redirect(url_for('main')) # redirect to main ro
            # if password does not match then...
            else:
                # error takes in a string message
                error = 'Incorrect username/password'
                #return 'wrong password' # Login failed
        else:
            error = 'Incorrect username/password'
```

4.0.3 Bootstrap

Bootstrap is an open source front-end HTML and CSS framework for designing websites and web applications. It contains HTML and CSS based

design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Bootstrap allows you to build templates quickly which saves lots of time on design aspect of a web application. Bootstrap responsive features make your web pages to appear more appropriately on different devices and screen resolutions without any change in markup. It is very easy to use for anyone with basic working knowledge of HTML and CSS. Bootstrap is compatible with modern web browsers such as Mozilla FireFox, Google Chrome, Safari, Internet Explorer and Opera.

In this project, we have implemented Bootstrap by using a CDN (Content Delivery Network) link instead of downloading Bootstrap packages. A CDN is a geographically distributed network of proxy servers and their data centers which distribute a service such as Bootstrap that can load CSS, JavaScript and images remotely from its servers. For example, to access CSS only:

Listing 4.1: Python example

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap
```

4.0.4 Flask application

Flask Framework was an easy choice for the team to build an application with as we all worked with the framewrk it was the right choice. Flask is a python web framework based on Werkzeug, Jinja2. Flask is easy to understand and also a prerequisite for Django. In this we will go through the overall layout of the app creating an app is very easy just adding to that and the overall design is where it can get tough.

4.0.5 Http Requests

HTTP knows different methods for accessing URLs. By default, a route only answers to GET requests, but that can be changed by providing the methods argument to the route() decorator. Here are some examples:

```
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        do_the_login()
    else:
        show_the_login_form()
```

If GET is present, HEAD will be added automatically for you. You don't have to deal with that. It will also make sure that HEAD requests are handled as the HTTP RFC (the document describing the HTTP protocol) demands, so you can completely ignore that part of the HTTP specification. Likewise, as of Flask 0.6, OPTIONS is implemented for you automatically as well.

You have no idea what an HTTP method is? Worry not, here is a quick introduction to HTTP methods and why they matter:

The HTTP method (also often called "the verb") tells the server what the client wants to do with the requested page. The following methods are very common:

GET

The browser tells the server to just get the information stored on that page and send it. This is probably the most common method.

HEAD

The browser tells the server to get the information, but it is only interested in the headers, not the content of the page. An application is supposed to handle that as if a GET request was received but to not deliver the actual content. In Flask you don't have to deal with that at all, the underlying Werkzeug library handles that for you.

POST

The browser tells the server that it wants to post some new information to that URL and that the server must ensure the data is stored and only stored once. This is how HTML forms usually transmit data to the server.

PUT

Similar to POST but the server might trigger the store procedure multiple times by overwriting the old values more than once. Now you might be asking why this is useful, but there are some good reasons to do it this way. Consider that the connection is lost during transmission: in this situation a system between the browser and the server might receive the request safely a second time without breaking things. With POST that would not be possible because it must only be triggered once.

DELETE

Remove the information at the given location.

OPTIONS

Provides a quick way for a client to figure out which methods are supported by this URL. Starting with Flask 0.6, this is implemented for you automatically.

Now the interesting part is that in HTML4 and XHTML1, the only methods a form can submit to the server are GET and POST. But with JavaScript and future HTML standards you can use the other methods as well. Furthermore HTTP has become quite popular lately and browsers are no longer

the only clients that are using HTTP. For instance, many revision control systems use it. [8]

4.0.6 Main Menu and design

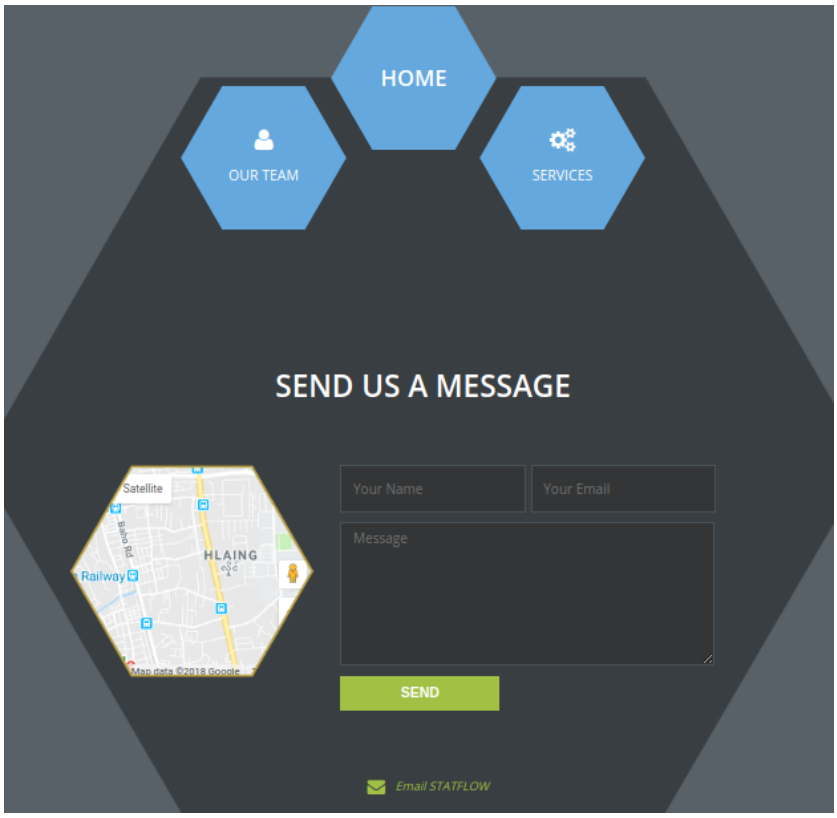
For the Design of The Staflow Hub website we looked at various templates on various websites offering templates we decided to use a Template Called Templatemo - Polystar 408

We then acid stripped this template down and worked our functioning code and flask jinja templates in with it as well as our own stylesheets and various external style sheets available with the template.

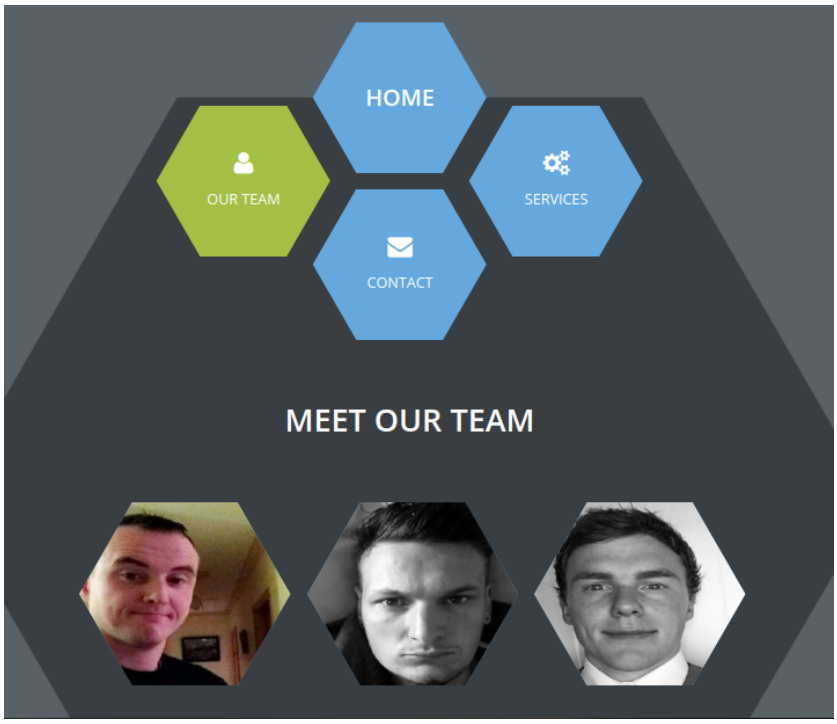
Polystar is free HTML5 template,It is styled with hexagon shapes and fixed width layout. Homepage as well as that it features a lightbox gallery.

The homepage - includes a 4 polygon Diamond style layout

Contact Page - has the ever-Present 4 diamond style polygon layout at the top of the page with a contact form to submit a query, a map image of the company's location as well as an email icon below for email's



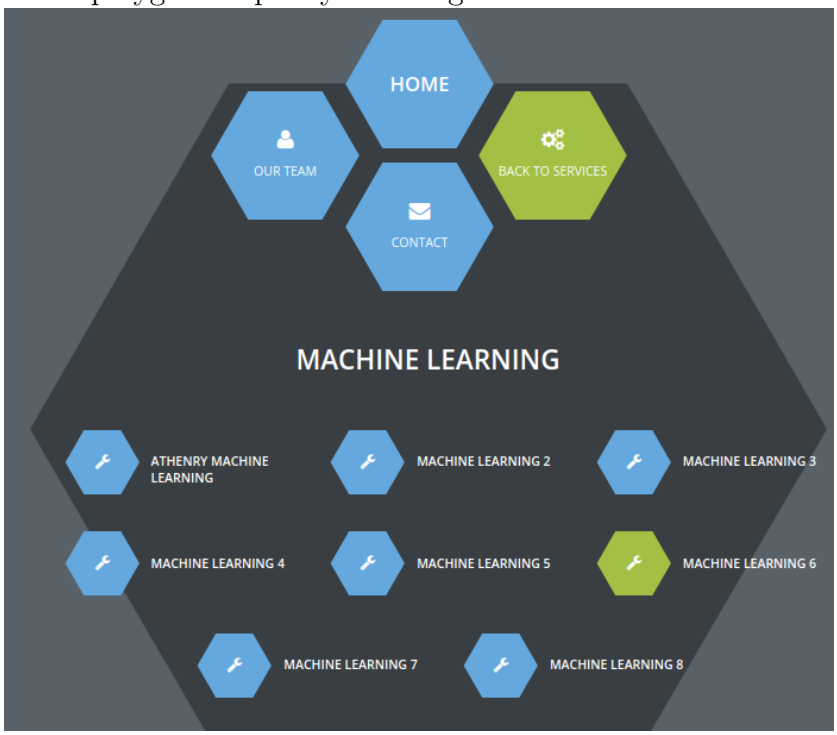
Our Team - includes a 4 polygon Diamond style layout below has a 3 in-line polygon layout with pictures of the team members and a brief description of each member



Services - shows another set of polygon inline below of 3 in-line followed by 2 in-line below and each polygon has a function service we provide

Machine Learning submenu - The Machine Learning SubMenu displays a

submenu of all the machine learning data/functions we are currently working with in a polygon shape layed out again in-line



4.0.7 Client and server

As we see above with the REST api and the Flask Framwork its pretty much all implemented when you import the Flask server. So overall we created an API.

4.0.8 Tensorflow

Deep learning is section of machine learning that is a set of algorithms that is inspired by the structure and function like a brain. TensorFlow is machine learning framework from Google created and used to design, build, and train deep learning models.You can use the TensorFlow library do to numerical computations, which in itself doesn't seem all too special, but these computations are done with data flow graphs. In these graphs, nodes represent mathematical operations, while the edges act like the data, which usually are multidimensional data arrays or tensors, that are communicated between these edges.[9]

The name "TensorFlow" is derived from the operations which neural networks perform on multidimensional data arrays or tensors. It's a flow of tensors. For this application we used tensorflow to build models using data sets from various weather stations in Ireland. We will will be using various models of regression models and statistical models.[10]

There is a massive amount of work behind the training of models but we will briefly talk through how we implemented into our project. First we had to import the dataset than create arguments like labels which is an array. A batch size which is the size of data and features which is the raw data. Import the csv and build the set usually skip the header all this does it reads in file but skipping header so a full csv is read in. Like above you would put python script to add features you want and arguments. Datasets have many methods for manipulating the data while it is being piped to a model. The most heavily-used method is map, which applies a transformation to each element of the Dataset. So this map makes a new function that describes each item in the dataset. So the map function changes or manipulates the dataset. The set contains labels and features pairs instaed of simple scaler strings. The tf.data module provides a collection of classes and functions for easily reading data from a variety of sources. Furthermore, tf.data has simple powerful methods for applying a wide variety of standard and custom transformations.[11]

Linear regression models

Linear regression is the simplest form of regression. We model our system with a linear combination of features to produce one output. That is,

$$yi = h(xi, w) = wTxi \tag{4.1}$$

Our task is then to find the weights the provide the best fit to our training

data. One way to measure our fit is to calculate the leasts squares error (or loss) over our dataset:

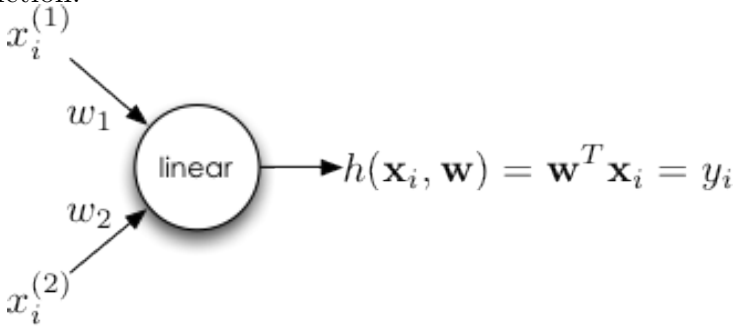
$$yi = h(xi, w) = wTxi \tag{4.2}$$

In order to find the line of best fit, we must minimize L(w). This has a closed-form solution for ordinary least squares, but in general we can minimize loss using gradient descent.

$$yi = h(xi, w) = wTxi \tag{4.3}$$

Training a neural network to perform linear regression

So what does this have to do with neural networks? In fact, the simplest neural network performs least squares regression. Consider the following single-layer neural network, with a single node that uses a linear activation function:



[12]

Here is some of the packages i was using in the table below:

Library	Description of Usage	Source
datetime	Used to increment our requests by day	Standard Library
time	Used to delay requests to stay under 10 per minute	Standard Library
collections	Use namedtuples for structured collection of data	Standard Library
pandas	Used to process, organize and clean the data	Third Party Library
requests	Used to make networked requests to the API	Third Party Library
matplotlib	Used for graphical analysis	Third Party Library

[13]

Here is a link to the notebook where you can see the packages working: For further references see notebook or go to the next url: `https://github.com/gtonra89/Final-Year-Project-4th-Year`

Chapter 5

System Evaluation

This chapter will evaluate the software developed in the project and will describe the testing that was carried out in order to evaluate all of the system components. It will also highlight the limitations of the software and analyze where there are opportunities to improve upon the way that the system has been developed.

5.0.1 Unit testing

Unit test is a piece of code that tests a unit of work, logical unit in the tested system. Your unit tests should be: automated independent consistent and repeatable maintainable.

We set up a Flask testing environment to test as we went along The flask documentation was very handy in terms of the you set up an virtual Environment.

```
from Flask import Flask

app = Flask(__name__)

class MyApplication():
    def __init__(self, param1, param2):
        app.add_url("/path/<methodParam>", "method1", self.method1, me
        # Initialize the app

    def getApplication(self):
        options = # application configuration options
        middleware = ApplicationMiddleware(app, options)
        return middleware
```

```
def method1(self, methodParam):
    # Does useful stuff that should be tested
# More methods, etc.
```

We did test the http calls with postman to see if all the post and get requests work

Postman is a powerful GUI platform that makes API development faster and easier. It is a complete tool-chain for API Development. Postman is a Google Chrome app for interacting with HTTP APIs. It presents you with a friendly GUI for constructing requests and reading responses. The people behind Postman also offer an add-on package called Jetpacks, which includes some automation tools and, most crucially, a Javascript testing library. This post will walk you through an example that uses those testing features. While they won't replace your focused unit tests, they do breathe new life into testing features from outside your applications. This makes it extremely valuable for functional testers or for developers who love to test outside-in. Postman[14]

5.0.2 Issues

With software development you do run into issues with all the technologies trying to merge but because the way we managed our app we didn't have many issues. We found coming back to the issue every once in a while helped us get it fixed. Good management and helping each other out in research of this problems we had no real issue stack overflow was great help and Google. We sometimes had issues where ask our supervisor and he would guide us in the correct area to find the solution. Git Issues was a major help we had some issues where some developers gave us a hint and we solved the issue.The main problem we could have encounterd was management of github but we were well organized .

Flask latency issue When we a couple of web pages that retrieve weather data from a database and representing the data into graphs, we faced a problem with flask where rendering a web page became slow. When you try to redirect to another page for instance, the request took a few extra seconds. To fix and debug this problem, I did some research on the matter and found that the issue was common on Stack Overflow. The solution was to introduce multithreading to the server as suggested on Stack Overflow.

Chapter 6

Conclusion

Throughout the project's development cycle, there have been many ups and downs and a lot of learning outcomes achieved. These ranged from issues with time management to learning about new software, The project is not fully finished but is in a good position to where we are happy with our accomplishments. Yes there was frustrating days and lots of disagreements but without these learning curbs learning would be non existing, With finishing the project to a place where we are happy and within the time giving we had reached the goal in terms of the overall project and a executable project. Sprints were carried out monthly as you can read on chapter 3, this helped greatly as we were not fully what direction we were going in the first few weeks of development. The team had an overall design in mind but no real subject to cover we knew what the outcome was going to be but didn't have a final idea in the type of data we were covering. With these Sprints in place we created milestones so we could tell where we were at every month, this helped greatly in the development of the project.

We had the milestones in place to help guide us and we had an application running before the December holidays but a lot of components were no where near ready. With the aid of Google docs we found it very useful in terms we knew exactly where we were in the project. Github was very useful in terms of production but also greatly useful for project tasks and some milestones in terms of the production. Yes we didn't get to use the branching properly as we had issues at first and lost work over it.

During the project there was various forms of methodologies used for this project mostly Agile but we did have scrum meetings most mornings near the end of project. Planning large parts of the project so that meeting like i said above these meeting consisted of handing workloads to individuals. RAD was used at times to build smaller applications or prototypes especially for testing components. These components would make their way to final project

and pushed up to github.

As we can see in the technologies chapter we had a lot to learn but found working python we found using third party libraries and dependencies we found python was so flexible and easy to configure. We mostly wrote our code in Microsoft visual code we found that the easiest to work and very nice for integration. The command line was mostly used to configure the databases and to run the app also to monitor our git repository. As this was a web application other languages like HTML,CSS, JavaScript and Bootstrap were used for different functionality like customization or for the workbench. Other frameworks like Mongodb and Rethink Db were so different it did cause us issues as you can read about in the last chapter.

The architecture of project was easy to run there was a few ways of doing the overall layout of the files within the flask app but we decided to keep it mostly the same as we did run into issues earlier. The machine learning parts of the project was a slow learning experience as we only covered a small bit in a semester so a lot of research went into that but we are satisfied with our findings and results.

- **Learning Outcomes** Here are some of the outcomes we learned on the project we have gone through most of the technologies ; Teamwork – Being able to work as part in a team is probably the most important of working in a group project. The software development team that were chosen for this project,we work extremely well together, there are times where we disagree but that is real in the work industry we might not like the people we work with or even the way they work but we were lucky we were allowed to choose to work for.

- **Communication** – This is a large part of anyone's daily life, especially where a team is working together to try and achieve a common goal. It became evident that as the project progressed that the better the communication, the better the result. With all the time we spend together it was still important to set up sprints and have a sit down and negotiate how we will progress. This module made it clear that communication is vital in software development as work can be accomplished quicker and deadlines be hit. We found when we had meetings and online conversations tasks were done.

- **new Technologies** – Being immersed in new technologies is something that is both challenging and rewarding. In one hand, there is the learning curve associated with learning something new, but equally the elation experienced when something finally clicks into place. Most of the technologies used in this project are modern and cutting edge,

displaying modern architectures and even updates or modifications of older software can be tough to keep up. Through a lot of research into new methods and technologies we found it that we were up to date in most of the technologies we used in this project and was new and innovated.

- **Analytical Approach** – From the outset, an analytical approach was encouraged and nurtured in order to reach goals and discuss problems that we encountered. We stood back a few times in the project which was great as we could have ran into bigger issues. If any major tasks had to done or big changes made we would analyze it and discuss it, we found this helpful as even the smaller changes can help massively later on in the project.

- **Mapping Project** – We found planning out project was best and is good practise, but to take our time in the approach and not make any rash decisions that would effect the work flow of the project was vital.

- **Documentation** – This is one of the most important skills to have when doing a major project we found commenting your code as well updating the milestones was essential in the project, and very important when reviewing the work that has been done. Most of the team took notes in what they worked on every week. Most of the project is documented we found very important especially if it was going to affect another team member. We found if a member was not working on the project for over a week he can see the changes and be updated in what has been coded. In industry level this is vital especially if there is more than one team working on a project.

- **Problem Solving** – This is software always learning and figuring out how to work something. When a project with a lot of moving parts we found that learning a tool or a new software was vital. Software is always evolving so your forever learning new skills or adapting to new changes. We found we were learning for a massive part of this project as together many different platforms we needed to do research on all and see how they were compatible.

Final thoughts As we hand up the project and we have reached most of the goals we set out we have manged greatly as a team and are satisfied with production.

The Whole development of this final year project from the very start in September all the way now in April was and seemed very challenging

but very rewarding experience. We are very proud of ourselves in such a big scale project to learn and master some of the new technology we came across. As well as learning new technology we learned how to work as members in a team and communicate accordingly. We discovered weekly or monthly sprints really helped out our research and development in this application. Every member had their strengths and weaknesses but worked them out and the help of each other when times were tough and stress-full during exam times and all. But the result was a well built app that is nicely designed well structured and easy on the eye but most of all working to our satisfaction.

Appendix A

Github Link

https://github.com/gtonra89/Final-Year-Project-4th-Year

Bibliography

[1] Vijay, "Relational database management system (rdbms) vs nosql./,"

[2] Wiki, "Nosql/,"

[3] A. Gajani, "The key differences between sql and nosql dbs./,"

[4] http://rethink.com/, "https://www.rethinkdb.com/docs//,"

[5] http://Bootstrap.com/, "https://getbootstrap.com//,"

[6] http://w3.com/, "https://www.w3schools.com/html//,"

[7] Tensorflow, "www.tensorflow.org,"

[8] F. HTTP, "www.flask.pocoo.org,"

[9] T. weather, "machine-learning-to-predict-the-weather/,"

[10] regression models, "machine-learning and regression models/,"

[11] regression models, "machine-learning and regression models/,"

[12] http://briandolhansky.com/, "Relational database management system (rdbms) vs nosql./,"

[13] A. Calls, "machine-learning-to-predict-the-weather/,"

[14] http://postman.com/, "Postman api testing/,"