SWEN30006 Project 3

Putting It All Together

The Task

This peroject combines everything you have learnt so far in the subject, your design skills, your work on data parsing and regression and your understanding of rails to build a service that predicts hyperlocal weather patterns and provides this information to users through both an API and Webview.

This project requires two submissions, the first (Part A) is a design document that details your intended overall system architecture and low level and the second is the actual implementation of the system.

Unlike the first two projects, this project is designed to be completed in groups of 3-4. You are required to submit your team to us by the 19th of April. You can do this by emailing Mat at mathew.blair@unimelb.edu.au, with the subject SWEN30006-2015-Project 3 Team, a list of your team members names and student numbers. Please ensure only **one** person from each team submits this.

Project Overview

The aim of this project is to use your skills in downloading, parsing and regressing data to look for correlations between different data sets use this information to provide a service that will perform hyperlocal weather predictions.

By hyperlocal, we mean predictions for your neighbourhood, rather than the usual predicitons per city. Your service must be a **Rails Application** written in **Ruby** that we will host and test with our own client in week 12.

Predictions

You will be required to make data predictions on the following weather features:

- Rainfall
- Wind Direction
- Wind Speed
- Temperature

Your service must provide probabilities and time frames with each predicition (i.e. it may predict a 90% chance of 5mm of rain in the next 20 minutes at my location).

You may use any method you like to predict the data, as long as you are making the calculations yourself and not using a third party's predictions. You may use any third party *gems* to enable statistical calculations. We reccommend you look into using Statsample as it provides a number of tools for analysing correlations and multi-variate regressions that may improve your predictions.

API Specification

Your application must provide an API that implements the following calls. You must take care to implement the calls exactly as specified as we will be testing them with our own client that will not handle variations. You may, however, extend these services and add extras as you see fit.

You should use this API specification to guide your extraction of functional requirements. The API Specification is written in the same format as a rails route file.

- `GET /weather/locations`
    - Returns a JSON object containing all valid stored locations with the following format

```
{
   "date":"21-04-2015",
   "locations":[
      {
         "id":"MELB_TULL",
         "lat":"39.129301",
         "lon":"23.1231",
         "last_update":"13:12pm 21-03-2015"
      },
      .... remaining locations
   ]
}
```

- `GET /weather/data/:location_id/:date`
    - Where location is the location_id is the unique identifier to a weather station and date is a

string of the format "DD-MM-YYYY". This should respond with the following format containing all entries for that location on that day. 'current_temp' should return null if no measurements are available from the past 30 minutes, 'current_cond' should return a description of the current conditions in one word (i.e. sunny, raining, cloudy, snowing etc.)

```
{
   "date":"21-04-2015",
   "current_temp":"21.291",
   "current_cond":"sunny",
   "measurements":[
      {
         "time":"12:30:23 pm",
         "temp":"24.5",
         "precip":"0.4mm",
         "wind_direction":"NNW",
         "wind_speed":"12.3"
      },
      .... remaining measurements
   ]
}
```

- `GET /weather/data/:post_code/:date`
    - Where postcode is a unique Victorian postcode [3000-3999] and date is a string of the format "DD-MM-YYYY". This should respond with the following format containing the measurements for all weather stations in that postcode region on that day.

```
{
   "date":"21-04-2015",
   "locations":[
      {
         "id":"MELB_TULL",
         "lat":"39.129301",
         "lon":"23.1231",
         "last_update":"13:12pm 21-03-2015",
         "measurements":[
```

```
            {
               "time":"12:30:23 pm",
               "temp":"24.5",
               "precip":"0.4mm",
               "wind_direction":"NNW",
               "wind_speed":"12.3"
            },
            .... remaining measurements
         ]
      },
      .... remaining locations
   ]
}
```

- `GET /weather/predicition/:post_code/:period`
    - Where postcode is a unique Victorian postcode [3000-3999] and period is one of {10,30,60,120,180} and is a measure of time in minutes from the current time. This should respond with the hyperlocal weather prediction aggregated for that postcode, every 10 minutes, for the next 'period' minutes, starting with the current conditions ('i.e. "0" minutes from now').

```
{
   "location_id":"MELB_TULL",
   "predictions":{
      "0":{
        "time":"13:12pm 21-03-2015",
        "rain":{
             "value":"0mm",
             "probability":"1",
          },
        "temp":{
             "value":"24.8",
             "probability":"1",
          },
        .... remaining measurements
```

```
      },
      "10":{
        "time":"13:22pm 21-03-2015",
        "rain":{
             "value":"5mm",
             "probability":"0.92",
            },
        "temp":{
             "value":"23.2",
             "probability":"0.96",
            },
            .... remaining measurements
      },
   }
}
```

- `GET /weather/predicition/:lat/:long/:period`
    - Where lat and long are strings representing lattitude and longitude,period is one of {10,30,60,120,180} and is a measure of time in minutes from the current time. This should respond with the hyperlocal weather prediction for that location, every 10 minutes, for the next 'period' minutes, starting with the current conditions ('i.e. "0" minutes from now').

```
{
   "lattitude":"39.12391",
   "longitude":"38.4920",
   "predictions":{
      "0":{
        "time":"13:12pm 21-03-2015",
        "rain":{
             "value":"0mm",
             "probability":"1",
          },
        "temp":{
             "value":"24.8",
```

```
            "probability":"1",
          },
        .... remaining measurements
      },
      "10":{
        "time":"13:22pm 21-03-2015",
        "rain":{
             "value":"5mm",
             "probability":"0.92",
            },
        "temp":{
             "value":"23.2",
             "probability":"0.96",
            },
            .... remaining measurements
      },
   }
}
```

Web Interface

Your service must provide a web interface that allows you to navigate the same level of functionality as the API provides. This does not need to look pretty, and can use the same calls as the API.

It should simply return HTML instead of JSON. This html should be well structured with appropriate classes and ids to mirror the JSON structure.

Note

> *Both the Web interface and API do not need to worry about setting data, or about authentication. You can assume each transaction is stateless.*

Data Sources

We are not restricting where you source your raw meteorological data from for this project. The only restriction we are placing is that you must not be downloading and using any data that already includes prediction for

hyperlocal rainfail, insolation or temperature.

You may download past data for these values (for example, Forecast.io is an excellent source of this information) but **only** for validation of past information. If you are found to be downloading your predictions from other sources and presenting them as your own, you will fail.

Software Design - Submission 1

For your initial software design you will be required to submit 4 design documents, collated into **one** pdf. They are as follows:

- A Component Diagram
    - Your component diagram should outline your higher level architecture, including major components (Models, Views, Controllers) as well as sub-components within these.
    - Your component diagram should also show all interfaces between the components and the methods available on them
- Sequence Diagrams
    - You are required to produce 4 sequence diagrams, one for each of the following:
        - Background data retrieval
        - API Call to retrieve all locations
        - API Call to retrieve data for a location id
        - API Call to retrieve the predictions for a period for a given lattitude and longitude
    - Thesemust be low-level sequence diagrams, showing the low level interactions between classes.
- Detailed Class Diagrams of **all** models and controllers.

In your design you should apply the design theory that you have learn so far in class, whilst also following the rails convention of thicker models and thinner controllers. That is, most of the 'business' logic should be in the models and the controllers should act as facilitators.

All documents **must** be provided in a single pdf document. We will not accept Microsoft Visio or Enterprise archiect documents.

Implementation - Submission 2

For your implementation you will be building a Rails Application based on the design your team has produced

for submission 1. This Rails application should be entirely self contained. You may choose any database format you choose, but if you are using anything other than 'sqlite' ensure the database is in your gem file.

You are **strongly** recommended to use Git for version control (through either github or bitbucket). It makes it much easier to work in a team environment on a Rails project. If you do use so, please provide us with access to the repository by adding Mat (on github: matblair, on bitbucket: mblair).

Your code will be marked on quality and maintability, therefore you should endeavor to make follow good object oriented practices and guidelines whereever possible. Gems like 'rubocop' may help with this process.

All projects will be assesed during a half hour marking session in Week 12 where you can demonstrate your application's functions and design whilst allowing us to ask questions about certain functions.

Along with your Rails application, you also required to submit, as a group the following:

- A Reflection on Your Design and Implementation, including:
    - Why changes, if any, were made to your original design, along with updated design documentation
    - Aspects you found challenging
    - Your thoughts on the value of spending time and effort developing a thorough design prior to implementaiton.
- A README on how to use your Rails Application, if it differs from the standard procedure we will run (outlined in submission requirements below).

As an individual we are also asking you to submit a short (200 words) explanation outlining the contributions of your team members and yourself to the final project along with any troubles that were faced in development that you believe we may need to hear about (this is not required if you have used Git as we can see the committ history).

Note

> *With any group project there tends to be a few groups that run into issues that, for some reason or another, impact upon their ability to deliver the project. If any issue has arisen in your group we want to know about it sooner, rather than later, so that there is appropriate time to fix the situation.*

> *If you do not tell us when the issue arises, but instead wait until your project demonstration, there is a lot*

> *less flexibility available in how we can handle it.*

Submission Instructions

Project submission will be enabled through the LMS. We will require you upload a zip file of your rails application and pdf of your design documentation to the submission links we provide.

Software Design Submission

You should submit a single PDF file to the provided *Turnitin* link by the due date. Only one member from each group needs to submit the file.

Implementation Submission

You must submit your rails application, along with the reflection document, in a zip file to the provided LMS submission page. Only one member from each group needs to submit the file.

The rails app you provide will be run as follows:

```
bundle install
rake db:create
rake db:migrate
rails s
```

Should your application require additional commands in order to set up, you **must** include a README file in your project detailing the additional commands.

If your program does not run using these commands (or those provided in your README), you will lose **1 mark** and we will attempt to fix it. If we cannot make your rails app run at all, you will receive 0 marks for JSON, HTML and Web View criteria.

Marks

This project will account for 16 marks out of the total 100 available for this subject. 10 of these marks will come from the design submission, and 6 from the implementation.

Design Document Criteria

Criterion	Percentage
Component Diagram	4%
Sequence Diagrams	3%
Detailed Class Diagrams for all Controllers and Models	2%
Fitness for Requirements	1%

All UML submissions will be marked on their correctness, how well the problem has been deconstructed and cohesion with other diagrams.

The final 1% mark relates to how well your design captures and implements the project requirements. That is, does it look like your design will fulfill all requirements if it is implemented.

Implementation Criteria

Criterion	Percentage
Functional Correctness	3%
Reflection on Design Adherance	2%
Code Quality and Maintainabilty	1%

Functional correctness will be marked on your project's adherance to the API specification, the use of the web view and whether or not the project returns reasonable predictions (i.e. not predicting a blizzard when it is 30 degrees celcius) and accurate past data.

Your reflection must comment on how well your implementation adheres to your original design implementation. If you have changed your design in your final implementation, you must include a new class diagram and explanatory reasoning as to why you made those changes to receive any marks.

Code Quality will be marked on following good object oriented principles, with no unneccesary code duplication and good functional decomposition.

On Plagiarism

> *We take plagiarism very seriously in this subject. You are not permitted to submit the work of others under your own name. More information can be found here.*

Submission Dates

Submission 1 is due at **11:59 p.m. on the 10th of May.** Submission 2 is due at **11:59 p.m. on the 29th of May.**

Any late submissions *will incur a 1 mark per day penalty* unless you have supporting documents. If you have any issues with submission, please email Mat at mathew.blair@unimelb.edu.au, before the submission date.