Scrum 1: Planning Meeting – 10th March 2017
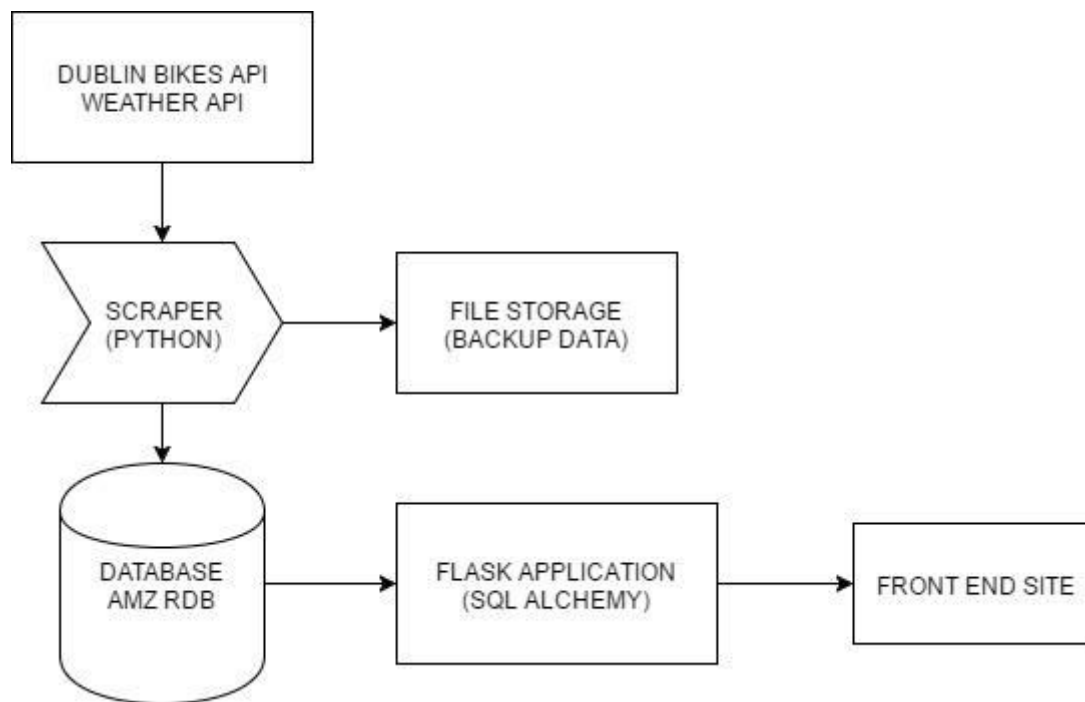
Attended by:

Customer: Devin Stacey
Team: Emma Byrne, Katherine Campbell, Pamela Kelly

Plan for Scrum 1:

Visual Overview:

Tasks for Sprint 1:
- Familiarise ourselves with the Dublin Bikes API, and as a secondary priority the Weather API.
- Formalise a structure for how the data scraped from the API will be stored in the database.
- Familiarise ourselves with the other technologies pertinent to the project: Amazon RDB, Amazon Hosting, Flask, SQL Alchemy.
- Sign up for a Trello account and begin organising the project tasks through that. Look at how to convert to burndown charts.
- Plan standups (may vary over the break in terms of what days suit – the team will decide on regular days prior to when classes recommence).
- Set up scraper to scrape information from the APIs. (Priority: High 1)
- Set up database so that information can be stored immediately once scraper is ready. (Priority High 1)
- Create basic html/css front end site. (Priority: Medium 1)
- Set up Flask App to process queries to the database from the front end site. (Priority: Medium 2)

Standup Meeting: 14th March 2017:

After our initial research, we confirmed that the highest priority tasks are the scraper, the database and the front end site. We divided these tasks up as reflected below. We decided that for the database we are going to use Amazon RDS and MySQL, because we are all already familiar with MySQL. We have moved the Flask application task to the backlog for now. Choosing to focus on the three above listed tasks as high priority for this Sprint.

Katherine:
- Looked at project organisation tools, i.e. Trello.
- Started researching Flask, looked at Flask tutorials.
- Looking at ways to develop the database from a user point of view.
- Next step: Work on database setup with Amazon RDS and MySQL.
- Not Blocked.

Pamela:
- Looked at Dublin bikes API.
- Defined a possible structure for the database based on data types given by API.
- Started a team Trello board and organised structure.
- Next step: Work on setting up the scraper in python.
- Not Blocked.

Emma:
- Looked at Dublin bikes API.
- Looked at scraping the API and using javascript to access information.
- Researched Amazon S3 and RDS.
- Next step: Work on the front end set up in HTML and CSS.
- Not Blocked.

Standup Meeting: 17th March 2017:

Pamela:
- Started work on the scraper. Signed up for a Dublin Bikes API key.
- Read through documentation and lecture notes on API requests.
- Wrote a basic script for sending requests.
- Next step: continue work on the scraper. Store information pulled from the API in files. For now store them in files until the database is up and running.
- Not Blocked.

Emma:
- Started working on the front end HTML and CSS. Signed up for a Google API key for use of maps and markers.
- Wrote a basic HTML webpage which includes the Google map and 1 marker set at a specific location.
- Next Step: Set up markers for all the locations of the bike stations from some of the data that we have scraped already. Redesign colour scheme of webpage.
- Note Blocked.

Katherine:
- Has thoroughly read the documentation for creating a MySQL instance and connecting to a database on a MySQL DB instance via Amazon Web Services.
- Read through Amazon VPC documentation.
- Next Step: Set up the database.
- Not Blocked.

Standup Meeting: 21st March 2017:

Pamela:
- The scraper script is set up to send queries every 5 minutes and store the results in files.
- Got sidetracked by some git and eclipse errors which needed to be resolved before moving forward.
- Next Step: Get the scraper running on EC2 continuously.
- Blocked: Have to wait for the database to be set up so that when the scraper is running on the EC2 it has somewhere to send the data to, other than storing it in the backup files.

Emma:
- Continued working on the basic HTML and CSS.
- The map and first marker marker are set up.
- Next Step: Set up markers for the rest of the station locations.
- Blocked: Waiting for data from the API in order to set up the remaining markers.

Katherine:
- Read over the Amazon RDS documentation. Decided the best route will be to go with a MYSQL database instance.
- Next Step: Start construction of the database.
- Blocked: Need to decide on an agreed password and username for the database so that we can all access it. Having some issues understanding Amazon VPC which is needed to move forward.

Issues Resolved:
- Used the data gathered from the scraper to send the rest of the station information on to Emma. Block removed.
- Decided on username and password for Amazon RDS.

Sprint 1: Retrospective 27th March 2017

Attended By:
Customer: Devin Stacey
Team Members: Pamela, Katherine, Emma.

Discussed current state of project:
- Front end HTML/CSS template is done, with all markers on the map. Pop up window showing individual station information appears for each marker. Information currently pulled from a static file (this will be refactored as part of the flask application).

- Web Scraper is scraping static and dynamic data from the Dublin Bikes API and storing it locally in text files in JSON format. The program is running continuously on EC2 – using screen – and storing the data there locally.

- Initial database is set up on Amazon RDS. Security groups are set to open so that it can be accessed remotely.

Issues to be addressed:
- We realised that the design of our database was flawed after Monday's lecture. We will have to push that back and prioritise redesigning and implementing it in Sprint 2. In hind-sight we could have spent more time thinking through and discussing out the design of the database before beginning implementation.

- Error in the files outputed by the scraper. The json dump is producing additional unwanted forward slashes.

Challenges in Sprint 1:
- Encountered some issues using git with eclipse – committed a mistake and had to try to reset the repository to a previous commit, as realised that this would affect other team members and could cause conflict issues. Resetting didn't have the expected results and deleted some of the files – had to go back through old commits to piece back together the appropriate file, reproduce it locally and commit again.

Feedback from customer:
- After initial functionality priority focus should be features like expanding on a historical way of accessing the data (real time updates), predictions. Weather and heatmap sensors are prioritised lower than the above features, but can be considered for Sprint 3.

- Ensure that as we move forward into the Flask application part of the project that we design efficient SQL queries. Aim for response time – less than a second. High priority in terms of user retention. This should be taken into consideration as we decide how long to leave the scraper running for – the more data to check the longer the query will take.

Sprint 2: Planning Meeting 27th March 2017

Attended by:
Customer: Devin Stacey
Team Members: Pamela, Katherine, Emma.

Plan for Sprint 2:
- Discussed the redesign for the database – it needs to have 2 tables, one for static data and one for dynamic data. Possibly a table for predictive data in the future.

- Implement this new design through Amazon RDS. We were initially using workbench but based on advice given by Devin, we are going to implement the db through python instead to avoid unnecessary dependencies for the client's machine and to increase abstraction of the final product for the client.

- Once the database is ready, implement functionality in the scraper for writing to the database, using SQL alchemy and/or pymysql.

- Transfer data already collected in file format to the database.

- Start working on the Flask application – in particular, look at how to decompose this aspect of the project – as Devin advised that each team member should get experience with using Flask.

- Start working on additional features such as predictive analysis, real time responses etc.

Standup Meeting: 31st March 2017:

Katherine:
- Redesigned DB from one large table to a 2-table one-to-one relation structure.
- Set-up instance for DB in RDS and made it accessible to everyone - fixed DB security issues.
- Wrote and tested code to create said tables in the DB.
- Have read flask documentation and have installed flask, sqlalchemy, and other packages in preparation for next step
- Next step: create weather data scraper
- Not Blocked.

Pamela:
- Fine-tuned the scraper as far as possible whilst waiting for the DB
- Helped debug the db script via code editing
- Next step: address and fix issue with the time for the scraper and other bugs
- Not Blocked.

Emma:
- Refactored code – tidied it up
- Changed the layout of the project
- Researched flask and its implementation
- Next step: Work on the front end set up in HTML and CSS and create weather database and code to create table within said database
- Blocked: Need the DB populated and Flask to be set up before it is possible to access the data.

Standup Meeting: 5th April 2017:

Katherine:
- Signed up and read documentation for open weather map API
- Created weather data scraper
- Wrote code to populate weather DB
- Next step: create skeleton of flask app to process dynamic data and formulate SQL queries that users would use pertaining to said dynamic data
- Not Blocked.

Pamela:
- Used object mapping in SQL alchemy to optimise scraper
- Identified and fixed a problem with Mercian prime concerning the main scraper and the Dublin bikes DB (see sprint 2 challenges for more details)
- Helped debug the DB and scraper script via code editing
- Next step: Host the site on EC2
- Not Blocked.

Emma:
- Created a DB to contain the weather data
- Helped debug scraper code via code editing
- Next step: Create a flask app to process the static data from the Dublin bikes DB
- Not Blocked

Sprint 2: Retrospective 7th April 2017

Attended By:
Team Members: Pamela, Katherine, Emma.

Discussed current state of project:
All items listed below are complete and additive items to what has been mentioned in the previous sprints.

- The main scraper is running and transferring data from the Dublin Bikes API, storing it to text files and then sending those files to and populating an appropriate database that has been set up with two tables – one for static and another for dynamic data. This improved database is set up on Amazon RDS. Security groups are set to open so that it can be accessed remotely.

- The weather scraper is running and transferring data from the current open weather API to the weather data base.

- The weather data base, set up on Amazon RDS, is complete with appropriate tables. Security groups are set to open so that it can be accessed remotely.

- The website is now being hosted through EC2

- A Flask app has been designed to process the static data from the databases, turning it into a JSON file, and then pushing that to the front end. This was done via paired programming.

Issues to be addressed:
- A better grasp of the function of flask is necessary – although the project is moving along well, flask seems to be a bit of a road block wherein we require more information than has been given by the slides and some tutorials. This is particularly important in terms of dynamic data.

Challenges in Sprint 2:

- There were some issues getting the main scraper configured, specifically in that SQL columns were too narrow. The prime example of this was the last updated column which continuously noted that there were duplicate values, which should not have been the case (it was a composite key with station_number. Setting up the table in such a way should have, theoretically, made duplicate values impossible). After several hours of research, rewriting code, and checking the database, it was discovered that the data type given for last_updated, date time, was too small to contain the number it was receiving. The database attempted to execute the code despite this issue by plugging in the largest possible integer it could handle, in this case an integer known as the Mercian Prime. Because each entry was the Mercian Prime, there were obviously several duplicates. This problem which caused a lot of grief had a straightforward and quick fix and was solved by simply changing the data type for that column to BigInt.

- There was a problem concerning the flask app for static data in that the static data didn't work with the syntax .items. After much research, it was discovered that this is because JSON is a dictionary stored as a list. This resulted in code changes.

- Along similar lines, initially there was a problem with the weather scraper in that the code threw errors concerning the way the JSON file was accessed. The method we had first learned to access the open weather API JSON file was through java script with JSON. This is different than how it is done in python.

- There is apparently an issue with windows 7 pip installing the correct version of pymysql. The first thought was that pip3 could resolve the issue, however, this was not recognized by the command line. This means that one of the team mates, Katherine, cannot run any code requiring pymysql. Work arounds were developed so that if such a code written by her needed to be tested, it would be pushed to git, run, and any errors would be reported back to her.

Sprint 3: Planning Meeting 7th April 2017

Attended by:
Team Members: Pamela, Katherine, Emma.

Plan for Sprint 3:
- Discussed the completion of the dynamic flask app and have decided that this would best be done via paired programming.
- Decided to create and implement user stories.
- Planned other predictive analytic SQL queries concerning both bike and weather data which might be helpful to the user.
- Once these additional SQL queries are created, add them to the dynamic flask app and connect that to the front end.
- Discussed the front end, user view design in terms of said queries
- Discussed the creation of user stories
- Discussed the method and possibility of real-time functionality