Important Note

Before delving into the details of this group project, it is important to note that the burndown charts are partially inaccurate – Katherine Campbell, although part of the git repository, was not detected and incorporated for whatever reason in any of the burndown charts. This was discovered as of the last few days of the project and the customer was notified. Please refer not to the burndown charts, but the git commits for a more accurate representation of group member contribution.

Sprint 1:

Burndown:
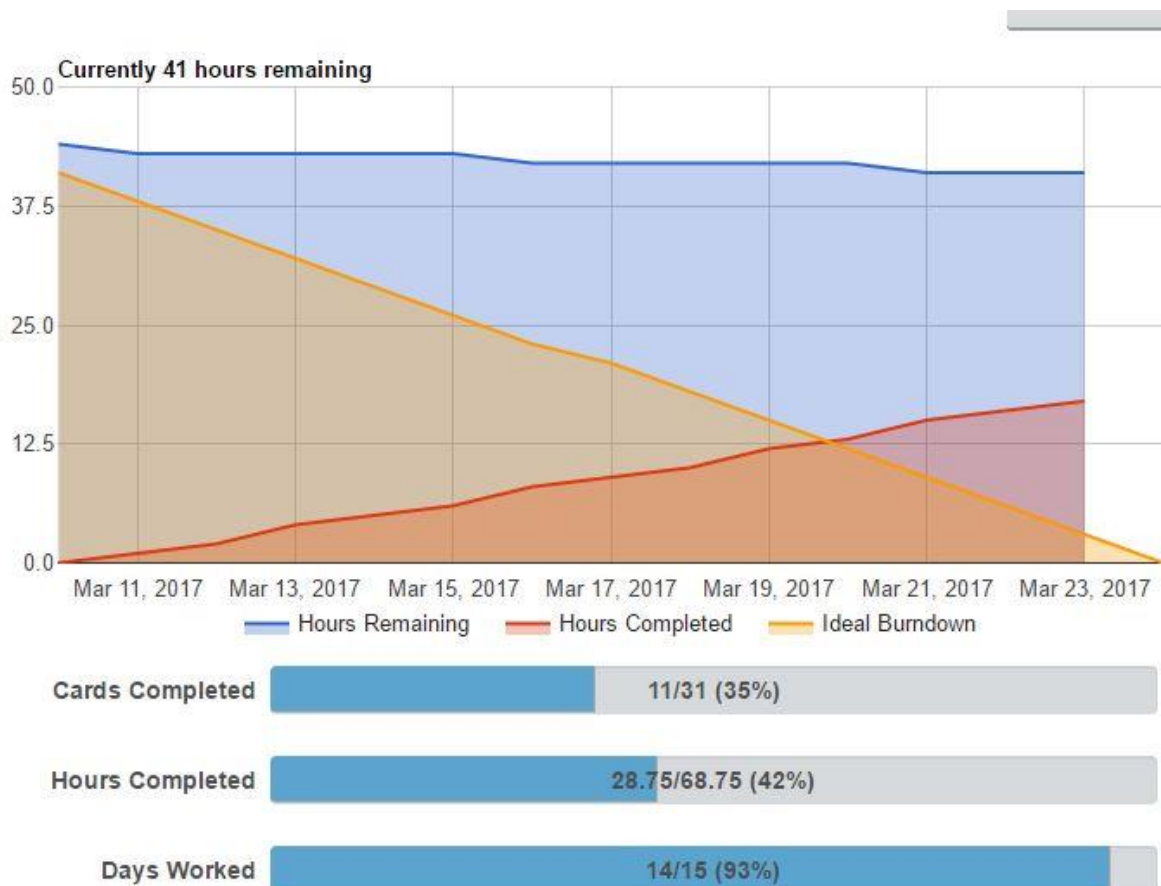
Comments on Burndown:

Sprint 1: Planning Meeting

Name: Pamela Kelly (scrum master)
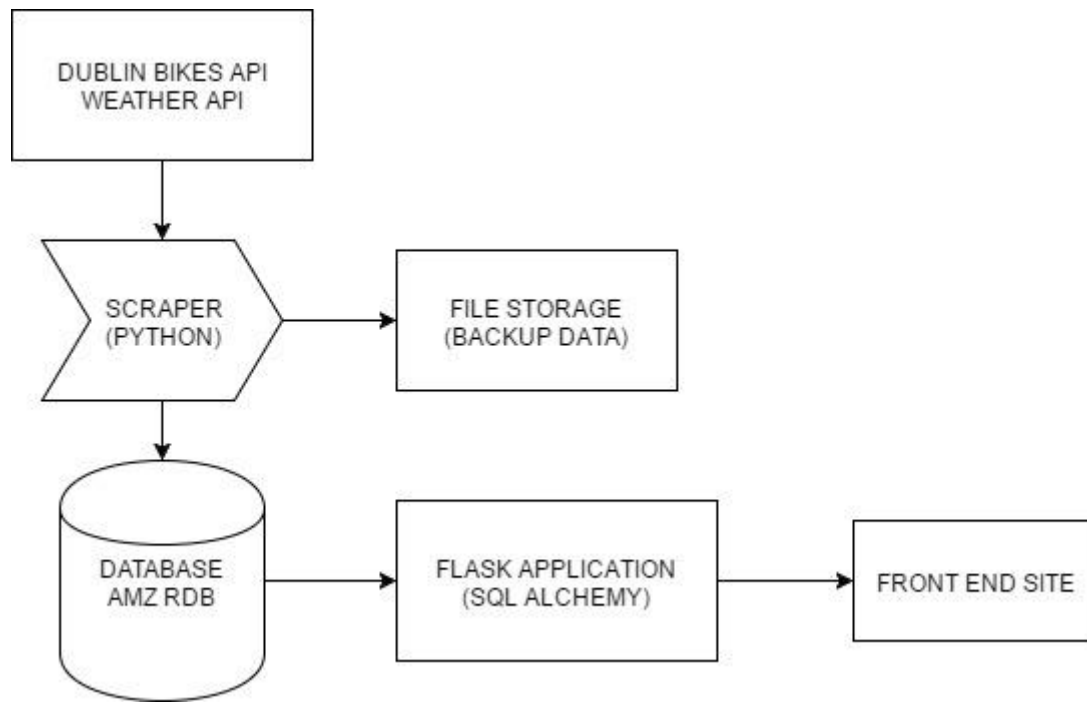Team: 7
Date: 10th March 2017

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

Plan for Sprint 1:

Visual Overview:

Tasks	Owner	Est	Due	Done	Notes
Research Dublin Bikes API	All team members	1hr	17th March	17th March	-
Research Weather API	All team members	1hr	17th March	17th March	-
Formalise structure & design for database	All team members – discussion	1hr	17th March	17th March	-
Research Amazon RDS Research Amazon Hosting	All team members	1hr	17th March	17th March	-
Research Flask	All team members	2hrs	17th March	17th March	-
Organise Trello & burndowns	Pamela	1hr			
Update Trello	All team members	-			-
Plan Stand-ups	Pamela	-			-
Write Scraper	Pamela	2hrs	25th March	Bugs – ran into Sprint 2	-
Set up Database	Katherine	2hrs	25th March	Issues – ran into Sprint 2	-
Write initial front end HTML/CSS/JavaScript	Emma	2hrs	25th March		-

Set up skeleton Flask App	TBD	-	-	-	-

Stand-up Meeting/Scrum:

<u>Name:</u> Pamela Kelly (scrum master)
<u>Team:</u> 7
<u>Date:</u> 14th March 2017

Attended by: Emma Byrne, Katherine Campbell, Pamela Kelly

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.

<u>Since Last Meeting:</u>

<u>Katherine:</u>
- Looked at project organisation tools, i.e. Trello.
- Started researching Flask, looked at Flask tutorials, created test flask apps.
- Looking at ways to develop the database from a user point of view.
- Not Blocked.

<u>Pamela:</u>
- 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.
- Not Blocked.

<u>Emma:</u>
- Looked at Dublin bikes API.
- Looked at scraping the API and using javascript to access information.
- Researched Amazon S3 and RDS.
- Not Blocked.

<u>Tasks From Now Until Next Meeting:</u>

Tasks	Owner	Est	Due	Done	Notes

Set up Database with Amazon RDS and MySQL	Katherine	2hrs	25th March		
Write Scraper in Python	Pamela	2hrs	25th March		
Front End with Map HTML/JavaScript/CSS	Emma	2hrs	25th March		

Stand-up Meeting/Scrum:

Name: Pamela Kelly (scrum master)
Team: 7
Date: 17th March 2017

Attended by: Emma Byrne, Katherine Campbell, Pamela Kelly

Since Last Meeting:

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.
- Not Blocked.

Emma:
- Started working on the frontend 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.
- Not 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.
- Not Blocked.

Tasks From Now Until Next Meeting:

Tasks	Owner	Est	Due	Done	Notes
Writing Scraper	Pamela				
Backup Data in Files	Pamela				
Set up markers for bikes stations on map	Emma				
Redesign Colour Scheme	Emma				

Set up Database	Katherine				

Stand-up Meeting:

Name: Pamela Kelly (scrum master)
Team: 7
Date: 17th March 2017

Attended by: Emma Byrne, Katherine Campbell, Pamela Kelly

Since Last Meeting:

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.
- 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.
- 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.
- Learned how create tables from python.
- 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.

Tasks From Now Until Next Meeting:

Tasks	Owner	Est	Due	Done	Notes
Get Scraper running continuously on EC2	Pamela				

Set up onclick info windows for markers on map	Emma				
Start construction of Database	Katherine				

Sprint 1: Retrospective

Name: Pamela Kelly (scrum master)
Team: 7
Date: 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 on EC2 (and backed up on local machine aswell) in text files. The scraper is running continuously on EC2 using screen.
- 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 output by the scraper. The json dump is producing additional unwanted forward slashes. If we were to lose data and have to repopulate the database using these files, this would make it more difficult to parse.

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 I 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. I had to go back through old commits to piece back together the appropriate file, reproduce it locally and commit again (Pamela).

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 1 Overview:

Name: Emma Byrne   Group: 7   Sprint: 1   Date:

Percentage of Group Submission: 33%

Skills Practised: HTML, CSS, JavaScript, incorporating Google Maps on front end

Learning Done: Research on Dublin Bikes API, Amazon RDS, EC2, python scrapers (brief)

Goals: Set up Flask app and incorporate Flask on the front end, setting up database for the weather information

Name: Katherine Campbell   Group: 7   Sprint# 1   Date:

Percentage of Group Submission: 33%

Skills Practised: Flask (made practice apps), creating RDS MYSQL database

Learning Done: ins and outs of Amazon RDS and VPC. Learned about Flask's use and basic functions

Goals: Redesign database structure, get it populated with data, identify the information we want and work on related queries to database

Name: Pamela Kelly   Group: 7   Sprint# 1   Date:
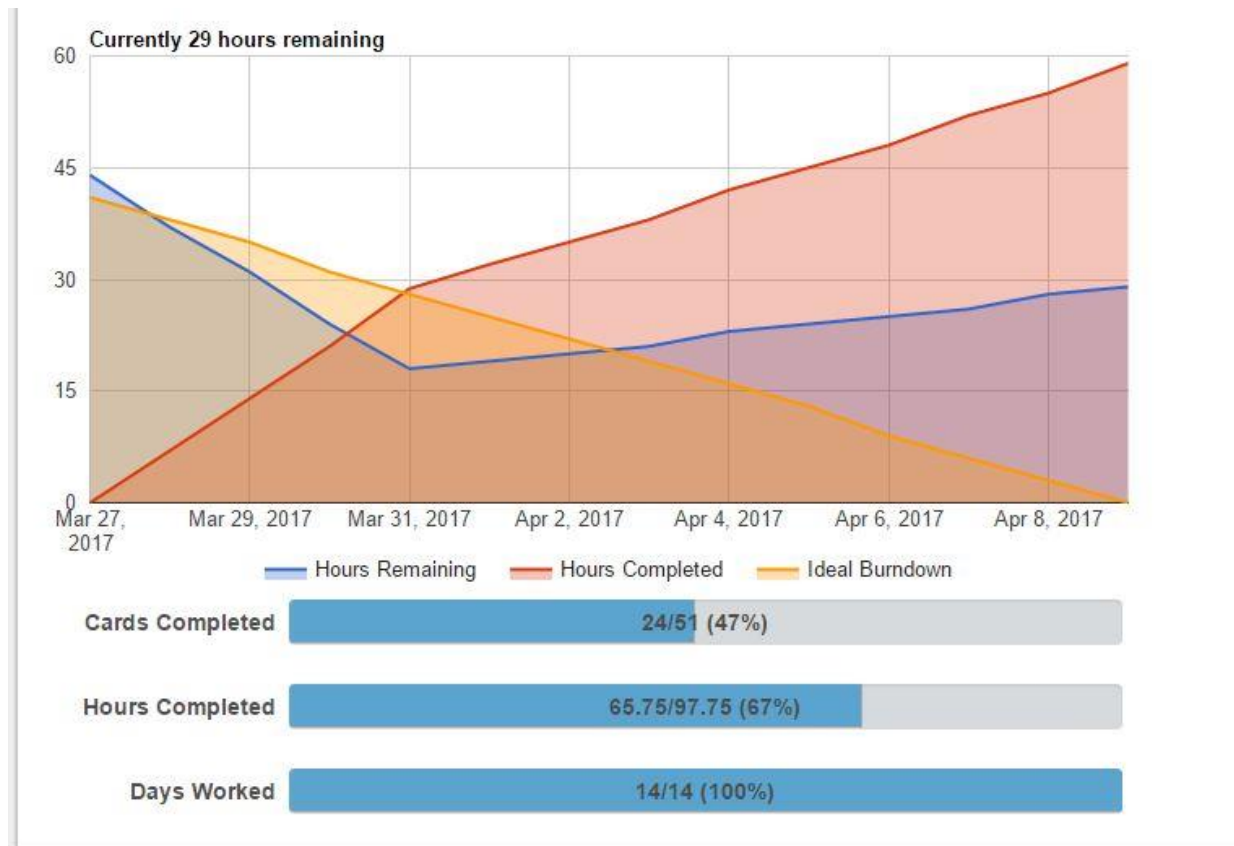
Percentage of Group Submission: 33%

Skills Practised:

Learning Done:

Goals:

Sprint 2:

Burndown:

Currently 29 hours remaining

Cards Completed: 24/51 (47%)
Hours Completed: 65.75/97.75 (67%)
Days Worked: 14/14 (100%)

Comments on Burndown:

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.
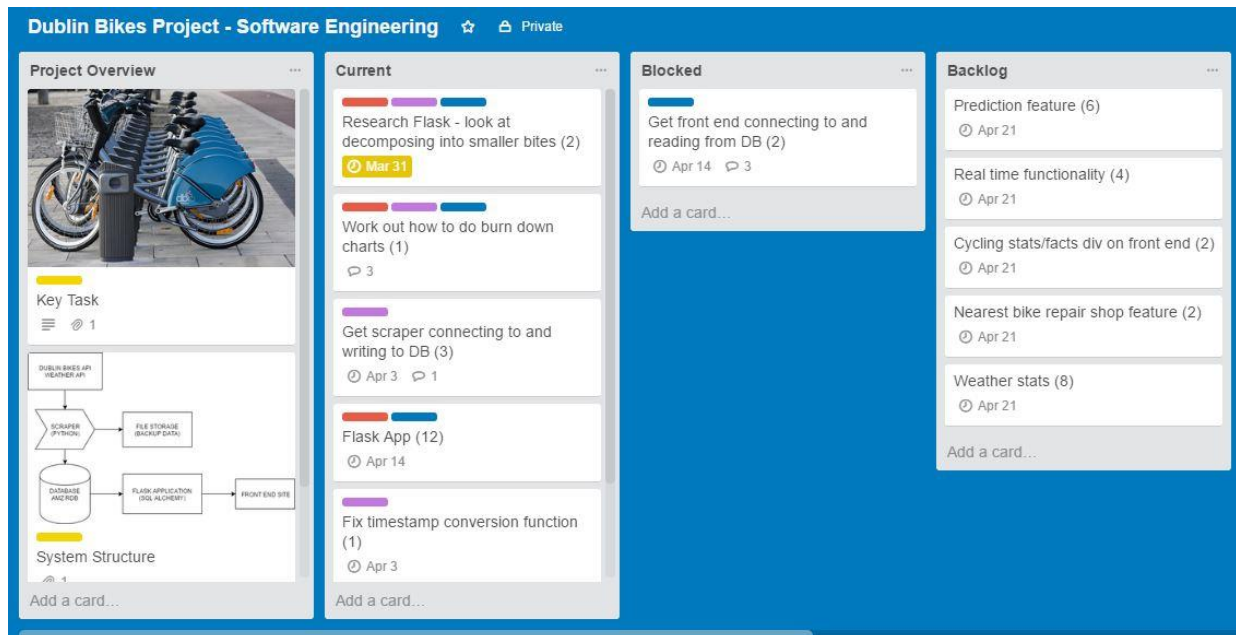
Stand-up Meeting: 31st March 2017:

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.

Dublin Bikes Project - Software Engineering ☆ 🔒 Private

Project Overview

Key Task

Dublin Bikes API / Weather API
Scraper (Python) → File Storage (Backup Data)
Database AWZ RDB → Flask Application (SQL Alchemy) → Front End Site

System Structure

Add a card…

Current

Research Flask - look at decomposing into smaller bites (2)
🕒 Mar 31

Work out how to do burn down charts (1)
💬 3

Get scraper connecting to and writing to DB (3)
🕒 Apr 3 💬 1

Flask App (12)
🕒 Apr 14

Fix timestamp conversion function (1)
🕒 Apr 3

Add a card…

Blocked

Get front end connecting to and reading from DB (2)
🕒 Apr 14 💬 3

Add a card…

Backlog

Prediction feature (6)
🕒 Apr 21

Real time functionality (4)
🕒 Apr 21

Cycling stats/facts div on front end (2)
🕒 Apr 21

Nearest bike repair shop feature (2)
🕒 Apr 21

Weather stats (8)
🕒 Apr 21

Add a card…

Stand-up 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

- 3rd April

- 4th April

- 6th April

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 2 Overview:

Name: Emma Byrne   Group: 7   Sprint# 2   Date:

Percentage of Group Submission: 33%

Skills Practised: Flask, JavaScript, creating a database in Amazon RDS using Python

Learning Done: Incorporating Flask on the front end. Setting up a weather database on RDS

Goals: Front end design, pass values from JavaScript into Flask to query the database

Name: Katherine Campbell   Group: 7   Sprint# 2   Date:

Percentage of Group Submission: 33%

Skills Practised: deleting, creating, and populating databases on rds via python code. creating tables and adding and deleting rows via code. Accessing APIs, working with APIs, reading JSON, interpreting and taking specific data from a JSON file and putting that in a table. Building a scraper with python, attaching code to databases and APIs, securely accessing APIs via code. Debugging.

Learning Done: Security on Amazon RDS and workarounds - learning to set everything to open access. how to work and manipulate tables with python and python packages. Learned how to make code accessing DBs secure. Regarding python packages learning how to work with pip and working around libraries not installing properly (ie. pymysql) Learning how to access and scrape APIs via python and use that data to populate a db.vLearning how sql alchemy reasons JSON files.

Goals: Join the weather and bikes databases via code, create a live weather display on the website, create sql queries for dynamic data/predictive analysis and connect that with flask.

Name: Pamela Kelly   Group: 7   Sprint# 2   Date:

Percentage of Group Submission:

Skills Practised:

Learning Done:

Goals:

Sprint 3:

Burndown:

Comments on Burndown:

Sprint 3: Planning Meeting 7th April 2017

Attended by:
Customer: Devin Stacey
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
- Discussed the addition of extra features – distance between each station etc.
- Discussed the challenges facing us in designing our queries as to maximise performance and not have queries that take a long time to load.
- Discussed incorporating Google Charts to the front end

Stand up meeting 11th April 2017

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

Pamela:

- Getting hourly/daily queries set up
- Fixed the scraper – the connection issues
- Looked at connection and pooling and disposing of them before creating new ones
- Nearly finished hosting the app on ec2 with the flask app
- Did some pair programming with Emma and Katherine with regards to flask
- Next step: Work on queries to get data for daily/hourly dynamic information for stations and visualise this with Google charts
- Not blocked

Katherine:

- Created sql queries for querying dynamic/predictive data
- Tested connecting flask with the SQL queries
- Did some pair programming with Pamela and Emma with regards to flask
- Next step: Joining databases together
- Creating a div for where the weather data will be displayed
- Pair programming with Emma to get predictive analysis for weather data

Emma:

- Set up the flask app
- Linked the flask app to the front end via pair programming with Pamela
- Got the map with markers up and running via the flask app and javaScript
- Accessed static station information via the flask app
- Accessed weather and occupancy information via flask app but not very efficiently – Katherine will be working on queries for the weather app

- Next step: Linking everything to the front end – linking the weather information, google charts and other necessary information to the javaScript and HTML
- Pair programming with Katherine to get the predictive analysis for the weather
- Not blocked

Stand up meeting 14th April 2017

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

Pamela:
- Modularised and refactored the code – scraper.py
- Transferred all of the files stored on EC2 to the database
- Scraper now collecting data directly from the Dublin Bikes API
- Started working on the sql queries for hourly/daily dynamic data
- Looked at Google graphs
- Not blocked

Katherine:
- Looked up how to merge databases: three ways to do this
- Tried to determine the best way to do this
- Also looked at/practiced how to do this by writing a python function
- Pair programmed with Emma on writing dynamic flask apps with queries
- Testing merging databases
- Not blocked

Emma:
- Researched how to connect flask to javaScript and back again – get and post methods
- Further research into connecting Flask to HTML
- Looked into new design ideas for the front end
- Started working on ideas for querying the weather database
- Pair programmed with Katherine on writing dynamic flask app with queries
- Blocked: Can't quite figure out how to send a variable to the database via flask and get the results of the query displaying on the front end

Stand up meeting 18th April 2017

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

Pamela:
- Worked on (and still working on) fixing hosting issues
- Set up port forwarding with Apache
- Pair programmed with Emma on flask

- Looked at getting dynamic data for each station to display on the home page
- Next step: Keep working on hosting
- Get daily/hourly stats and create Google charts from that

Katherine
- Researched and wrote python/MySQL alchemy code to copy a table and its data from one database to another
- Checked and updated predictive queries (average bikes/stands currently available)
- Next step: Create a div in the HTML page and create a javaScript function to display live weather stats in that div
- Incorporate weather icons also

Emma
- Worked on passing the station number for a specific station into a sql query – pair programming with Pam – for now this value is only passed into a javaScript function
- Added banking available to the info window popup
- Made small changes to the CSS, HTML and javaScript files
- Next step: Get the station number for a station to pass into a sql query to get the occupancy information for a specific station
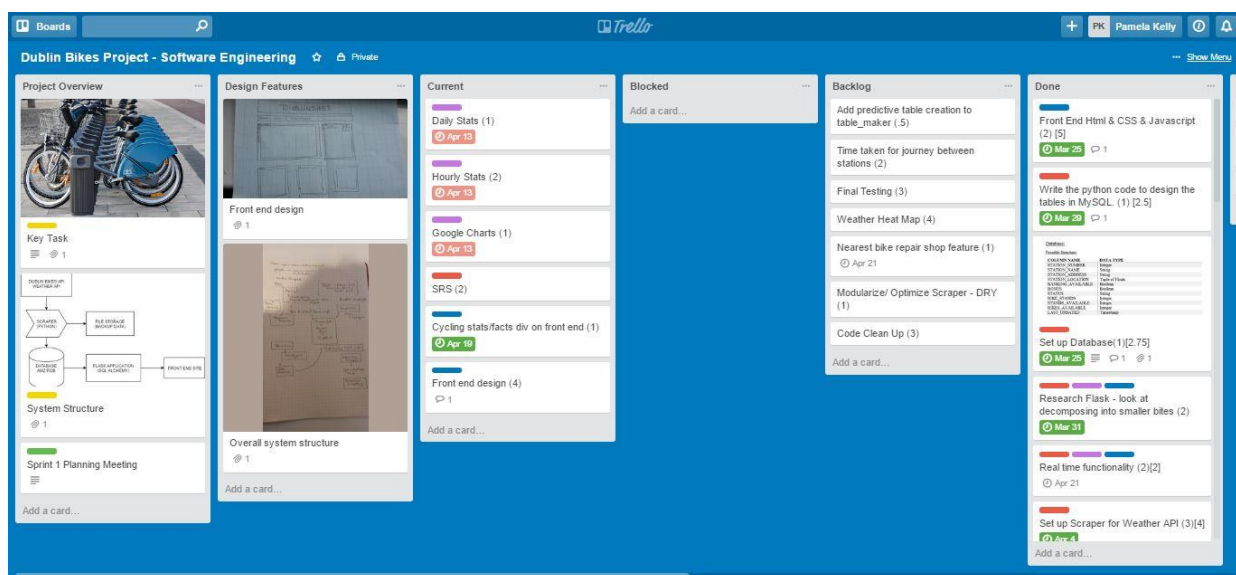
- 19th April

Stand-up meeting 21st April 2017

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

Pamela:

- Got the Google Charts working and incorporated them into the and displayed them on the front end
- Worked on getting the app running on EC2 and the website is now hosted on ec2

- Next step: Tidy up code and annotate code. Get documentation written up

Katherine:
- Worked on getting the live weather data displaying on the front end
- Incorporated Open Weather Map API via JavaScript and JQuery with HTML to display live weather data on the front end
- Incorporated weather icons in the weather information
- Next step: Tidy up code. Get documentation written up including the SRS

Emma:
- Worked on front end design, displaying everything together on the webpage in an attractive way
- Added a header and a footer along with various images and facts to the front-end design
- Next step: Tidy up and annotate code. Write up necessary documentation.

Sprint 3: Retrospective 22nd April 2017

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

Discussed current state of project:
- All goals of the project have been achieved
- The app/website is complete with all of the necessary features
- The website is up and running on an Amazon ec2 instance
- The website contains a map with markers indicating the location of all of the Dublin Bikes stands across the city
- The website displays live weather information
- The website supports interactivity by being able to select station markers for a specific station
- The website displays occupancy information for a specific station when it is selected
- Hourly and daily statistics are also displayed for a specific bike station

Challenges in sprint 3
- There were many issues when hosting the app on ec2. There were numerous import issues with python 3 modules which meant that we couldn't use the wsgi system. The solution to this was setting up port forwarding so that the site running on the local host was redirected to the public IP for the ec2.
- There were several JavaScript issues when trying to send parameters to the url and then passing that information into a sql query in the flask app. Solution: Emma and Pamela utilised pair programming to create an on-click function which is called within the loop when creating the markers so that the on-click was dynamic and specific for that station. This enabled the correct occupancy information to be displayed for whichever station was selected.

- Several issues occurred when trying to merge databases – the weather and bike databases via sql alchemy. Other issues encountered included copying the data from one database to another. The solution to this would have been to use the Amazon migration functionality but because this is a paid service, it was decided that the tables would not be merged.
- Other challenges included getting the live weather data to display on the front end using jquery but this ended up being because of the order in which it was being called on the front end.

Features in Backlog
- There were a few features in the backlog that were not added to the project due to time constraints. These included a heat map, calculating the time it would take to get from one stop to another and showing the nearest bike repair shop.
- The historical weather data was to be used so as to make predictions about the availability of bikes depending on the different weather conditions. For example, what would be the chance of getting a bike from a particular station at a certain time if it is raining.

Feedback from customer
- The project fulfils the necessary requirements as outlined in the documentation
- Ensure everything is working perfectly including hosting and statistical charts
- Ensure that the code can be run on different environments including different laptops and on ec2
- Ensure that no part of the application takes too long to load – map, weather, statistics etc.
- Ensure code is concise and not reused

Dublin Bikes Project - Software Engineering ☆ 🔒 Private

Project Overview

Key Task
≡ 📎 1

System Structure
📎 1

Sprint 1 Planning Meeting
≡

Add a card…

Design Features

Front end design
📎 1

Overall system structure
📎 1

Add a card…

Current

Daily Stats (1)
⏱ 13 Apr

Hourly Stats (2)
⏱ 13 Apr

Google Charts (1)
⏱ 13 Apr

SRS (2)

Cycling stats/facts div on front end (1)
⏱ 19 Apr

Front end design (4)
💬 1

Add a card…

Blocked

Add a card…

Backlog

Unit Testing (3)

Add predictive table creation to table_maker (.5)

Time taken for journey between stations (2)

Weather Heat Map (4)

Nearest bike repair shop feature (1)
⏱ 21 Apr

Modularize/ Optimize Scraper - DRY (1)

Code Clean Up (3)

Add a card…

Done

Front End Html & CSS & Javascript (2) [5]
⏱ 25 Mar   💬 1

Write the python code to design the tables in MySQL. (1) [2.5]
⏱ 29 Mar   💬 1

Database:
Possible Structure:

COLUMN NAME	DATA TYPE
STATION_NUMBER	Integer
STATION_NAME	String
STATION_ADDRESS	String
STATION_LOCATION	Tuple of Floats
BANKING_AVAILABLE	Boolean
BONUS	Boolean
STATUS	String
BIKE_STANDS	Integer
STANDS_AVAILABLE	Integer
BIKES_AVAILABLE	Integer
LAST_UPDATED	Timestamp

Set up Database(1)[2.75]
⏱ 25 Mar   ≡   💬 1   📎 1

Research Flask - look at decomposing into smaller bites (2)
⏱ 31 Mar

Real time functionality (2)[2]
⏱ 21 Apr

Add a card…

Done

Set up Scraper for Weather API (3) [4]
⏱ 4 Apr

Get front end connecting to and reading from DB (2)[3]
⏱ 9 Apr   💬 3

Get scraper connecting to and writing to DB (3) [3]
⏱ 3 Apr   💬 1

Access static data using Flask (1) [0.5]
⏱ 7 Apr

Get the scraper running continuously on Amazon EC2 (2) [4]
⏱ 25 Mar   💬 1

Set up scraper for Dublin Bikes API (2) [2]
⏱ 25 Mar   💬 1

Add a card…

Done

Look at relevant technologies: Amazon RDB, Amazon Hosting, Flask, SQL Alchemy (6) [6]
⏱ 25 Mar

Redesign DB Layout(1)[.5]

Configure html templates to work with Flask (2)[1]
⏱ 9 Apr

get the DB code w/ tables up and running on DB. (2)[3]

Access Dynamic Data with Flask (1) [0.5]
⏱ 8 Apr

Change from pymysql to SQLAlchemy (.5) [.5]
⏱ 31 Mar

Fix timestamp conversion function

Add a card…

Done ...

Fix timestamp conversion function (1)
🕐 3 Apr

Process JSON from Flask with Javascript and display on front end (2)[3]
🕐 10 Apr

User stories (1)[1]
🕐 9 Apr 💬 2

Format API responses to feed into write to db function (2) [4]
🕐 3 Apr

Make the weather database (2)[2]
🕐 4 Apr

Work out how to do burn down charts (1)
💬 3

Add a card…

Look at hosting the website with Amazon (1) [3.5]
🕐 7 Apr

Transfer collected data from files to db (2) [10]
🕐 3 Apr

Connecting javaScript/HTML to Flask to query the database (2)[2]

Import Weather Database table structure to Bikes Database (1.5)[3]

Pandas Analysis - SQL Queries(2)
🕐 8 Apr

Display weather info on front end (1) [5]

Backlog ...

Add predictive table creation to table_maker (.5)

Time taken for journey between stations (2)

Weather Heat Map (4)

Nearest bike repair shop feature (1)
🕐 Apr 21

Modularize/ Optimize Scraper - DRY (1)
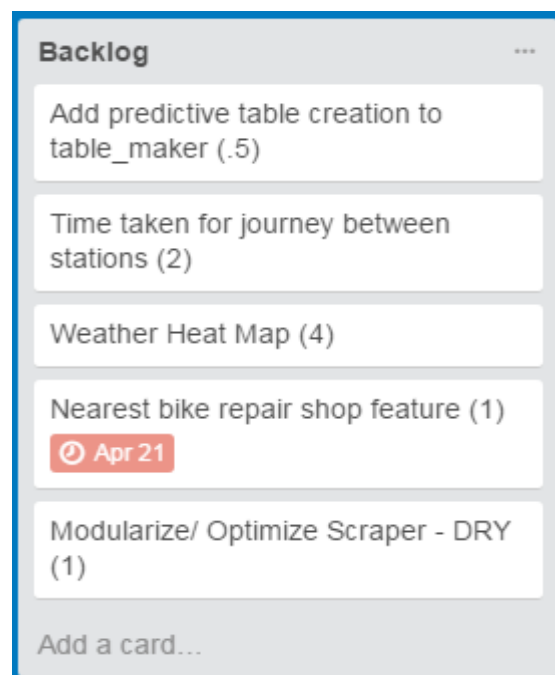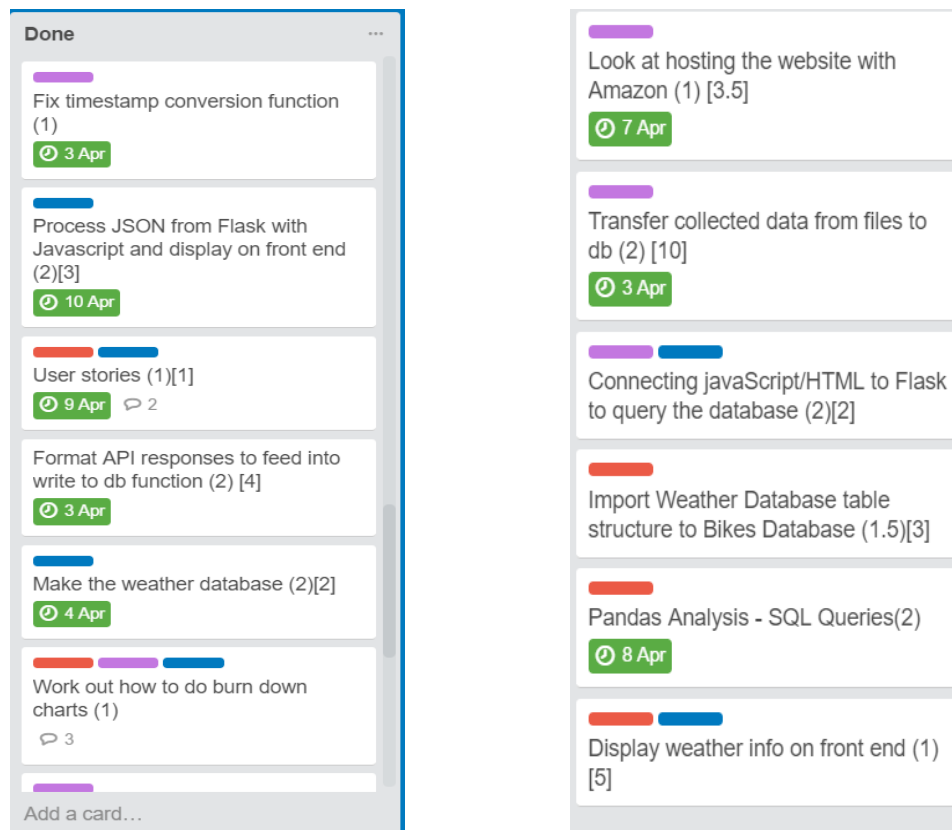
Add a card…

Fig. X: The remaining backlog on the final day

Sprint 3 Overview

Sprint 3:

Name: Emma Byrne   Group: 7   Sprint# 3   Date:

Percentage of Group Submission: 33%

Skills Practised: HTML, JavaScript, CSS, Flask, Python

Learning Done: Passed JavaScript variable to Flask to query the database about occupancy information for a specific station – peer programming with Pamela

Final Thoughts: Successful project, set out what we wanted to achieve. Given more time some of the features in the backlog could have been incorporated.

Name: Katherine Campbell   Group: 7   Sprint# 3   Date:

Percentage of Group Submission: 33%

Skills Practised: Writing python code copying a table from one db into another using sql alchemy. Dumping data import/export with mysql, working with dynamic flask applications, Queries in sql alchemy. JavaScript, HTML, JQuery, JQuery uses connecting with HTML, connecting and reading JSON from Openweathermap API, writing SRS.

Learning Done: Learned three ways to copy a table from one db to another - using dumps, CREATE TABLE..LIKE db1.table; INSERT INTO db2 SELECT * FROM db1.table, and via python using alchemy, 2 engines and metadata. Copying said table and data with python code. Learned that rds allows tables to be copied but not data. Must use charged migration service. Learned JQuery. Learned how to use JQuery with APIs and html. Learned how to write sql queries, select only the most updating entries. Learned how to work around the previous pymysql issue with mysqldb. Learned how to work with predictive data, and tie said data into a dynamic flask app to get it to the front end.

Final Thoughts: Overall a successful project – ensured a very steep learning curve to gain a lot of valuable experience. Now much more comfortable with sql, and have learned something about Flask and JQuery. We were so close to incorporating predictive weather data and had we stored in the same data base that wouldn't have been an issue. Would have liked more time to focus only on this project – many cool items in the backlog would have been nice to incorporate (e.g. heatmap, search bar, predictive weather data, etc.). Otherwise, very pleased.

Name: Pamela Kelly   Group: 7   Sprint# 3   Date:

Percentage of Group Submission:

Skills Practised:

Learning Done:

Goals: