SOFTWARE ENGINEERING

CARA DELOREY: 16201120
DARRAGH MINOGUE: 16200080
TAO LI: 16203180

Assignment 4

Contents

Purpose

The goal of Assignment 4 was to develop a web application to display occupancy and weather information for Dublin Bikes following a scrum methodology. The purpose of this document is to outline the project deliverables, detail the project management approach and demonstrate our learning as a team.

Project Summary

The project took place over three two-week sprint cycles between March 10-April 23, 2017. It had seven main deliverables:

1. Data collection through API
2. Data management/storage in DB on AWS
3. Display bike stations on map
4. Occupancy information
5. Weather information
6. Interactivity (click, API request, handle response)
7. Project served on EC2: http://ec2-52-37-147-107.us-west-2.compute.amazonaws.com:5000/
8. GitHub repo: https://github.com/minogud2/softEngAssignment4

The application developed met all objectives specified and can be accessed via: xxxxxx. In doing so, it addressed the following user stories developed:

As a…. (SITUATION)	I want to (MOTIVATION)	So I can (EXPECTED OUTCOME)	Priority
As a basic user,	I want to see a google map of Dublin	So I can click on a station for bike availability and address information	1
As a daily user	I want to have a search function for a station	So I can quickly access my regular station.	6
As a daily user,	I want to see the average occupancy time around my search	So I can plan my imminent trip	3
As a basic user,	I want to see available stations from adequate glance	So that I can see where I should get a bike	2
[Stretch Card] As a basic user,	I want to access the website on my smartphone and provide my locational details	So I can see the availability of bike stands in my proximity.	7
As a commuter,	I want to view weather info	So I can decide whether or not I want to get the bus.	4
As a future user,	I want to see daily and weekly averages	So I can plan a future trip	5

In total, six from the seven user stories were achieved and are visible from the screenshots below.

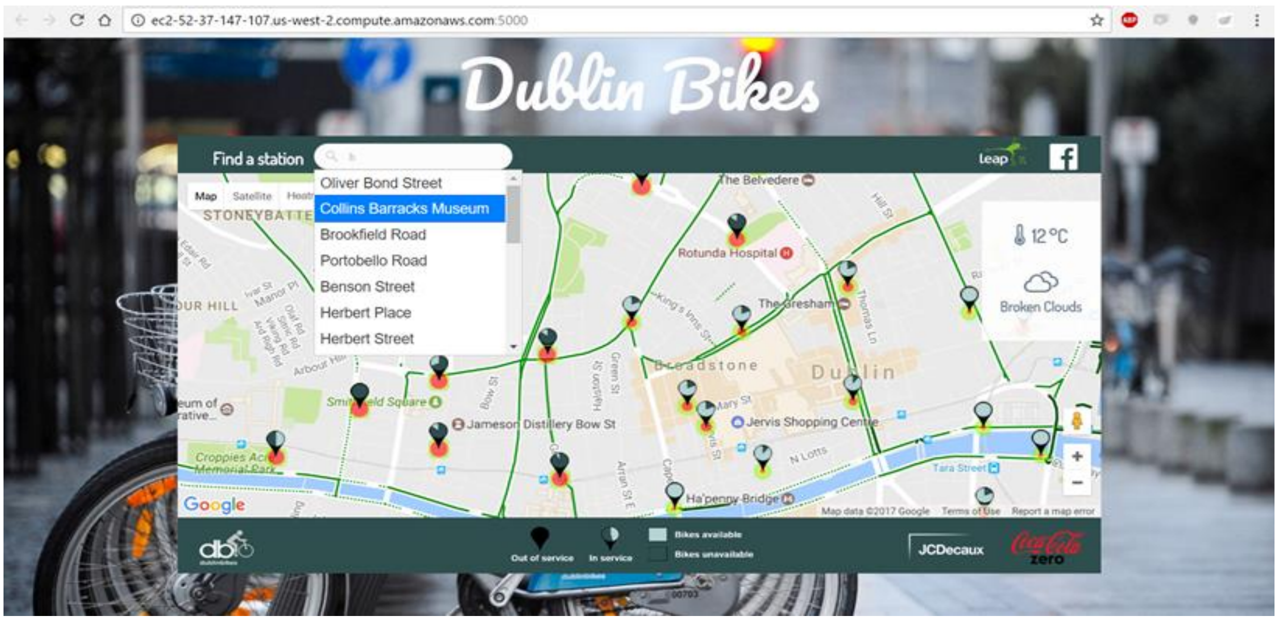
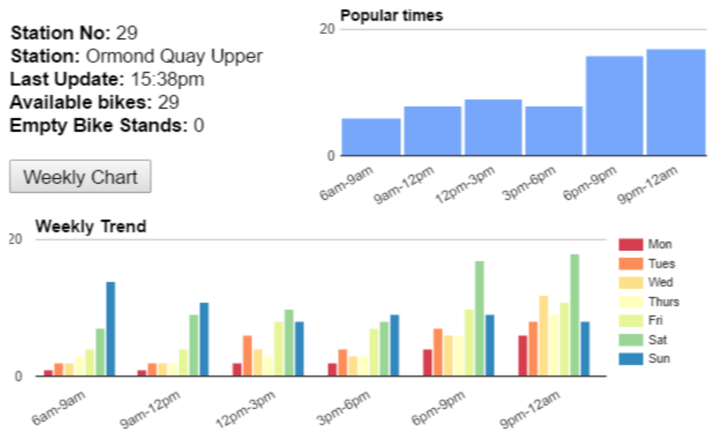


Figure 1.1 Dublin Bikes Project Deliverable

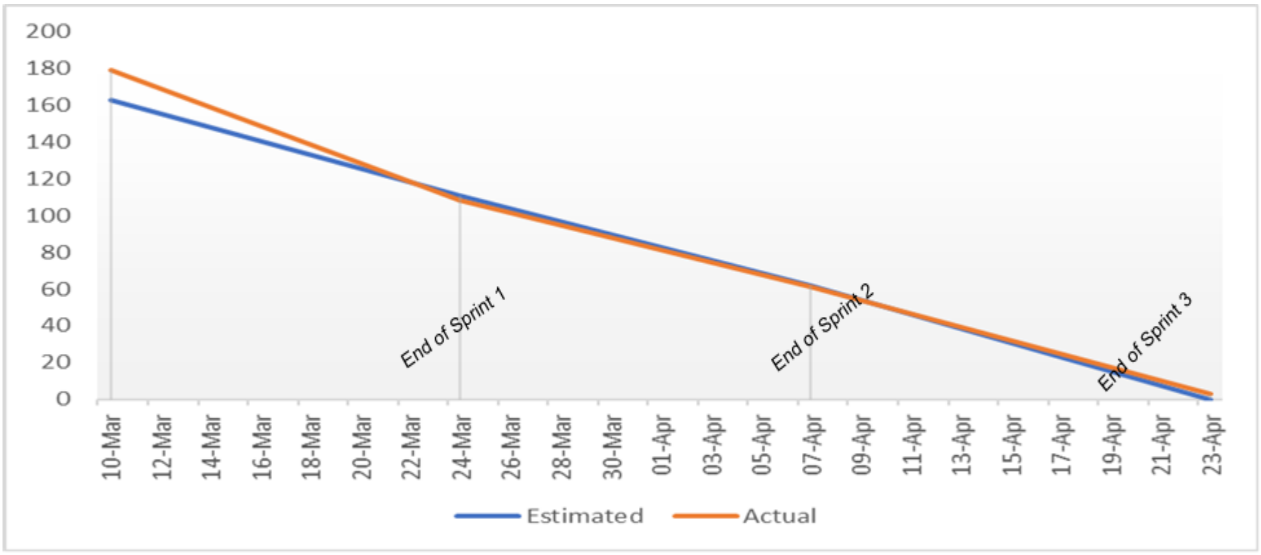
In addition to the seven deliverables, a major component of the assignment was the project management approach. Specifically, teams were required to follow scrum project management, select site features to develop, produce a product backlog, and meet deliverables within each sprint. This document addresses each of these components. It outlines the team's approach to Scrum according to each Sprint. The log of meeting notes can be seen in Appendix D.



Station No: 29
Station: Ormond Quay Upper
Last Update: 15:38pm
Available bikes: 29
Empty Bike Stands: 0

Figure 1.2 Dublin Bikes Project Deliverable. Daily and weekly statistics per station.

Project Burndown Chart



The goal of a sprint review is not to give a demonstration; rather, the goal of the sprint review is to inspect and adapt the product that is being built

Sprint 1

Dates: 10-25 March 2017
Scrum Master: Cara Delorey

Sprint Preparation and Planning

Before our first sprint began, the group held informal meetings to outline goals for the sprints and discuss the general structure of the project. We agreed on what project aspects would be prioritised and what would be deferred. Tasks were then organised sequentially to prevent us from creating roadblocks in our development process.

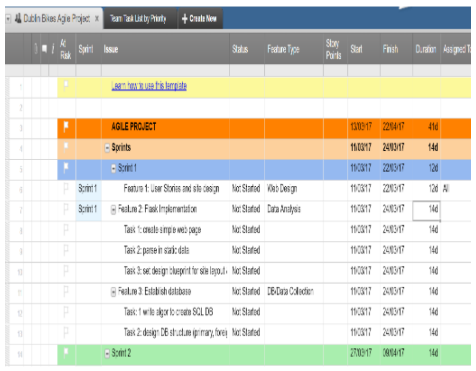
Following these talks, a basic backlog was developed and tasks were allocated based on Sprint 1 goals.

Delivered Materials and Tasks Completed

1. *Management and Communication*

The Scrum Master was tasked with researching tools and software for sprint management. The team's primary communication was through Slack and Facebook Messenger. We kept contact with our product owner, Satish, via email and during class hours.

To keep track of what tasks we were working on, Trello boards were chosen. Smartsheet was considered and piloted (right) but it had an unyielding and unappealing interface. Trello was viewed as a simple, concise management tool ideal for meeting our management requirements with a user-friendly interface. It's quick, easy to edit and provides a straightforward list system to subdivide project workload. At a glance, it was possible to see what work was done and what needs to be completed (See figure 2.2.)

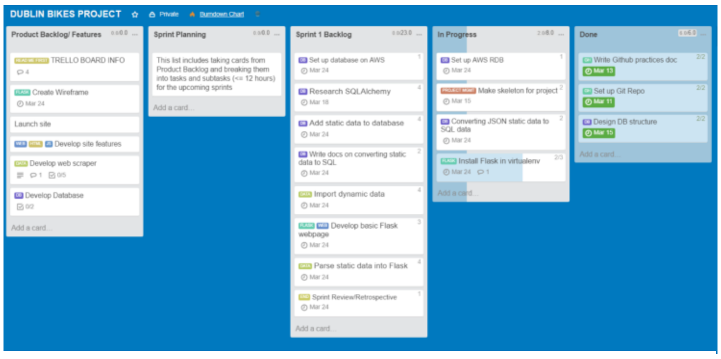


Dublin Bikes Agile Project | Team Task List by Priority | Create New

	At Risk	Sprint	Issue	Status	Feature Type	Story Points	Start	Finish	Duration	Assigned T
1			Learn how to use this template							
2										
3			AGILE PROJECT				13/03/17	22/04/17	41d	
4			Sprints				11/03/17	24/03/17	14d	
5			Sprint 1				11/03/17	22/03/17	12d	
6		Sprint 1	Feature 1: User Stories and site design	Not Started	Web Design		11/03/17	22/03/17	12d	All
7		Sprint 1	Feature 2: Flask Implementation	Not Started	Data Analysis		11/03/17	24/03/17	14d	
8			Task 1: create simple web page	Not Started			11/03/17	24/03/17	14d	
9			Task 2: parse in static data	Not Started			11/03/17	24/03/17	14d	
10			Task 3: set design blueprint for site layout	Not Started			11/03/17	24/03/17	14d	
11			Feature 3: Establish database	Not Started	DB/Data Collection		11/03/17	24/03/17	14d	
12			Task: 1 write algor to create SQL DB	Not Started			11/03/17	24/03/17	14d	
13			Task 2: design DB structure (primary, forei	Not Started			11/03/17	24/03/17	14d	
14			Sprint 2				27/03/17	09/04/17	14d	

Figure 2.1. Smartsheet pilot.

To facilitate code sharing, we set up a Git repository and applied a skeleton structure for our project. Branches were then made to practise pushing, pulling, and resolving merge conflicts. Documentation was stored on the repository and for other project management needs, online resources were shared across Slack.



DUBLIN BIKES PROJECT | Private | Burndown Chart

Product Backlog/ Features
- TRELLO BOARD INFO
- Create Wireframe (Mar 24)
- Launch site
- Develop site features
- Develop web scraper
- Develop Database

Sprint Planning
- This list includes taking cards from Product Backlog and breaking them into tasks and subtasks (<= 12 hours) for the upcoming sprints

Sprint 1 Backlog
- Set up database on AWS (Mar 24)
- Research SQLAlchemy (Mar 18)
- Add static data to database (Mar 24)
- Write docs on converting static data to SQL (Mar 24)
- Import dynamic data (Mar 24)
- Develop basic Flask webpage (Mar 24)
- Parse static data into Flask (Mar 24)
- Sprint Review/Retrospective (Mar 24)

In Progress
- Set up AWS RDB (Mar 24)
- Make skeleton for project (Mar 15)
- Converting JSON static data to SQL data (Mar 24)
- Install Flask in virtualenv (Mar 24)

Done
- Write Github practices doc (Mar 13)
- Set up Git Repo (Mar 11)
- Design DB structure (Mar 15)

Figure 2.2. Trello sheets (initial backlog)

User stories were developed for the project individually and collated at the end of the sprint. The focus remained on what information was asked for and how best to deliver it to the user. These stories were used to flesh out site features, steer product development and guide the project backlog. They were reviewed and updated on a regular basis.

2. *Site Development*

All team members researched flask to gain a better understanding of the software and the tasks involved in hosting the site including software templates and Flask notations for displaying features. Initially, it was thought that the team could design a site in HTML/CSS/JS and then pass it into a Flask application in the final sprint. From our research, it was clear there was a need to develop a Flask wireframe in the first week of Sprint 2 and integrate components as we went along in the process. An online wireframe design tool was therefore used to create a mock-up of the site (**see Figure 2.3.).** This design integrated the user stories and feature positioning. This included required and potential features to the design (stretch cards). A site template was then developed to test our code and allow us to apply features quickly during Sprint 2.

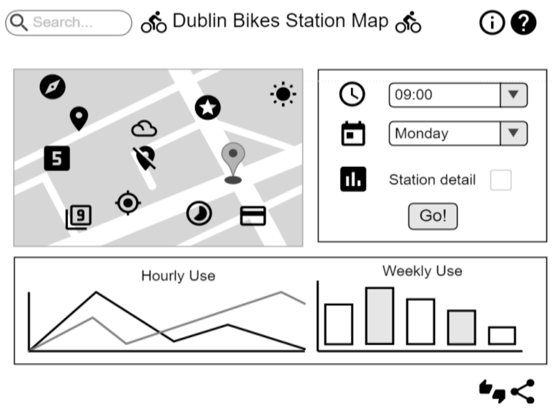


Figure 2.3. UI Design

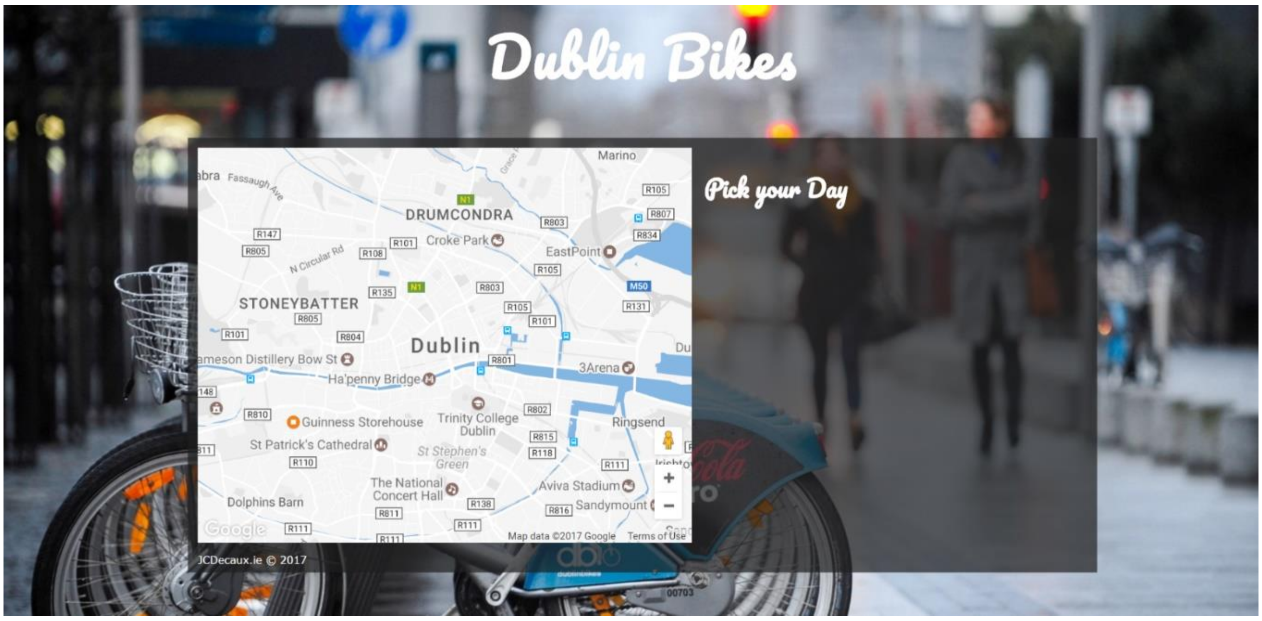


Figure 2.4. Basic HTML/CSS Template.

3. *Data Collection and Database Management*

The web scrapper and database design were given top priority during the sprint. Considering the product requirements, three datasets were required: **1)** static data from Dublin bikes API, **2)** dynamic data from Dublin bikes API, and **3)** dynamic weather from OpenWeatherMap API. A database was designed and from this the scraper was set up using python on an AWS instance.

Documentation was created for each of these processes to ensure all team members understood each other's work. At the end of the sprint all code was merged.

Furthermore, the Flask application was developed and the test sets required outlined for the back end. From reviewing the project structure, a list of security and error handling measures were identified to review the code in future sprints.

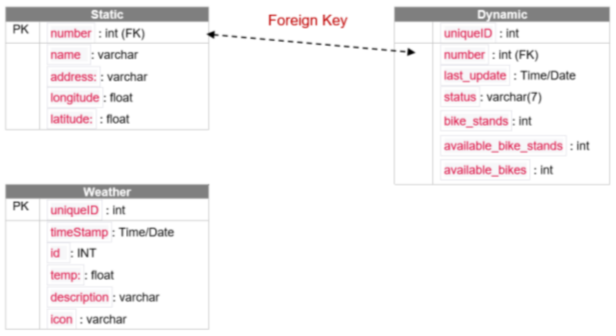


Figure 2.5. Database Design. ER Diagram. ER Source: http://www.datanamic.com/dezign/erdiagramtool.html
See appendix C.

Workload

Backlog	Assigned to	Estimated	Actual
Parsing of Dynamic Data into a Database Structure on AWS	Tao	6	6
Meetings	All	5	6
Review Notes Developed	Cara	1	2
Summary documentation and log of meetings and stand ups	Cara	2	2
Basic HTML/CSS site	Darragh	2	2
Develop user stories	All	1	1
Research GitHub for project management and set up repos	Darragh	3	3
Write GitHub practices guide	Darragh	2	2
Retrospective conducted	All	3	3
Set up AWS instance for project	Tao	2	2
Design DB structure	Darragh	2	2
Make skeleton for project	Darragh	2	2
Parse static data to database	Darragh	2	2
Develop JSON to SQL algorithms	Darragh	2	2
Research Project Management needs and set up working environment	Cara	4	4
Research potential security weak points	Cara	3	2
Implement Trello lists, Burndown Chart for Sprint 1	Cara	2	2
Write up individual learning journals	All	3	3
	Total	**47**	**47.5**

Retrospective

What went well during the sprint cycle?

- **Preliminary Organisation.** The team successfully implemented a simple communication set up which served the development process in the first sprint. Slack and Facebook allowed team members to share thoughts, ideas, code snippets and online resources quickly.

- **Development environment established.** Github was the code sharing platform used for this project. Individual team members set up their preferred code development environment including Eclipse, Pycharm and Brackets. Programs like MySQL Workbench were chosen as an associated tool for data query development. Linux instances on AWS were run to insure all instructional setups were streamlined.

What went wrong during the sprint cycle?

- **Communication**. The first sprint occurred during the mid-semester break which limited the opportunities to hold regular group meetings. Stand-ups were limited to slack. Towards the end of the sprint, a few online calls were made to discuss the project progress. Tao spent most of the break in China. This limited our ability to communicate online about the project as Slack, Google and Facebook are banned in China. Darragh and Cara held a couple of meetings to discuss progress before Tao returned. Ultimately, the group communication for sprint 1 was limited but this was unavoidable. The next sprint should see an improvement in group discussion and we plan to hold regular stand ups.

- **Using GitHub for documentation.** For Sprint 1, documentation was also tracked in GitHub. However, due to the iterative nature of the documents, they led to numerous merge conflicts. As such, we decided that for Sprint 2, we would look to Google Drive to handle our project management documentation.

- **Different AWS Instances led to complications for development environments.** Team members had different AWS instances initially: Ubuntu vs Linux. This led to complications during set up as instructions for one instance was different on another. This caused hours of delays for team members.

What could we do differently to improve?

- **Site Features.** Discussion of feature development was limited in the first sprint. We planned to discuss these tasks in more depth in the second sprint.

- **Duplication of Efforts.** One benefit of sprint management is preventing issues that delay project development. The given assignment involved several areas we had prior experience in and several more we were unfamiliar with. Given this, the first sprint involved extensive research on databases, web scraping and Flask. This inevitably resulted in some duplication of code and tasks. Although this may have been necessary during the first sprint so that any group member can work on any section of the project, going forward this should be minimized.

- **Task Prioritisation and Project Design.** In the second sprint, it was necessary to reconsider our approach to the remaining project tasks. Improved communication through short to medium length meetings was key for the team to develop a clearer understanding of the product features and functionalities. Tasks for Sprint 2 must be assigned time estimates and tracked effectively.

Sprint 2

Dates: March 24-April 7, 2017
Scrum Master: Darragh Minogue

SPRINT REVIEW

1. *Management and Communication*

During Sprint two, the project backlog was updated based on our retrospective. Sprints were categorised in Trello using coloured labels and time estimations were completed on all the backlog tasks. With more experience now in scrum and with a better understanding of the software needed in the project, the backlog was more detailed and itemised than in our first sprint. A new template was also developed and used to capture our daily stand-ups.

Communication was more regular in Sprint 2 as all team members were in close proximity. This allowed us to plan and execute all deliverables from the sprint in a timely manner.

2. *Site Development*

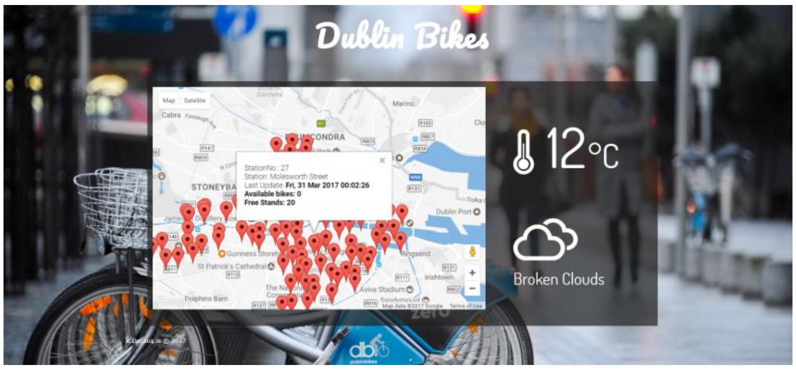


Figure 3.1. Prototype delivered at end of Sprint 2.

By the end of Sprint 2, the application prototype was developed. The site was loosely based on the UI design developed from Sprint 1. JavaScript, JQuery, HTML, CSS, Flask and Python were the main languages used to develop the site beyond its previous iteration. Few site features were outstanding at this stage: 1) charts, 2) search function, 3) feature to easily display bike usage, and 4) finalising site design and hosting.

3. *Data Collection and Database Management*

The prototype displayed the live Dublin Bike and weather data from JC Decaux and OpenWeatherMap via our database. Queries were developed in MySQL workbench and migrated to our flask application for live deployment.

Delivered Materials and Tasks Completed

What went well during the sprint cycle

- **Planning and task management.** Having conducted Sprint 1 with limited communication, a large focus of Sprint 2 was placed on planning and ensuring that sprint goals were met in a timely manner. Emphasis was placed on task management to ensure there was no last-minute rush to finish the project during Sprint 3. This was particularly important considering the increased workload with other modules during this period. The initial sprint 2 meeting was key to this improved planning. The backlog was updated and reformed and tasks were organised and prioritised. Once a task was completed, another item on the backlog was taken up by a team member. No team member was pigeon holed into a specific area but instead we collaborated across each thematic area.

- **Tasks divided up based on backlog and prioritised items**- SQL queries, weather, charts, markers on map. All geared towards finalising a prototype for the end of sprint 2.

- **Communication.** With a full team in presence, regular stand-ups held and increased interaction, team members could grasp each other's work, and maintain a focus on the backlog items.

- **Simple and Basic Design.** Instead of overreaching for a complex system and under delivering, we set out for a simple site design. The site features were developed from our user stories and although there were still some stretch cards, our priorities were set and these tasks were achieved and delivered.

- **Group documentation improved**. Given our limited interaction in Sprint 1, our backlogs were handled haphazardly. For sprint 2, we were organised in our approach to updating the daily stand-ups, updating the backlog, and being up to date on our learning journals.

- **Google Drive for documentation.** To address the merging documentation challenge faced in Sprint 1, we migrated our documentation to Google Drive. This proved useful and there we no merging complications experienced thereafter.

- **Caching data increased speed.** Google maps was slow in loading our data onto the map due to the large query being processed. Caching the data helped improving the performance here.

- **Improved project structure.** Research conducted during Sprint 1 was helpful in organising our code for sprint 2. JavaScript, Python, HTML, CSS files were all separated out into separate components for easier understanding. It's also easier then to understand and troubleshoot issues based on the source.

What went wrong during the sprint cycle?

- **Issues gaining Access to AWS.** MYSQL issues with workbench, getting access to the AWS. Weary about running queries on live data. Created a dump using WINSSHFS. Designed queries then ran connection to AWS using workbench to run live queries. This worked and was imported to flask.

- **Scraping stopping and starting**. With no unique ID, our scraper stopped a couple of times because of some duplicates in the stations. We had the last-update and station as the unique id, but if a station doesn't change the dataset sends back the previous data. So there are duplicates of the primary key. As this is not possible, the scraper stopped. We removed the primary key and ran the scraper again and this resolved the issue. A similar issue also happened for the weather data.

- **Getting the SQL dump.** Exports not as a database like other RDMS like MS Access, instead creates a dump with files. Not as portable. Required each of us to create a separate database locally and then try our queries. Time consuming. Eventually decided to just run using like data once piloted with the

- **Not querying most up to date data.** Because of the issue above, we also ran into issues retrieving the 101 stations most up to date data. Our date time format is in string format and when we sorted our dates, if a station didn't send change, we wouldn't get the 101 stations. We needed to organise in descending order and then query this by limiting to only the 101 stations. We did this by creating an extra column and giving each column a unique id in INT format. This resolved our issue. We also did this for weather in anticipation of any similar issues.

- Working on trello boards and completing time estimations required more regular update and input.

What could we do differently to improve?

- **Troubleshooting and debugging affecting task completion**. Unanticipated delays were experienced on many items throughout the sprint which delayed our estimated delivery for the task. There was a need to work more closely together on these issues and pair program if necessary to help with debugging and improve continuous integration. More communication can help us resolve these issues.

- **Poor continuous integration.** Merging our programming continually taken too long. Instead of using git to resolve our integration. These delays stem from our lack of understanding on how continuous integration works when merging in Git. Going forward we need to increase our git commits to ensure we are integrating quicker and more regularly.

- **Time estimation could be improved.** In some cases, many tasks took much longer than expected. In some cases, simply displaying the weather or the icons on the map took long due to our inexperience with JavaScript and jQuery. We therefore need to factor this into our planning processes.

Workload

Backlog	Assigned to	Estimated	Actual
Parsing of dynamic weather data into database	Darragh	4	4
Merge all Sprint 2 code	All	1	1
Write up individual learning journals	All	3	4
Summary documentation and log of meetings and stand ups	Darragh	2	2
Research data analytic features	Cara	2	2
Sprint 2 Review/Retrospective conducted	All	3	3
Populate map with bike stations on load	Cara & Tao	3	6
Flask integrated with basic HTML/CSS	Tao	3	3
Displaying weather data on map	Darragh	3	4
Develop HTML/CSS prototype	All	2	3
Create multilevel access to AWS for continuous development	All	1	4
Python script to connect to SQL database	Darragh	2	2
Create Relational Database	Darragh & Tao	2	2
Develop SQL Queries and flask application for google map and weather	Darragh	2	2
Merge database/AWS work and configure continuous web scraping framework	Darragh & Tao	3	3
Install and web scraping framework on AWS (including debugging)	Tao	3	6
UI Design in Wireframe	Cara	2	2
Scrum 2: Meetings	All	8	13
Search engine optimisation	Tao	3	5
Unit Testing	Tao	2	0
	Total	**54**	**71**

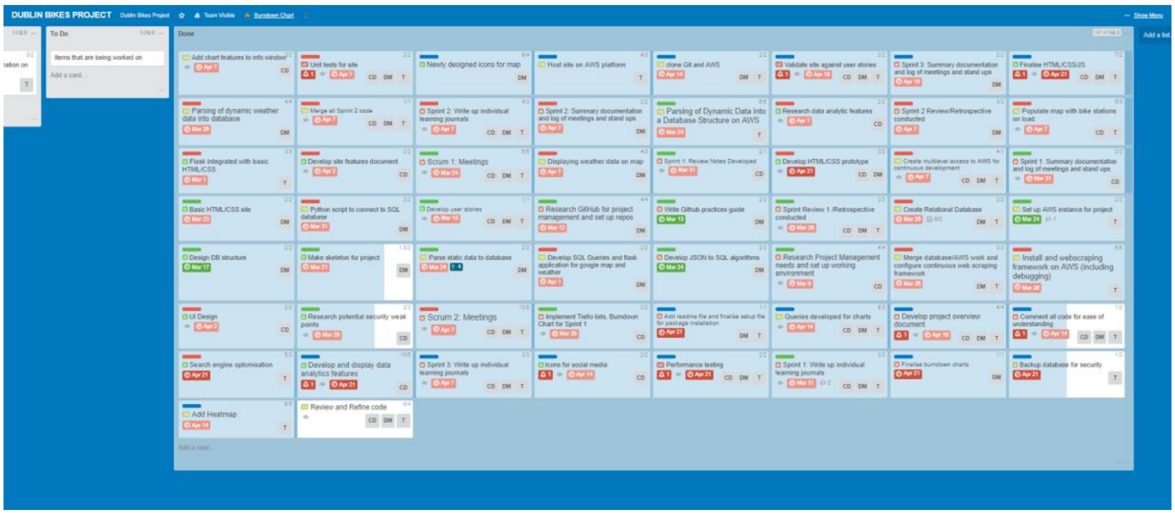
Sprint 3

Dates: 14-23 April 2017
Scrum Master: Tao Li

SPRINT REVIEW

1. *Management and Communication*

Sprint 3 was the final sprint of the project. Tasks were allocated for completion during the sprint with three days of preparation time allocated to finalise the project.


DUBLIN BIKES PROJECT Dublin Bikes Project Team Visible Burndown Chart Show Menu

To Do
Items that are being worked on
Add a card...

Done
Add chart features to info window — Apr 7 — CD
Unit tests for site — 1 — Apr 7 — CD DM T
Newly designed icons for map — DM
Host site on AWS platform — T
clone Git and AWS — Apr 14 — DM T
Validate site against user stories — 1 — Apr 19 — CD DM T
Sprint 3: Summary documentation and log of meetings and stand ups — Apr 19 — DM
Finalise HTML/CSS/JS — 1 — Apr 21 — CD DM T
Parsing of dynamic weather data into database — Mar 28 — DM
Merge all Sprint 2 code — Apr 7 — CD DM T
Sprint 2: Write up individual learning journals — Apr 7 — CD DM T
Sprint 2: Summary documentation and log of meetings and stand ups — Apr 7 — DM
Parsing of Dynamic Data into a Database Structure on AWS — Mar 24 — T
Research data analytic features — Apr 7 — CD
Sprint 2 Review/Retrospective conducted — Apr 7 — DM
Populate map with bike stations on load. — Apr 7 — CD T
Flask integrated with basic HTML/CSS — Mar 1 — T
Develop site features document — Apr 2 — CD
Scrum 1: Meetings — Mar 24 — CD DM T
Displaying weather data on map — Apr 7 — DM
Sprint 1: Review Notes Developed — Mar 31 — CD
Develop HTML/CSS prototype — Apr 21 — CD DM
Create multilevel access to AWS for continuous development — Apr 7 — CD DM T
Sprint 1: Summary documentation and log of meetings and stand ups — Mar 31 — CD
Basic HTML/CSS site — Mar 23 — DM
Python script to connect to SQL database — Mar 31 — DM
Develop user stories — Mar 10 — CD DM T
Research GitHub for project management and set up repos — Mar 12 — DM
Write Github practices guide — Mar 13 — DM
Sprint Review 1 /Retrospective conducted — Mar 28 — CD DM T
Create Relational Database — Mar 20 — 0/2 — DM T
Set up AWS instance for project — Mar 24 — 1 — T
Design DB structure — Mar 17 — DM
Make skeleton for project — Mar 21 — DM
Parse static data to database — Mar 24 — 4 — DM
Develop SQL Queries and flask application for google map and weather — Apr 1 — DM
Develop JSON to SQL algorithms — Mar 24 — DM
Research Project Management needs and set up working environment — Mar 9 — CD
Merge database/AWS work and configure continuous web scraping framework — Mar 20 — DM T
Install and webscraping framework on AWS (including debugging) — Mar 28 — T
UI Design — Apr 2 — CD
Research potential security weak points — Mar 29 — CD
Scrum 2: Meetings — Apr 7 — CD DM T
Implement Trello lists, Burndown Chart for Sprint 1 — Mar 28 — CD
Add readme file and finalise setup file for package installation — Apr 21 — DM T
Queries developed for charts — Apr 14 — CD DM T
Develop project overview document — 1 — Apr 18 — CD DM T
Comment all code for ease of understanding — 1 — Apr 14 — CD DM T
Search engine optomisation — Apr 21 — T
Develop and display data analytics features — 1 — Apr 21 — CD
Sprint 3: Write up individual learning journals — Apr 7 — CD DM T
Icons for social media — 1 — Apr 14 — CD
Performance testing — 1 — Apr 21 — CD DM T
Sprint 1: Write up individual learning journals — Mar 31 — 2 — CD DM T
Finalise burndown charts — Apr 21 — DM
Backup database for security — Apr 21 — T
Add Heatmap — Apr 14 — T
Review and Refine code — CD DM T
Add a card...

Add a list...

Figure 4.1. Finalised Trello board.

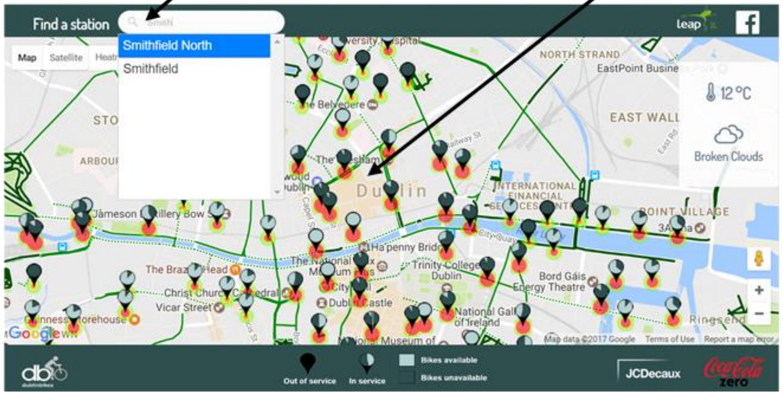
2. *Site Development*

The final product was tested against the user stories identified at the beginning of the project. All site features were met apart from the stretch card: displaying geolocation. With multiple competing deadlines from other modules, a decision was made to drop this feature from our final deliverable.

Test against user stories

User Story 2: I want to have a search function for a station, so I can quickly access my regular station.
Test: Success
Details: Autocomplete search function included. Brings user to new page with only selected station. User can return to main page by selecting the home page.

User Story 3: I want to see available stations from adequate glance, so that I can see where I should get a bike
Test: Success
Details: Small pie charts display the occupancy once site is loaded. Additional layer of heatmap can also display usage giving the user a preference over methods.



User Story 1:
I want to see a google map of Dublin, so I can click on a station for bike availability and address information.
Test: Success

User Story 4: I want to view weather info, so I can decide whether or not I want to get the bus.
Test: Success
Details: Live temperature and description of weather included on map.

Figure 4.2 Tests against user stories.

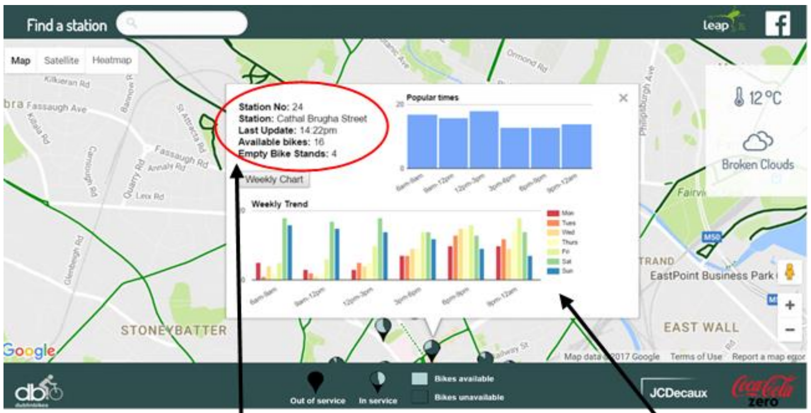

Station No: 24
Station: Cathal Brugha Street
Last Update: 14.22pm
Available bikes: 16
Empty Bike Stands: 4

Figure 4.3 Tests against user stories cont.

User Story 5: I want to see the average occupancy time around my search, so I can plan my imminent trip
Test: Success
Details: Info window displays the exact time of my search. Details around average occupancy are addressed in user story 6

User Story 6: I want to see daily and weekly averages, so I can plan a future trip
Test: Success
Details: Info window displays the daily chart. With the click of a button, weekly data is displayed for the station.

User Story 7: I want to access the website on my smartphone and provide my locational details, so I can see the availability of bike stands in my proximity.
Test: Failed
Details: Stretch card. Unable to complete within timeline.

3. *Data Collection and Database Management*

The end of sprint met all data requirements of Sprint 3. Graphs display daily averages and weekly averages based on the station selected. Querying our data performed poorly in tests and as a result, we decided that daily graphs would be displayed immediately as their response times were adequate (2 seconds to load), but weekly graphs would be displayed from a button click (11 seconds to load). The graph would load in the background and by the time the user clicks for weekly information, the data was loaded.

Delivered Materials and Tasks Completed

What went well during the sprint cycle

- **Improved team collaboration.** A weakness identified in Sprint 2 was our determination to persist in resolving a problem instead of collaborating openly to debug and problem solve. In sprint 3, the team was open in areas of difficulty and we collaborated to debug code.

- **Deadlines achieved.** All major deadlines were achieved during the sprint.

- **Documentation up to date.** Often on projects, there can be a major rush to complete documentation prior to submission. Our team had kept up to date with all documentation throughout this sprint and as a result we didn't face this challenge.

- **Frequent communication.** Taking lessons from sprint 1, our team ensured that there was frequent dialogue during the final sprint. This included daily stand-ups and working together on project work in college instead of working in isolation. This allowed us to maintain our focus and quickly find solutions to collective issues.

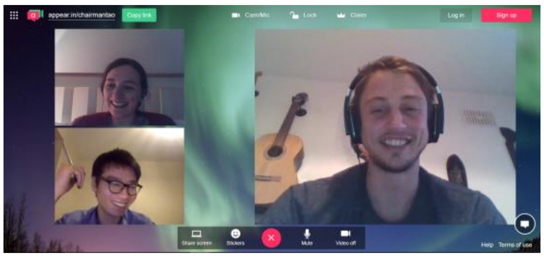


Figure 4.3 Team meeting on appear.in

What went wrong during the sprint cycle?

- **Database design affected final deliverable.** The performance of our graphs is affected by our inability to conduct calculations on the datatime column in SQL.

- **Troubleshooting and debugging affecting task completion**. This problem persisted from sprint 2 despite more collaboration. We could resolve some aspects together as a group but for others, hours were spent debugging simple tasks.

- **Time estimations didn't improve.** At the end of sprint 2, we believed there were only few tasks left for completion. The main task left to complete was the charts. We believed this would take approximately 5 hours to complete. However, we ran into many debugging problems in dealing with the date time and pandas and as a result took twice as long as expected. Again, our time estimations failed to account for inexperience. We also failed to account for other deadlines appropriately. The final sprint was at the end of the semester. With many other deadlines to meet, we failed to leave enough time to iteratively refine our solution, particularly querying the data. In the end, we submitted our first and final solution.

- **Continuous integration.** Merging code continued to be a challenge for us as a team as we didn't use git appropriately to integrate code.

- **Reviewing and refining code.** As our time estimations were off, our final code submitted was not clean. The structure of our python code required segmentation to reduce the amount of repetition. Many functions were overloaded and not reusable. Our search page (index1) could also be improved upon instead of hardcoding the new page.

What could we do differently to improve?

- **More effort into continuous integration.** As this issue continued for the entire project, it's clear that more investment was needed in git to ensure that merging of code was more efficient. This would have reduced our time on task and may have resolved many debugging challenges we faced.

- **Alter structure of the database design for more efficient querying of data**. As the performance of our data querying was poor, more effort should have been placed on the database design. Although in hindsight our date time should never have been string, we could have created a new column to translate the string column into a date time format for easier querying.

- **Allocate more time for refinement of deliverable.** This was perhaps the most important mishap of the final sprint. Our task estimations were assuming we would have no major delays which inevitably arose.

Workload

Backlog	Assigned to	Estimated	Actual
Newly designed icons for map	Darragh	4	6
Queries developed for charts	All	3	8
Sprint 3: Write up individual learning journals	All	3	3
Finalise burndown charts	Darragh	1	2
Backup database for security	Tao	2	1
Add Heatmap	Tao	5	6
Icons for social media	Cara	2	2
Add readme file and finalise setup file for package installation	Tao	1	1
Comment all code for ease of understanding	Tao	2	2
Merge Git and AWS data	Tao and Darragh	1	1
Validate site against user stories	Darragh	1	1
Develop project overview document	Darragh	4	4
Sprint 3: Summary documentation and log of meetings and stand ups	Tao	2	2
Review and refine code	All	3	0
Finalise HTML/CSS/JS (6/2)	Darragh	2	7
Display geospatial information on map (0/2)	Tao	2	0
Performance testing	Tao	2	2
Host site on AWS platform	Tao	2	2
Develop and display data analytics features	Cara	5	10
Unit Testing	Tao	2	2
	Total	**57**	**58**

Conclusion

Overall, the project was an extremely beneficial learning experience. Not only did it introduce a wide range of programming languages, it also tied together our learning from other modules thereby giving us a better understanding of programming in the workplace. In terms of the project management, following the Scrum methodology was a beneficial learning experience in that it showed us how to break up software projects into short and achievable deliverables. For each sprint, team members were continuously working on the most important tasks required for the Sprint instead of getting ahead of themselves. Although our final deliverable could be improved upon, we have learned from our experience and become more self-sufficient programmers as a result.

Appendix A: User Stories and Design

USER STORIES

User stories describe high-level demands from users of the final product. They provide context for the functionality of the features. User stories detail product requirements. These large scale, long-term requirements are broken up into tasks and subtasks, given an estimated timescale and are well understood.

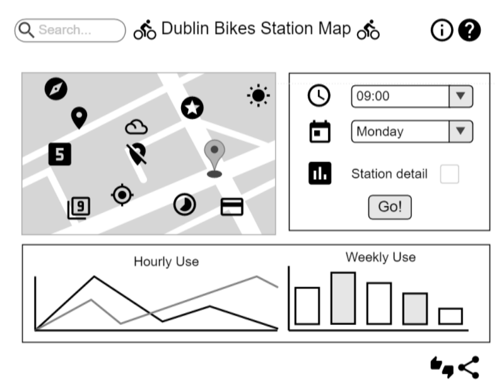
Who will use a feature(s), what do they need from the product, how will the feature provide benefits/value?

As a <type of user>, I want <some goal> so that <some reason>

- As a basic user, I want to see a google map of Dublin So I can click on a station for bike availability and address information
- As a daily user I want to have a search function for a station So I can quickly access my regular station.
- As a daily user, I want to see the average occupancy time around my search So I can plan my imminent trip
- As a basic user, I want to see available stations from adequate glance So that I can see where I should get a bike
- [Stretch Card] As a basic user, I want to access the website on my smartphone and provide my locational details So I can see the availability of bike stands in my proximity.
- As a commuter, I want to view weather info So I can decide whether or not I want to get the bus.
- As a future user, I want to see daily and weekly averages So I can plan a future trip

Examples of user stories:
https://www.mountaingoatsoftware.com/agile/scrum/scrum-tools/product-backlog/example
https://www.scrumalliance.org/community/articles/2011/august/5-common-mistakes-we-make-writing-user-stories
designing backlog slideshare: https://www.slideshare.net/rpannone/creating-a-product-backlog

Initial Project Design



Appendix B: Database Design

Purpose

This document outlines the structure of the SQL database for Assignment 4 of the MSc. Software Engineering Module. It describes the static and dynamic data imported in JSON format from JCDecaux Developer and suggests a database structure for the project.

Data Structure

Static Data

Feature	Data Type	Description	Comment
number	**Int**	Number of the station. This is NOT an id, thus it is unique only inside a contract.	Without multiple contracts (i.e. multiple cities), this can become the unique identifier for the stations.
contract_name	**Varchar**	Name of the contract of the station	Constant column. Only one city being analysed. Should be dropped.
name	**Varchar**	Name of the station	Looking at data, both the name and address have the same values.
address	**Varchar**	Address of the station. As it is raw data, sometimes it will be more of a comment than an address	Same as name
position	**Float**	position of the station in WGS84 format	Data is split into two: 1) longitude and 2) latitude.
banking	**Boolean**	Indicates whether this station has a payment terminal	Not included in Dublin static data. Not required.
bonus	**Boolean**	Indicates whether this is a bonus station	Not included in Dublin static data. Not required.

Dynamic Data

Feature	Data Type	Description	Comment
status	**Varchar**	Indicates whether this station is CLOSED or OPEN	-
bike_stands	**Int**	The number of operational bike stands at this station	-
available_bike_stands	**Int**	The number of available bike stands at this station	-

available_bikes	**Int**	The number of available and operational bikes at this station	-
last_update	**Data/Time**	Timestamp indicating the last update time in milliseconds since Epoch	Should be in a format that can be used in line with the open weather application.

With each instance of real-time/dynamic data, the static data is again replicated. See example below:
Example of Real-time data

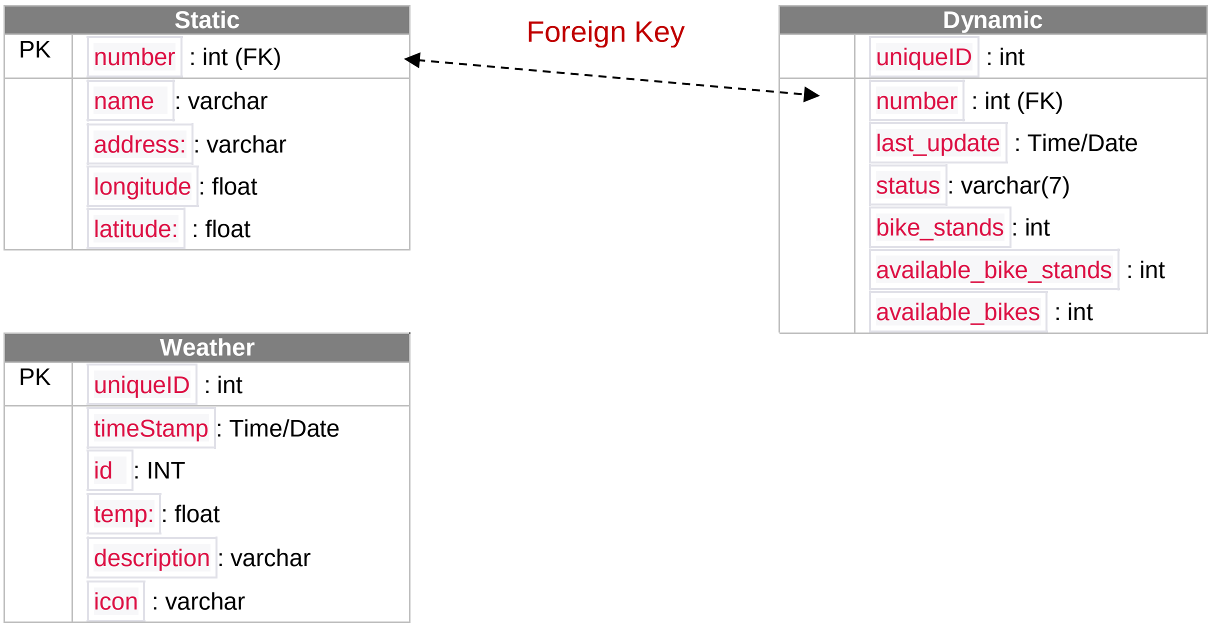
```
{
  "number": 123,
  "contract_name" : "Paris",
  "name": "stations name",
  "address": "address of the station",
  "position": {
    "lat": 48.862993,
    "lng": 2.344294
  },
  "banking": true,
  "bonus": false,
  "status": "OPEN",
  "bike_stands": 20,
  "available_bike_stands": 15,
  "available_bikes": 5,
  "last_update": <timestamp>
}
```

Proposed Design

Three simple tables can be created for the database for 1) static, 2) dynamic, and 3) weather. Since the dataset is only dealing with Dublin as a contract, all numbers are unique. The `number` should therefore be the primary key for static data. The only constant column: `contract_name` will also be dropped as we're only dealing with Dublin city data.

For dynamic data, we're dealing with multiple weeks of data. As such, the primary key for the dynamic data will be the `number` and the `last_update` column in conjunction. Both tables will link via the `number` as its foreign key. To reduce replication and address complications that arise from data repetition, the tables will be normalised. All data present in the dynamic data will not be scraped, except for the `number` which is required for the primary key.

See below the proposed design in an entity relationship diagram:



Static	
PK	number : int (FK)
	name : varchar
	address: : varchar
	longitude : float
	latitude: : float

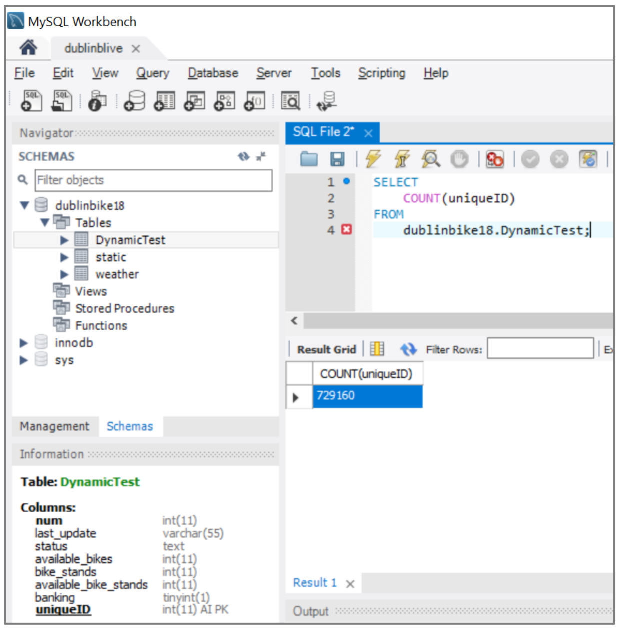
Dynamic	
	uniqueID : int
	number : int (FK)
	last_update : Time/Date
	status : varchar(7)
	bike_stands : int
	available_bike_stands : int
	available_bikes : int

Weather	
PK	uniqueID : int
	timeStamp : Time/Date
	id : INT
	temp: : float
	description : varchar
	icon : varchar

ER Diagram source: http://www.datanamic.com/dezign/erdiagramtool.html

Final Database

Dynamic Data from Dublin Bikes



```
SELECT
    COUNT(uniqueID)
FROM
    dublinbike18.DynamicTest;
```

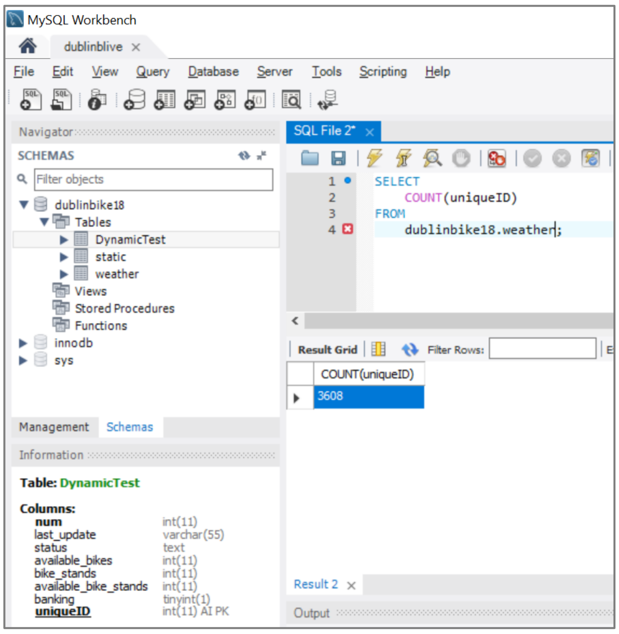
```
COUNT(uniqueID)
729160
```

Table: DynamicTest

Columns:

```
num                    int(11)
last_update            varchar(55)
status                 text
available_bikes        int(11)
bike_stands            int(11)
available_bike_stands  int(11)
banking                tinyint(1)
uniqueID               int(11) AI PK
```

Dynamic Data from OpenWeatherMap



```
SELECT
    COUNT(uniqueID)
FROM
    dublinbike18.weather;
```

```
COUNT(uniqueID)
3608
```

Table: DynamicTest

Columns:

```
num                    int(11)
last_update            varchar(55)
status                 text
available_bikes        int(11)
bike_stands            int(11)
available_bike_stands  int(11)
banking                tinyint(1)
uniqueID               int(11) AI PK
```

Appendix C: Project Guides

1. Access to Open Weather Map

How to get accurate API response

1. Do not send requests more than 1 time per 10 minutes from one device/one API key. Normally the weather is not changing so frequently.
2. Use the name of the server as **api.openweathermap.org**. Please never use the IP address of the server.
3. Call API **by city ID** instead of city name, city coordinates or zip code. In this case you get precise respond exactly for your city. **Dublin:** {2964574, Dublin, 53.343990, 6.267190, IE)
4. Free accounts are limited in availability so may not respond.
5. Store requested data.

Access limitation

If account exceeds the limits, then a notification about limits exceeding is sent. If it repeats again, then the account is blocked for an hour. Therefore, the lock period is increased by one hour until 4 hours block sets. When blocking repeats the fifth time, then the lock period lasts 24 hours. This rule is cycled. Please be careful with the number of API calls you complete.

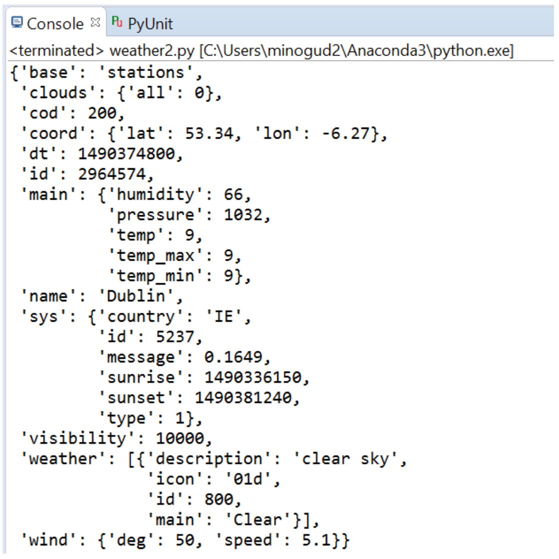
How to Request Dynamic Data

```
import requests
import json
import pprint

url='http://api.openweathermap.org/data/2.5/weather?id=2964574&APPID=33e340fbba76a4645e
26160abb37f014&units=metric'
```

- **From the Above:** url='http://api.openweathermap.org/data/2.5/weather?id=2964574' - this is the url to request. Then you need to add your API key for Open Weather App.
- **Add URL key:** url='http://api.openweathermap.org/data/2.5/weather?id=2964574&APPID=33e340fbba76a4645e26160abb37f014&units=metric'
- **Metric added at end for specialised units.**

```
data = requests.get(url).json()

pprint.pprint(data)
```

- **Pretty Print will print out the dictionary obtained in vertical format vs horizontal list**



```
<terminated> weather2.py [C:\Users\minogud2\Anaconda3\python.exe]
{'base': 'stations',
 'clouds': {'all': 0},
 'cod': 200,
 'coord': {'lat': 53.34, 'lon': -6.27},
 'dt': 1490374800,
 'id': 2964574,
 'main': {'humidity': 66,
          'pressure': 1032,
          'temp': 9,
          'temp_max': 9,
          'temp_min': 9},
 'name': 'Dublin',
 'sys': {'country': 'IE',
         'id': 5237,
         'message': 0.1649,
         'sunrise': 1490336150,
         'sunset': 1490381240,
         'type': 1},
 'visibility': 10000,
 'weather': [{'description': 'clear sky',
              'icon': '01d',
              'id': 800,
              'main': 'Clear'}],
 'wind': {'deg': 50, 'speed': 5.1}}
```

No timestamp included.

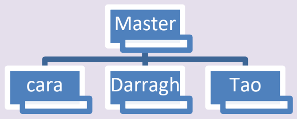
For more info:
https://openweathermap.org/current

1. Could request data every hour from open weather?

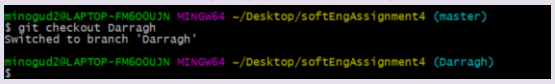
```
{'base': 'stations',
 'clouds': {'all': 0},
 'cod': 200,
 'coord': {'lat': 53.34, 'lon': -6.27},
 'dt': 1490374800,
 'id': 2964574,
 'main': {'humidity': 66,
          'pressure': 1032,
          'temp': 9,
          'temp_max': 9,
          'temp_min': 9},
 'name': 'Dublin',
 'sys': {'country': 'IE',
         'id': 5237,
         'message': 0.1649,
         'sunrise': 1490336150,
         'sunset': 1490381240,
         'type': 1},
 'visibility': 10000,
 'weather': [{'description': 'clear sky',
              'icon': '01d',
              'id': 800,
              'main': 'Clear'}],
 'wind': {'deg': 50, 'speed': 5.1}}
[9, 1032, 9, 9, 66, 5.1, 50, 0, 800, 1490336150, 1490381240]
```

2. Use of GitHub for Assignment

1. **Accept link** to git-hub repository from sender.
2. Open git-bash on windows or bash on mac. **Move to directory** where you would like to store the git repository. E.g. cd Desktop
3. **Clone the repository**
    a. git clone https://github.com/minogud2/softEngAssignment4
    b. If problems encountered, make sure the SSH key is established before cloning repository.
4. Once established, **cd into the new directory**
    a. cd softEngAssignment4
5. **Check out the existing branches**
    a. git branch
    b. A list of four branches should appear(see tree)- If you still see the previous branches I set up (base and working tree), git remote prune origin. If that doesn't work then forcibly delete them by doing the following: git branch -D base and git branch -D WorkingTree



6. **Log into your branch**
    a. git checkout Darragh
    b. Bash will then display your change. See below.



```
minogud2@LAPTOP-FM6OOUJN MINGW64 ~/Desktop/softEngAssignment4 (master)
$ git checkout Darragh
Switched to branch 'Darragh'

minogud2@LAPTOP-FM6OOUJN MINGW64 ~/Desktop/softEngAssignment4 (Darragh)
$
```

7. **Get up to date version of master repository**
    a. git pull master origin
    b. Do this only, if you want the most up to date version of the repository. Ensure that you have not worked on any documents in your repository before doing this. Otherwise, you will pull from the master and overwrite your version of the repository.
8. **Begin working.**
    a. Create folder in repository and use this to develop your code. We will then discuss which files need to be pushed to the main skeleton folders.
    b. git add .– adds the files and stations them for commit.
    c. git commit -m "This is my first commit" - Ensure all commits are relative to what you are doing. Everyone shouldn't be writing- version1 commit or initial commit. Instead- "Json file parsed into database. Committing database- version 1".
    d. git status - check that the files have been updated.
    e. git push origin master- or git push Pushes your completed work to the master for merging. If the push doesn't work. Complete the following:
        i. git push -u origin master-the first time that you push that branch. You only need to do it once, and that sets up the association between your branch and the one at origin in the same way as git branch --set-upstream does.
9. **Master will merge changes.**
    a. As git administrator for the project, I will handle the merges for the time being. The changes will be merged by doing:
    b. git checkout master
    c. git merge Darragh

Other useful information

Adding branches

- git branch NewBranchName

Deleting Branches

- git branch -d NewBranchName- deletes if there are no merge conflicts.
- git branch -D NewBranchName- forcibly deletes irrespective of merge issues.

Undo commits

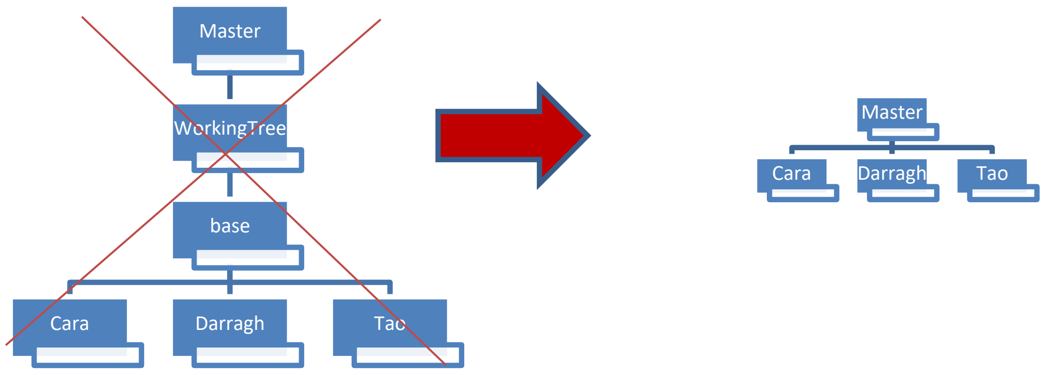
- if you accidently delete your files when merging, as I have done while making this document. It's possible to get back your info using the following:
- git reset
- git stash
- git add .
- git commit -m "undoing delete"
- git push

Merging upstream changes into your local repository

- git pull <remote>- does a git fetch and a git merge all in one**.** Brings the local branch up to date with the remote branches/master. It automatically merges changes without reviewing them first. May run into conflicts if branches not closely managed.

Tutorials:
For all aspects of Git:
https://www.atlassian.com/git/tutorials
For undoing a delete:
http://stackoverflow.com/questions/927358/how-to-undo-last-commits-in-git

Deductive Process



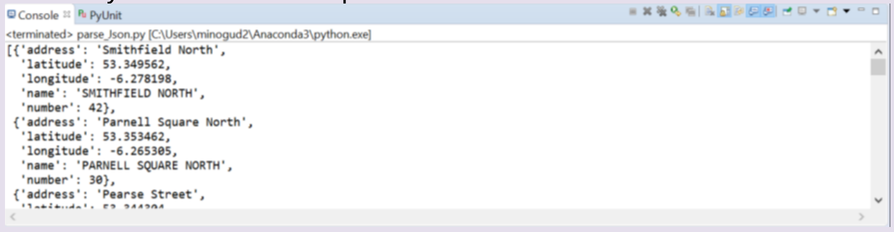
Based on guidance from tutors and lecturers, we decided against the model of the left in favour of the model on the right for the sake of simplicity.

3. JSON to SQL Approach

1. **Install sqlite in virtual environment**
    a. conda install sqlite
2. Download static file into repository with parse_Json.py file
3. **Open file:**

```
with open(json_info) as jsonfile:
    data = json.load(jsonfile)
```

This creates a list with a dictionary inside. See below. To get into the list, you must first index the list. E.g. print(data[0]) will produce the first dictionary inside. In the example below, this would be only Smithfield North. print(data[1]) would produce the entire dictionary then for Parnell Square North. This includes all fields between the {} brackets.



```
<terminated> parse_Json.py [C:\Users\minogud2\Anaconda3\python.exe]
[{'address': 'Smithfield North',
  'latitude': 53.349562,
  'longitude': -6.278198,
  'name': 'SMITHFIELD NORTH',
  'number': 42},
 {'address': 'Parnell Square North',
  'latitude': 53.353462,
  'longitude': -6.265305,
  'name': 'PARNELL SQUARE NORTH',
  'number': 30},
 {'address': 'Pearse Street',
```

4. **Convert data to sql database**
    a. Need to convert the key values into variables: i.e. address, latitude, etc. So first create empty list variables with the appropriate names.
    b. Iterate through to the end of the json_file and append the empty lists with the indexed values.

```
i = 0
while i < len(data):
    number.append(data[i]["number"])
    name.append(data[i]["name"])
    address.append(data[i]["address"])
    latitude.append(data[i]["latitude"])
    longitude.append(data[i]["longitude"])
    i += 1
```

    c. **Create a new database.**

```
# create database to parse data into.
con = lt.connect('db_Dataset.db')
```

    d. **Create table for database.** (REAL is float in sqlite)

```
with con:
        cur = con.cursor()
                    cur.execute("CREATE TABLE Static(num INT,
            Sname TEXT, addr TEXT, lat REAL, long REAL)")
```

    e. **Loop through list variables and populate the new database**

```
for i in range(0, len(data)):
            cur.execute("INSERT INTO Static (num, Sname, addr,
lat, long) VALUES(?,?,?,?,?)",
                    (number[i], name[i], address[i], latitude[i],
longitude[i]))
                         con.commit()
```

    f. **Close database connection and cursor**

```
                cur.close()
                con.close()
```

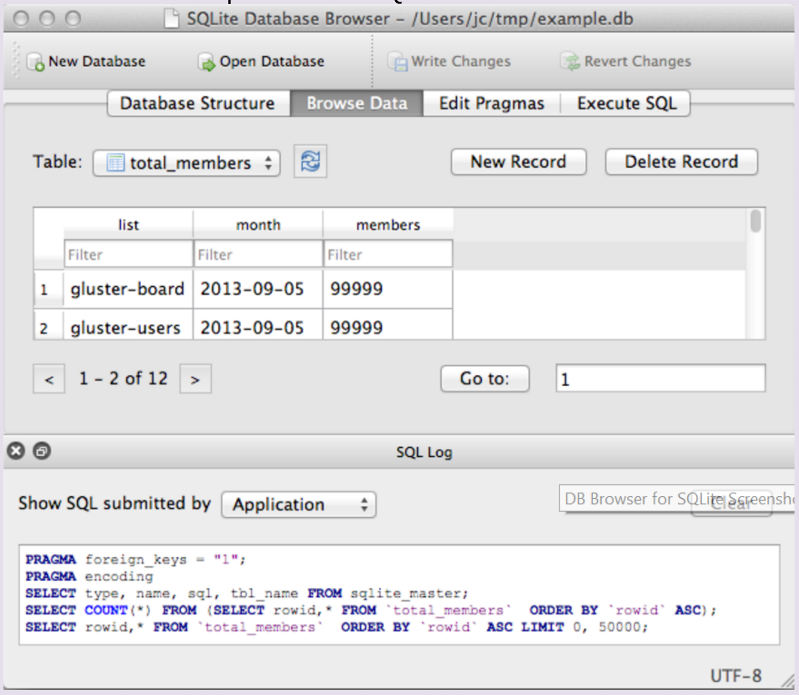
g. Print out dataset.

- Connect to database, load all the data.
- Fetch all the data loaded
- Print the data
- Close the connections.

```
con = lt.connect('db_Dataset.db')
cur = con.cursor()
cur.execute('SELECT * FROM Static')
data2 = cur.fetchall()
pprint(data2)
cur.close()
con.close()
```

Other useful information

Download SQLITE Browser
http://sqlitebrowser.org/
DB Browser for SQLite is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite.



	list	month	members
	Filter	Filter	Filter
1	gluster-board	2013-09-05	99999
2	gluster-users	2013-09-05	99999

```
PRAGMA foreign_keys = "1";
PRAGMA encoding
SELECT type, name, sql, tbl_name FROM sqlite_master;
SELECT COUNT(*) FROM (SELECT rowid,* FROM `total_members`  ORDER BY `rowid` ASC);
SELECT rowid,* FROM `total_members`  ORDER BY `rowid` ASC LIMIT 0, 50000;
```

Tutorial for SQLITE

- Inserting variables to database table - SQLite3 with Python 3 part
  https://www.youtube.com/watch?v=qfGu0fBfNBs

- Read from (SELECT) Database table - SQLite3 with Python 3 part 3
  https://www.youtube.com/watch?v=NCc5r7Wr7gg

4. Skeleton Structure for Project

Source: Shaw, Zed (2014), Learn Python the Hard Way:
http://www.souravsengupta.com/int2pro2014/python/LPTHW.pdf

```
Master/
     src/
         __init__.py # to initialise directory as module
     bin/ # This is just a standard place to put scripts that are run on the com
mand line, not a place to put modules.
     docs/
     static/
      /css # stylesheets for website
      /images # background images for website
      /js # any javascript used
     setup.py # to install project later and provide basic info.
     tests/
         NAME_tests.py
         __init__.py # to initialise directory as module
```

Appendix C: Daily Stand-ups

Sprint 1

Date	Team Member	What did you do yesterday?	What are you going to do today?	What impediments did you face?
Fri 10 March 2017	**Cara**	Pre-sprint meeting, general research.	Write project summary/plan doc based on assignment	Last day before two-week break. Limited time for meeting. We had difficulty estimating work targets for Sprint 1 before we split up.
	Darragh	Pre-sprint meeting held on March 6.	- Start Sprint - Develop basic user stories and backlog - Request guidance on git hub repository development from lecturer	- Lack of understanding on all tasks needed in project. - Plan requires consistent updating as our project understanding increases.
	Tao	General research.	Set up general working environment	Trip to China. Won't be able to meet up and communication might be a problem
Sat 11 March 2017	**Cara**	Completed yesterday's tasks	Review Sprint management tools (Trello, Smartsheet, Excel).	N/A
	Darragh	- Schedule and backlog developed for team members.	N/A	N/A
	Tao	Completed yesterday's tasks	learn Sprint management tools (Trello, Smartsheet, Excel).	N/A
Sun 12 March 2017	**Cara**	Completed yesterday's tasks	Research project components (Flask, Ec2, Maps API) to help outline a backlog.	N/A
	Darragh	N/A	- Research GitHub for project mgmt & repo setup	N/A
	Tao	N/A	N/A	N/A
Mon 13 March 2017	**Cara**	Completed yesterday's tasks	Draft Sprint 1 & product backlog, research flask & write simple flask hello world app.	Given the scope of the tasks required it was difficult to accurately assess the time required, difficulty of tasks.
	Darragh	- Scheduled activity completed.	- Write up git practices guide	N/A
	Tao	N/A	N/A	N/A
Tue 14 March 2017	**Cara**	Completed yesterday's tasks	Collate meeting notes, populate trello board with drafted Sprint 1 tasks, assign tasks, estimate time required. Make burndown chart. Slack call with Darragh.	Difficulty in knowing who was working on specific tasks. Little time to discuss overall project plan of action before course break.
	Darragh	- Scheduled activity completed. - Feedback provided on Trello cards.	N/A	N/A
	Tao	N/A	N/A	N/A
Wed 15 March 2017	**Cara**	Completed yesterday's tasks	N/A	N/A
	Darragh	N/A	- Familiarise myself with flask and create small hello world app. - Develop basic database design document.	N/A

	Tao	N/A	N/A	N/A
Thu 16 March 2017	**Cara**	N/A	N/A	N/A
	Darragh	N/A	N/A	N/A
	Tao	N/A	N/A	N/A
Fri 17 March 2017	**Cara**	N/A	N/A	N/A
	Darragh	N/A	N/A	N/A
	Tao	N/A	N/A	N/A
Sat 18 March 2017	**Cara**	N/A	N/A	N/A
	Darragh	N/A	N/A	N/A
	Tao	N/A	More Research project components (Flask, Ec2, Maps API), learn basic fram work of Flask, trying to understand the structure of the project overall	N/A
Sun 19 March 2017	**Cara**	N/A	N/A	N/A
	Darragh	N/A	N/A	N/A
	Tao	Completed yesterday's tasks	Research on EC2 ans AWS	having problem to connect to ec2
Mon 20 March 2017	**Cara**	N/A	N/A	N/A
	Darragh	N/A	N/A	N/A
	Tao	Completed yesterday's tasks	Created AWS relational database , learn how to parse data locally, convert data from json to db	need more research on data converting
Tue 21 March 2017	**Cara**	N/A	Git management and organisation.	
	Darragh			
	Tao	Paritially Complete the task	finish convet static json into sql db, write code to build database on AWS	N/A
Wed 22 March 2017	**Cara**	Completed yesterday's tasks	Scrum with Darragh. Write python script for creating, adding to database, pulling in data from json file (followed tutorial and looked at D's script)	N/A
	Darragh	N/A	Sprint 2 meeting - Develop Json to SQL tutorial doc - Develop Skeleton project and upload to git.	- Unable to meet. Team member unavailable due to flight cancellation. Meeting changed to Sunday. Unable to complete some tasks which require all input. Anticipated scrum 1 activities will carry over to Scrum 2.
	Tao	Completed yesterday's tasks	travlling back to dublin	
Thu 23 March 2017	**Cara**	Completed yesterday's tasks	N/A	N/A
	Darragh	- Scheduled activities completed.	- Meeting with Cara	Difficult to go much further without the basic html.
	Tao	N/A	travlling back to dublin	N/A
	Cara	N/A	Write individual learning journal. Group video chat with entire team.	N/A

Fri 24 March 2017	**Darragh**	- Develop basic html with css for testing.	- Update database design document. - Parse static data into a sql database & develop JSON to SQL algorithms - Research and documented info on access to open weather map data. - Online meeting with Tao.	- Design may alter based on interaction with AWS. Communication.
	Tao	N/A	Write individual learning journal. Group video chat with entire team, writing on user story	N/A

Sprint 2

Date	Team Member	What did you do yesterday?	What are you going to do today?	What impediments did you face?
Sat 25 March 2017	**Cara**	Completed yesterday's tasks	Write userstories and breakdown of Sprint 2 work. Design mockup site view, list site features, functionality, UI. R+R prep.	Site features dev will be pushed to Sprint 3
	Darragh	- Scheduled activities completed	N/A	N/A
	Tao	Completed yesterday's tasks	basic code done, need to add Darragh's code (for weather) and some research on scheduling for the programm to run on ec2	nohup is not working on my ec2, always quit after i log out
Sun 26 March 2017	**Cara**	Completed yesterday's tasks	Review + Retrospective with team, Git push to master. Plan sprint 2. Discuss and develop Sprint 2 Backlog.	N/A
	Darragh	N/A	- Create relational Database. - Parse in weather information to RDB using API. - Sprint Review. - Developed user stories (all). - Developed more detailed backlog () - Merge static/dynamic data info with Tao to host on AWS.	- No access to AWS yet, need - Schedule work limited due to two tests on the 28th and 29th of March. - Lots of debugging problems when merging. Caused delays for Tao. - I was using SQLLITE and TAO was using MYSQL. Created integration issues during merge.
	Tao	Completed yesterday's tasks	debugging the code , fixded the problem from merging with the code, and officially launched our programme to start to collect data	debbugging is timeconsuming.
Mon 27 March 2017	**Cara**	Completed yesterday's tasks	Populate site map with JS station icons and info windows.	Had to hard code in data for now.
	Darragh	- Scheduled activities completed	- Update backlog based on Sunday's meeting.	- Debate on whether to continue or not with trello/smartsheet or excel.
	Tao	N/A	N/A	N/A
Tue 28 March 2017	**Cara**	Completed yesterday's tasks	Write .py, .html to display db data in basic Flask app.	DB Connection issue, values not displaying.
	Darragh	- Update backlog.	N/A	N/A
	Tao	N/A	N/A	N/A

Wed 29 March 2017	**Cara**	Completed yesterday's tasks	Set up AWS EC2. Have group meeting. Connect to Tao's DB	Can't connect to Tao's DB
	Darragh		- Meeting with Team - Allocate tasks for completion by Friday - Update trello with new backlog	Unable to gain access to AWS on windows.
	Tao	N/A	configuring AWS to let team member get access to db	have setting problems
Thu 30 March 2017	**Cara**	Group meeting. Got familiar with mysqld on EC2.	Connect to Tao's DB. Write doc on aws/mysql set up doc. Fixed Flask app DB connection from Tues.	Can't connect to Tao's DB
	Darragh	- Updated backlog on trello.	- Update daily stand up document. - Update meeting notes - Research on displaying weather to map.	- No access to aws, so will run it locally and then once up on AWS, we can merge. - "yum" command doesn't work on a Ubuntu server. Instead had to launch Linux server to follow Tao's documentation on accessing the dynamic data.
	Tao	Completed yesterday's tasks	Write code on flask and try to build up web site.	N/A
Fri 31 March 2017	**Cara**	Wrote doc aws/mysql set up doc. Displayed DB data in Flask app.	Group meeting, integrate map segment and db connection to working routes.py script. Research ChartsAPI.	
	Darragh			
	Tao	Not finished	get google map and marker up and running on the web, get test data in	
Sat 1 April 2017	**Cara**	N/A	N/A	N/A
	Darragh	N/A	N/A	N/A
	Tao	N/A	N/A	N/A
Sun 2 April 2017	**Cara**	- Looked at importing the google maps and google charts	Finalise the javascript for the dynamic data for the map. Made basic google chart, hard coded data into it. For charts, wrote sqlalchemy script for mapping/querying db.	Unable to gain access data using workbench. Need to instead just run from the dynamic data. Couldn't integrate map segm, db connection.
	Darragh	- created sql queries for parsing into map/weather - created sql dump and ran cara's flask code using the dumped info.	Alter flask applications to only use dynamic data. Push to git for everyone Parse in weather data and display on map.	sql queries not the most up to data.
	Tao	Worked on developing index.html file with all flask modules.	Finalise flask index page and import darraghs/caras javascript	tried to get our code working in object oriented style but unable. Can revert back to this later once we have a working prototype. data scraper stopped. Primary key in weather created problem. So removed from the weather table.
Mon 3 April 2017	**Cara**	Researched google charts and made static chart	Site features document	Need to figure out how to convert sql query to google chart json format.

	Darragh	Merged daily standup documentation. Team meeting. - Researched jquery and javascript - developed weather module	Merge code and import weather info Update trello	Needed to learn jquery and revise javasript which took a lot of time.
	Tao	Created flask app to merge our content	Build in search functionality	Learning jquery and revising javascript.
Tue 4 April 2017	**Cara**	Tasks achieved.	Using javascript to create basic google chart.	
	Darragh	Tasks achieved.	N/A	
	Tao	Tasks achieved.	N/A	
Wed 5 April 2017	**Cara**	Research on Google Charts with our live data and developed basic template.	add chart as an on click event in js. Research on pandas and daily/weekly avg info	
	Darragh	N/A	- First project meeting. - Give icons to Tao - SQL update for most up to date data. Updated flask based on this	Not up to date data on site. Need to alter tables with unique key in sql. Google map is slow in loading data. Need to cache some items.
	Tao	N/A	- Research on search function	
Thu 6 April 2017	**Cara**	Parse in data for google charts	google charts incorporated into infowindow.	
	Darragh	- Updated HTML to create the prototype. - Weather now displayed on top of map.	- N/A	
	Tao	Search function code developed.	- N/A	
Fri 7 April 2017	**Cara**	Google charts developed and need sql queries now.	Work with sql query for google charts. Learning journal update	String sql and figuring out how to manipulate the query.
	Darragh	- N/A	Template developed for final report. Review Retrospective Merge code Learning journals update	
	Tao	- N/A	Merge code Finalise search function Learning journals update.	

Sprint 3

Date	**Team Member**	**What did you do yesterday?**	**What are you going to do today?**	**What impediments did you face?**
Sat 8 April 2017	**Cara**	Review and Retrospective	N/A	
	Darragh	Review and Retrospective	N/A	
	Tao	Review and Retrospective	Basic version of search function developed.	
Sun 9 April 2017	**Cara**	N/A	N/A	
	Darragh	N/A	N/A	
	Tao	N/A	N/A	
Mon 10 April 2017	**Cara**	edited icon bar: twitter, facebook;	trying to finish icon implementation	
	Darragh	Write template for find retrospective work	working on editing icon for google map marker	

	Tao	merge code , update learning journal	trying to implement heatmap function into web , trying to passing chart data to javascript	
Tues 11 April 2017	**Cara**	N/A	N/A	
	Darragh	got half the algorithm for the dataframe working for the charts. Developed icons for the map..	Will look at the algorithm tonight again to see if i can get the datetime working	- once resize the icons, half their content disappears. They become just little blobs. Don't know why. Didn't spend too much time focusing on this as i thought the chart was more important. - unable to get the data between 06:00 to 00:00 for three hour intervals. Keep getting NAN values which wont average. Need to fix this. - need to incorporate the weather data. Not sure how best to do this.
	Tao	N/A	N/A	
Wed 12 April 2017	**Cara**	N/A	N/A	Test following day.
	Darragh	N/A	Icons developed for map pie charts.	Test following day.
	Tao	N/A	N/A	Test following day.
Thur 13 April 2017	**Cara**	N/A		
	Darragh	Icons for map developed.	Developing algorithms for sql to pandas	
	Tao	N/A		
Fri 14 April 2017	**Cara**	N/A	Finalsed the datetime parseing query for google charts. Grouping query by time intervals.	
	Darragh	N/A		
	Tao	N/A		
Sat 15 April 2017	**Cara**	Daily data parsed into map. Ajax query to send station to flask.	working on chart data query/pandas, google chart javascript	
	Darragh		Working on chart data query/pandas	
	Tao		trying to write some test , working on pandas on Jupiter notebook to test weekly query	
Mon 17 April 2017	**Cara**	n/a	easter holiday, exam revision	
	Darragh	n/a	easter holiday, exam revision	
	Tao	/n/a	easter holiday, exam revision	
Tue 18 April 2017	**Cara**	n/a	working on generating daily chart with google chart on the map	
	Darragh	n/a	working on generating daily chart with google chart on the map, working on pass station num to falsk	
	Tao	n/a	working on toggle function on heatmap, merge new code ,working on pass station num to falsk	
	Cara	done	working on weekly chart	

Wed 19 April 2017	**Darragh**	done	re-work on new merged code (css, html), working on weekly chart	
	Tao	done	working on search function auto complete function	
Thur 20 April 2017	**Cara**	n/a	Continuing working on weekly chart, rewrite pandas code, and google chart	
	Darragh	n/a	some css , html refinement; working on weekly chart;	
	Tao	done	working on deploy website on ec2	
Fri 22 April	**Cara**	done	working on javascript for weekly chart , social icon , sprint 3 meeting for retrospective work	
	Darragh	done	finalise css, http, working on weekly chart javascript, merge code, retrospective work	
	Tao	done	Working on deploy on ec2, fix github, do some code comments, write some test,retrospective work	
Sat 23 April	**Cara**	Finalise charts	Social media icons Finalise code	Debugging issue with html/css/js and weekly chart
	Darragh	Finalise charts and html/css	Finalise group report. Finalise html/css Finalise code Finalise burndown.	Debugging issue with html/css/js and weekly chart
	Tao	EC2 and organised retrospective.	Performance tests Unit tests Finalise code Skeleton	
Sun 24 April	**Cara**	Group Report finished. Project finalised	Individual Report	
	Darragh		Individual Report	
	Tao		Individual Report	