1. **Project Overview**
   1. **Introduction**

Dublin Bike is a public bicycle rental scheme which has operated in the city of [Dublin](https://en.wikipedia.org/wiki/Dublin) since 2009. As a user-friendly software, Dublin Bike Monitor is mainly used to monitor, analyze and predict the Dublin Bike availability based on the historical data and give guidance for user on their travel plan.

插图网页最终版本

All of the code/graph/notes are available on GITHUB at: <https://github.com/sedublinbike/SE>

And the working EC2 address of the live website is:

[http://35.160.137.228:5000](http://35.160.137.228:5000/)

The contribution of our group team member is:

|  |  |
| --- | --- |
| Name | Contribution |
| Menghao Su | 34% |
| Renjie Fu | 33% |
| Anna Xuejiao Ge | 33% |

* 1. **Objective**

The main object of this software is to give users an overall understanding of the distribution and coverage of the Dublin Bike, and encourage more people to travel by bike and achieve ‘Green transportation’ as well as ‘Smart city’.

* 1. **Target**

This software helps users to:

1. Get a better look at the usage status and availability of Dublin Bike
2. Get some suggestion on how they can get a bike from a nearby station with sufficient bikes, as well as, in a nice weather.
   1. **Function**

There are five main functions of this software:

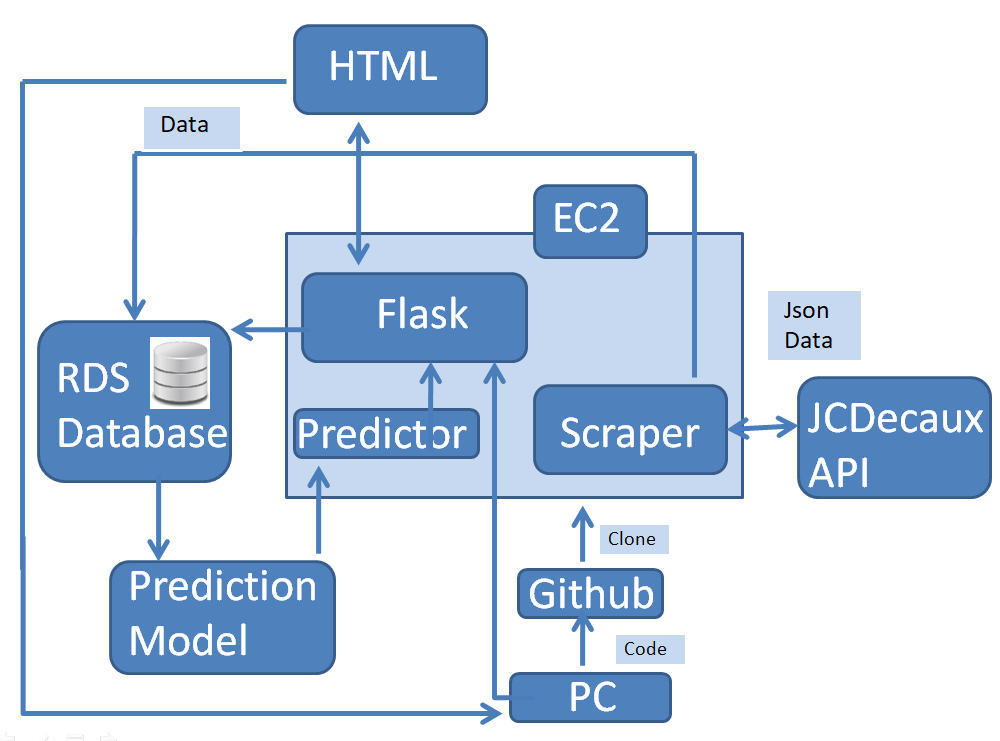
1. Display the distribution of Dublin bike
2. Show the availability and weather condition of each station
3. Predict the future weather and availability of a specific station
4. Show the analytics tendency of each station hourly and daily
5. Schedule the route for users with a give start point and end point

加软件截图

* 1. **Feature**

1. Cloud-based: This software deploys on the AWS RDS and EC2 running 24/7.
2. Chart-visualization: Chart is widely used in our software to give users a direct and clear picture of the availability hourly and daily. Different color of the markers is used to show the different availability of each station in the map.
3. Accurate prediction model: Around 70% accuracy rate in the prediction of availability after cleaning the data and comparing four different models.
4. User-friendly interface and dynamic weather icon:
   1. **Structure**

The Scraper on EC2 is runing continuously and gets data through JCDecaux API all the time before puting the data into AWS RDS. Programmers pulled the Flask template and front-end HTML pages into Github and clone it into EC2. For the prediction feature, we trained four different models locally and used the Random Forest in Predictor.py and deploy it in the cloud.



1. **Team Organization**
   1. **Teamwork**

In this project team, there are three students cooperating together. At the beginning of the whole project, three of us were not quite familiar with each other. But we get a pretty high efficiency in discussion and teamwork through the project, and there are two main reasons for that:

First reason is that we set up a really great communication style and method to solve disagreement. For instance, when we design the interface of the website, three of us had completely different opinions. Su Menghao preferred to use four different pages as homepage, map\_all, search\_station, predict\_future and route\_scheduling separately. While Anna would like to put all functions into one single page and allow users to drop down and see their search or click results. Fu Renjie preferred to use single page but show the availability and prediction results in a popup. We tried all the possible methods and finally decided to use Su’s suggestion, as we continually added in new features and this way could make each feature relatively independent and cause less effect to the others features.

The second reason is the way we divided our workload. We did not assign a big part such as back-end/front-end to a specific team member, as we felt it actually took longer to solve the problem once someone met a particular technical difficulty, and others might not be able to know what’s going on or provide suggestion to solve the problem. We always had a discussion before going into every big part during the sprint planning. Let’s take an example of the prediction feature. We allotted the prediction part into many sub tasks and then divide the sub tasks to each team member. After we discussed the data structure of dataset together, and Su was in charge of cleaning the data, Anna and Fu trained two predict models separately and then Fu deployed the model into the website. In this way, we could learn the needed skills together and help each other to improve. Besides, fully understanding what and how the other’s part is look like helps us to debug and communicate more efficiently as well.

Besides, we divided the whole project period into four sprints and made sure that every team member had an opportunity to be the sprint master of at least one sprint. Therefore, every member could learn Scrum/Agile and know how to set up daily stand up meeting and organize team discussion.

Here is an overview of four sprints:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sprint | Master | Main Tasks | Time estimated | Actual time |
| One | Anna Xuejiao Ge | 1. Set up the EC2 and RDS, 2. Set up group Github account 3. Learn SCRUM and Agile 4. Design data structure and get data through API | 1. 3h 2. 0.5h 3. 2h\*3 4. 3h | 21h |
| Two | Anna Xuejiao Ge | 1. Learn Falsk basic knowledge 2. Achieve the back-end template using FLASK 3. Basic UI design 4. Connect Flask, RDS and HTML website | 1. 30h 2. 20h 3. 20h 4. 40h | 106h |
| Three | Menghao Su | 1. Achieve analysis based on hourly and daily chart 2. Add in weather API 3. Beauty UI | 1. 25h 2. 30h 3. 15h | 72h |
| Four | Renjie Fu | 1. Prediction of availability 2. Schedule the route 3. Beautify UI 4. Deploy the website on EC2 5. Project Report | 1. 30h 2. 25h 3. 20h 4. 2h 5. 20h | 104h |

We all feel grateful to be in such a fantastic team and work together. Everyone in our team shows a humble attitude and everyone learns quite a lot from each other. We agreed on and obeyed the way/mechanism to solve disagreement from the beginning to the end. When we had some different ideas within our team, such as looking for a weather API, we always tried multiple ways to achieve the goal, and tried to find a common/nice way to understand each other’s thinking without causing misunderstanding and tested whether it made sense or not. If not, we tried the others and then we would find the most efficient solution.

All of our team members are willing to share our own opinion and we all like to learn more and contribute more to our project. We all enjoy sharing new ideas on what we just learned with other teammates and other students. And we believe this is just how a good team should be look like.

At the end of this project, we reviewed what we have achieved. Beyond learning the technical skills such as cloud service, Flask (back-end), modeling, we realized that one of the best achievements of this project is learning and attempting to cooperate with others efficiently under advanced team cooperation methods: Agile and Scurm.

Huge thanks our lecturer Lawlor Aonghus and tutor Karl Roe (as product owner) for their nice and patient guidance and help. Lawlor and Karl both helped us to understand the whole project goal and get a big picture from it. Karl assisted us to split the project into four sprints and set goal/main tasks for each one. Lawlor helped us to understand the necessary knowledge on scrum, flask, could service, and the structure, design pattern and real case, which are used in our project as well as pointing us a pathway about what we should continue learning in the future.

* 1. **Time Management**

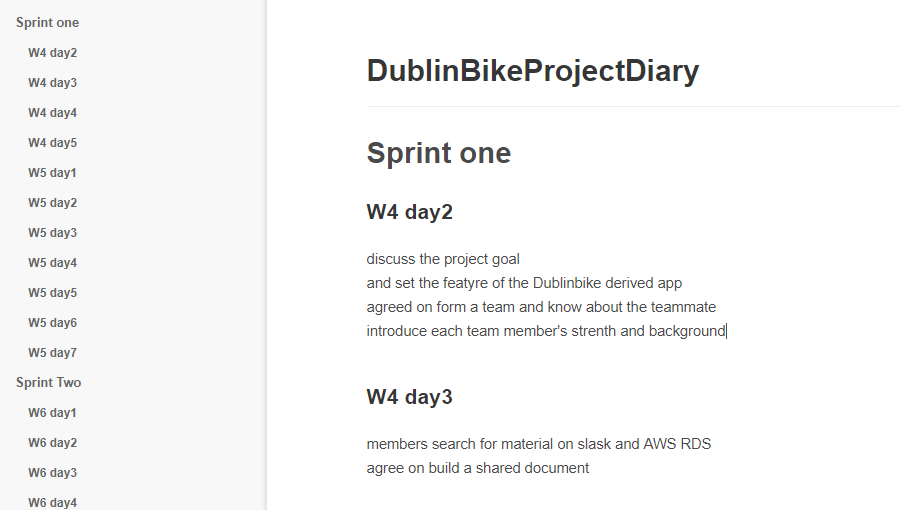
In order to use four sprints efficient and achieve the goal on time, it was necessary for our team to have a good time management style, which contained three basic areas:

1. The stand up meeting to let every team member know how is the project going on.
2. The way to solve the clash on time with other modules (multi-task handling).
3. The way to estimate workload and make adjustment when debugging spent longer time than expected.
4. **Stand up meeting and team discussion**

As the school timetable scheduled, we have class from Monday to Friday, which means all of our team member could meet each other every workday. We have two labs (on Tuesday and Thursday) and two lectures (Monday and Thursday) each week. Therefore we could solve some difficulties we met during the lab and communicate with our tutor/product owner Karl.

For the first and second sprints, we insisted on stand up meeting every day and made daily record on Shimo, which is a cloud based text editor that allows user to edit together and generate directory on the left side automatically. For the third and fourth sprints, as the features became more and more complex, we needed more time to work together and discuss on some technical issues such as transfer the data from prediction model into website. Therefore we agreed to add more hour-long team discussion in library study rooms. Besides, we kept a good record style and fully usage the tool such as Trello and Shimo as well.

Our daily meeting record on the Shimo editor is available at: <https://shimo.im/docs/XEjbgB7qkwYufO6j>



Added team discussion in study room:

|  |  |
| --- | --- |
| Time | Location |
| 10am-12pm, 19th March | Group Study Room A, James Joyce Library |
| 9am-10am, 29th Marth | Group Study Room 1, James Joyce Library |
| 4pm-6pm, 9th April | Group Study Room A, James Joyce Library |
| 12pm-1pm, 12th April | Group Study Room E, James Joyce Library |
| 5pm-7pm, 15th April | Group Study Room 1, Health Center Library |
| 5pm-7pm, 16th April | Group Study Room 2, Health Center Library |
| 9am-10am, 17th April | Group Study Room 5, Health Center Library |
| 10am-3pm, 19th April | Group Study Room F, James Joyce Library |

(The group discussion draft and study room booking information are in the appendix session.)

1. **Multi-task handling**

Due to the fact that this project was not the only one we had during this semester, sometimes we found it was pretty difficult to keep on making contribution to our project every day. For instance, we had four project deadlines and two tests during week 6 and 7, and seven project deadlines plus one lab-test during week 10, 11 and 12.

It pushed us to learn how to handle multi-tasks and improve our skill on time management. The solution we got is making an early plan about the whole stuff of all modules. As we could know the exam time and project deadline two to four weeks in advance, we took these dates into account when making the sprint planning. Besides, we agreed on arranging some self-study tasks such as basic understanding of Flask before mid-term during sprint two and continue to work on our project during spring holiday for one week together.

1. **Estimation of workload**

Our team found that it was a little hard to estimate time/workload when we never did it before. And we found some tasks/sub-tasks took longer time than we expected to be. Besides, it was a little tricky to predict the time needed to debug when our application became more and more complex as well. Therefore at the beginning of each sprint, we tried to arrange some extra time for each main task so that leaving enough time for debugging and learning new stuff.

For example, the dynamic weather icon is one of our best achievements in this project. In order to show weather condition dynamically, we used six existing weather icons to generate eight different ones based on the real weather condition of Dublin that we generated from the database. Then we used gif tool to unify the eight icons’ size. And we used the API called \_\_\_\_\_ to get weather information of each station, rather than the city, and showed the weather condition of both now and two hours later. At the beginning, we felt that weather was not a quite tough feature in our project. But later on we realized that it took a bit longer than we expected in creating gif files, selecting weather API, transferring parameters and so on.

Although we spent longer time in weather feather, we felt it was worth to do so and we achieved a better effect than just using a simple weather plug-in. What’s more, we felt that we learnt quite a lot while designing our own weather feature while presenting the weather condition of each station on website, and got a better understanding of how to connect front-end and back-end through JSON.

When we reviewed the whole project at the end, we found that we improved quite a lot on time management skill. In the first and second sprints we just achieved some basic tasks. While in the third and forth ones, we didn’t only achieve some complex features, but with less time for each feature as well. It also proved that our efficiency improved quite a lot after this project.

1. **Process**
   1. **Overview**

Scurm is an Agile framework for managing project, especially software development. In our project, three students worked together as a team on four sprints, each sprint was two weeks long, except sprint two for three weeks. Every team member got an opportunity to be sprint master and hold stand-up meeting, sprint planning and sprint review. And we had a project retrospective at the end of this project.

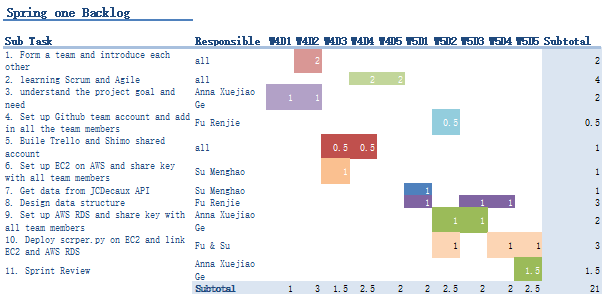
* 1. **Sprint One**

As we started to learn Scrum during sprint one, there is no sprint planning for sprint one and we started to hold stand up meeting since Tuesday in the first week of sprint one (Week 4).

* + 1. **Backlog**

There are four main tasks in sprint one and they could be divided into ten subtasks.

Here is a backlog chart:



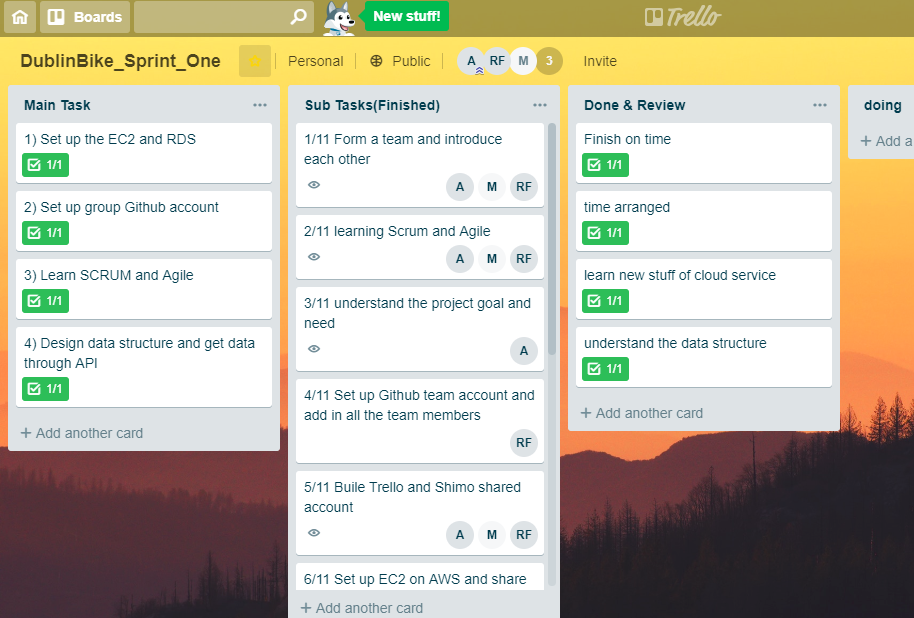
* + 1. **Problem encountered**

When deploying scrper.py onto EC2, it failed multiple times due to lack of some packages. And the reason was that we didn’t install all the needed packages (using pip command) when setting up/initializing EC2 instance. And the key is to find which package that we lacked is. Actually some packages relating to mysql and pymysql have parient-child relations and only after installing specific packages the needed one could be installed, just like how the Linux packages related with each other.

There are three ways to help us solve this problem. First is to reinstall all the needed packages in the vertical environment. Second is running the python file on local environment successfully and then upload the whole environment which containing all the needed packages for sure onto the virtual environment (EC2). And the final method is to use yml and check which package caused this problem in makeup language. Later on, we added in a line of code in the scrper.py that appointed to the MySQLdb package and then the file running successfully. The command is ‘pymysql.install\_as\_MySQLdb()’.

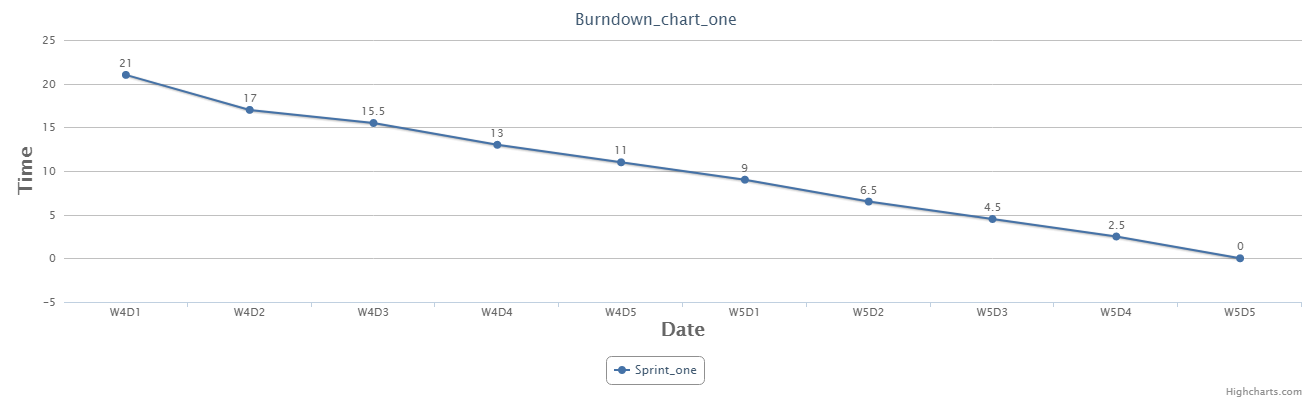
* + 1. **Sprint Review (Burn-down chart)**

In this sprint we achieved four main tasks and 10 subtasks:



**Review:** In this spring, we had a good start of the whole project. And we successfully completed all the planning tasks on time.

And the Burn-down chart is as followed:



* 1. **Sprint two**
     1. **Sprint Planning**

At the beginning of sprint two, we knew that there would be many project deadlines and tests in the following two weeks and then we all agreed to do more jobs during the reading week. Therefore, this sprint is three weeks long and we mainly focus on the understanding of basic Flask Routing/Decorator/Template during week 6 and 7. And then we built our own Flask frame and connect to RDS and HTML/JS in the spring holiday.

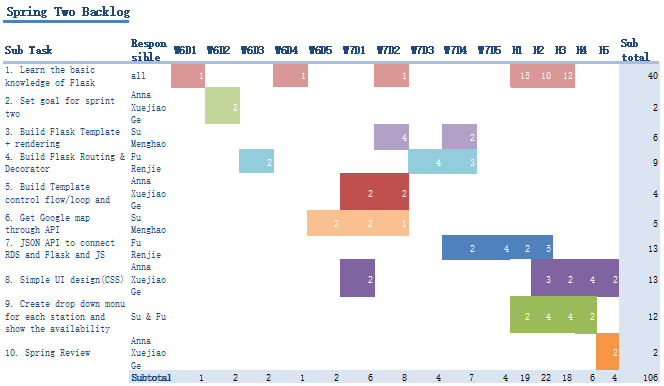
And we agreed to do four main tasks together:

* 1. Learn Flask basic knowledge
  2. Achieve the back-end template using FLASK
  3. Basic UI design
  4. Connect Flask, RDS and HTML website

Then we divided the main tasks into nine sub tasks and separated them to three team members. Please see the following blockage for more details.

* + 1. **Blockage**

There are nine subtasks in this sprint and here is a backlog chart below:

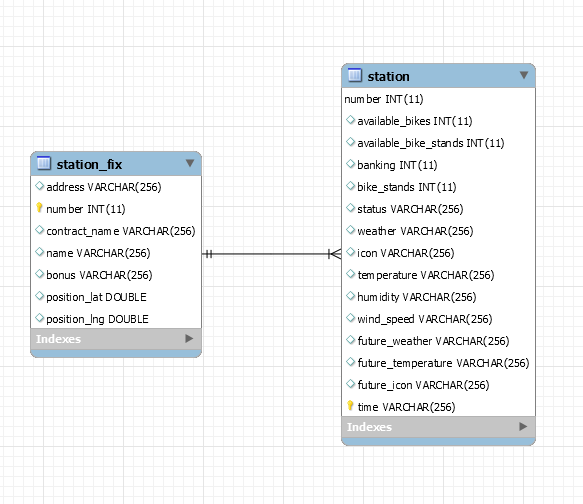


* + 1. **Problem encountered**

It took us a period of time to understand the way of transferring parameter/data between database, Flask and website. Jinjia2 is a full featured template engine for Python. We tried to use Jinja2 template is used to get data. And getJSON based on JSON API could achieve a similar unction as well.

When we were transferring data between Database, Flask and website, we used two Ajax function: one to load all the stations and another to load the information of a specific station. And the second is bedded in the first Ajax function.

Besides, there are two table in our database, at the beginning we use the number as key of both of them. Then we found warning about ‘duplicate key’ when running the file. Then we set a combined key of number and time, while using datatime() to get the time on the moment.

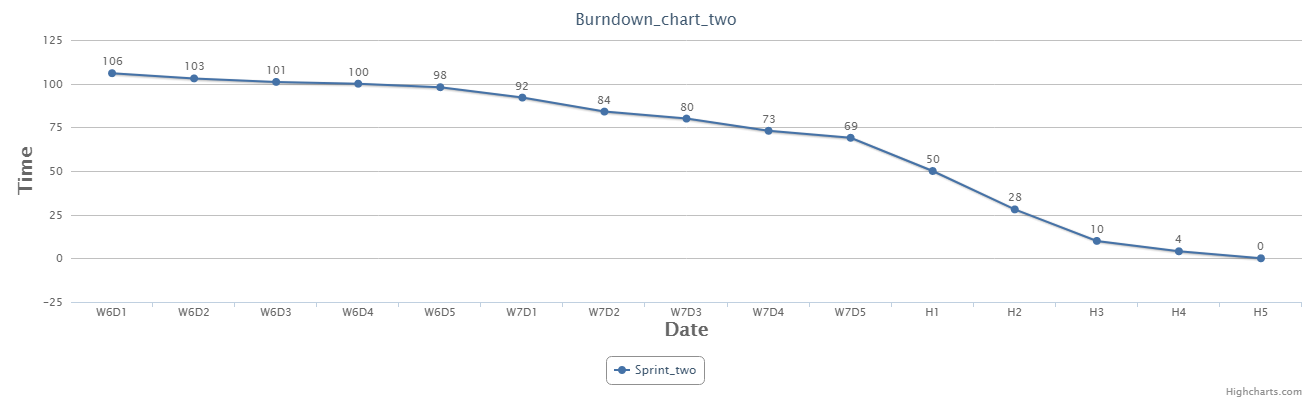


ER

数据库加天气

* + 1. **Sprint Review (Burn-down chart)**

And the Burn-down chart is as followed:



* 1. **Sprint Three**
     1. **Sprint Planning**

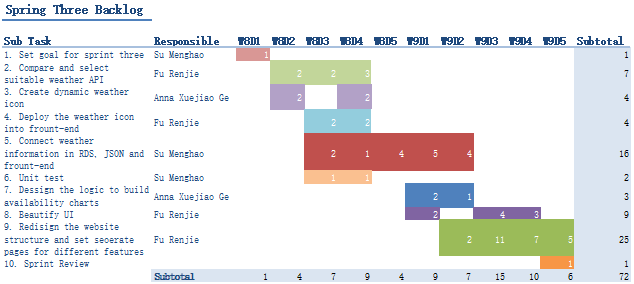
In sprint three, we agreed to do three main tasks together:

1. Achieve analysis based on hourly and daily chart
2. Add in weather API
3. Beauty UI

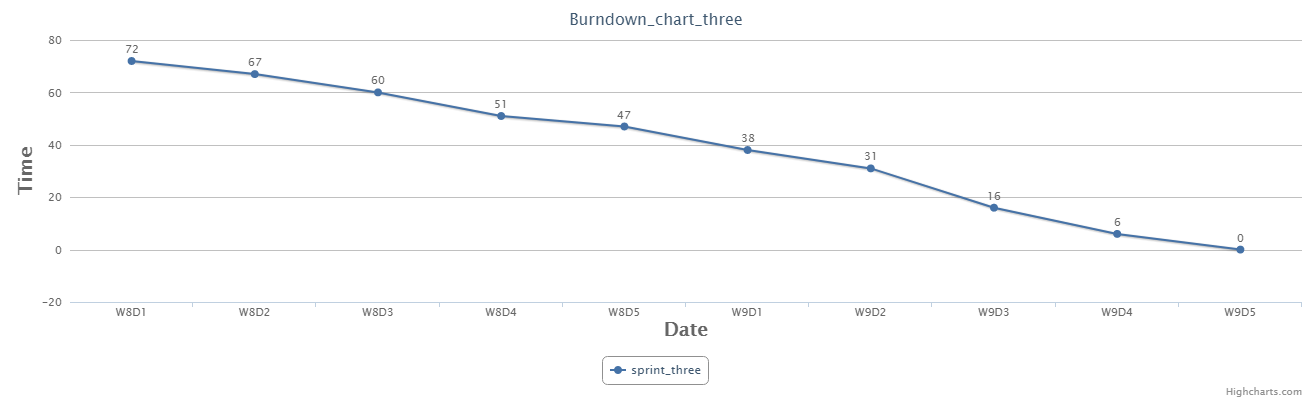
Then we divided the main tasks into ten sub tasks and separated them to three team members. Please see the following blockage for more details.

* + 1. **Blockage**

There are nine subtasks in this sprint and here is a backlog chart below:



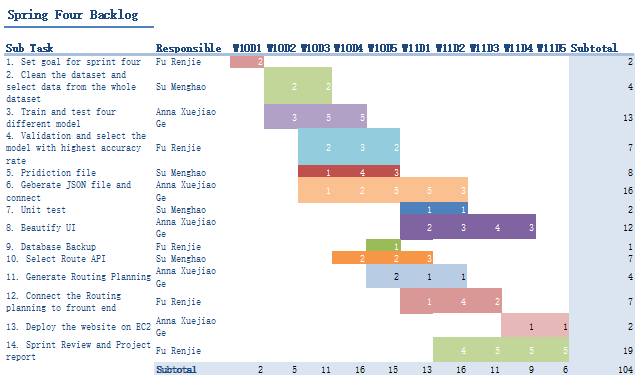
* + 1. **Problem encountered**
    2. **Sprint Review (Burn-down chart)**



* 1. **Sprint Four**
     1. **Sprint Planning**

|  |  |
| --- | --- |
| * + 1. Prediction of availability     2. Schedule the route     3. Beautify UI     4. Deploy the website on EC2     5. Project report |  |

* + 1. **Blockage**



* + 1. **Problem encountered**

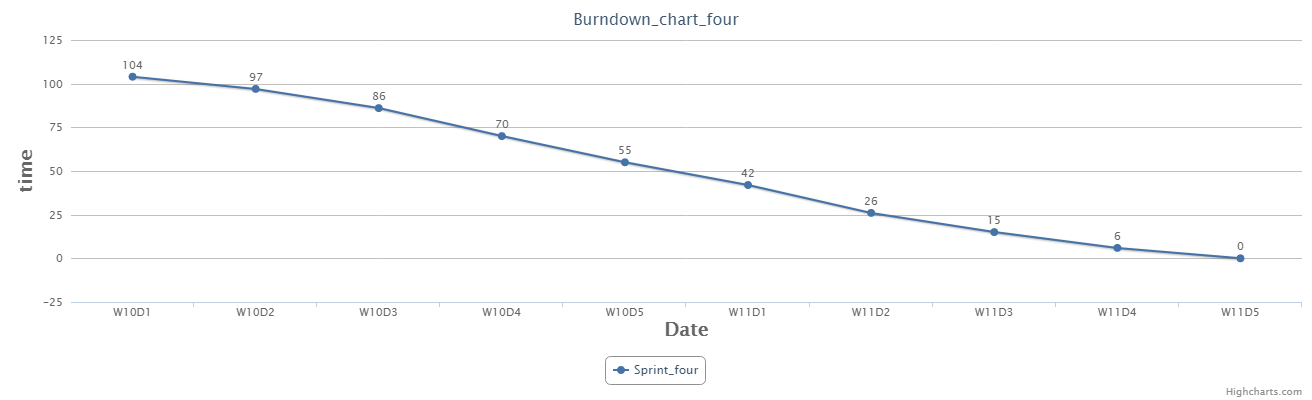
重新训练第二天模型

数据库备份

Test

在ec2上部署

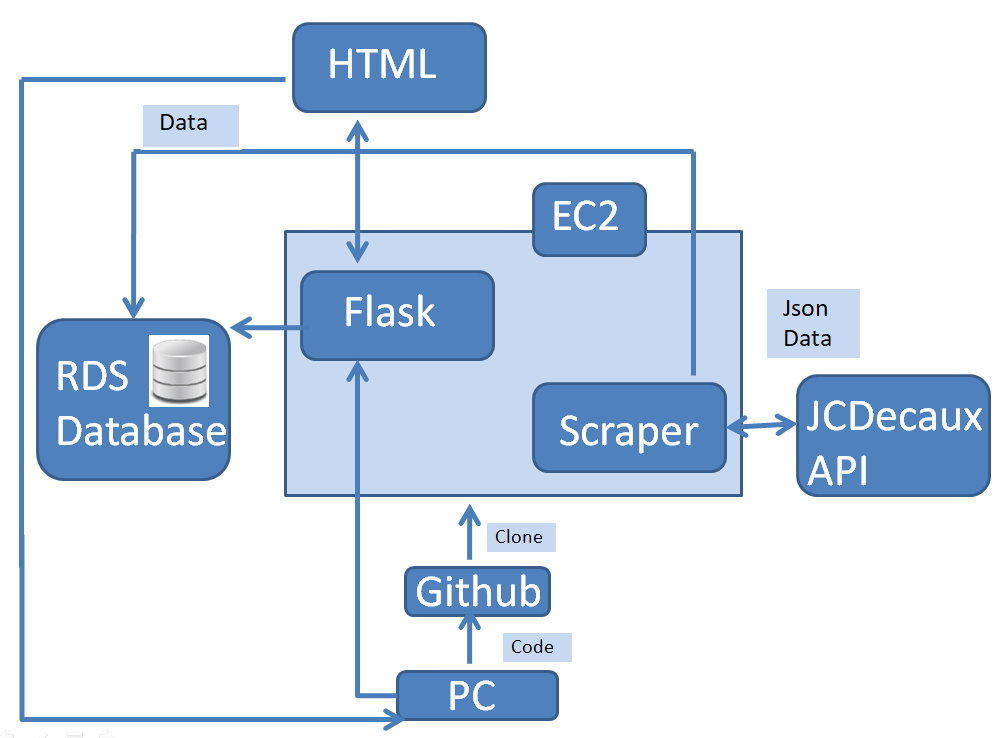
* + 1. **Sprint Review (Burn-down chart)**

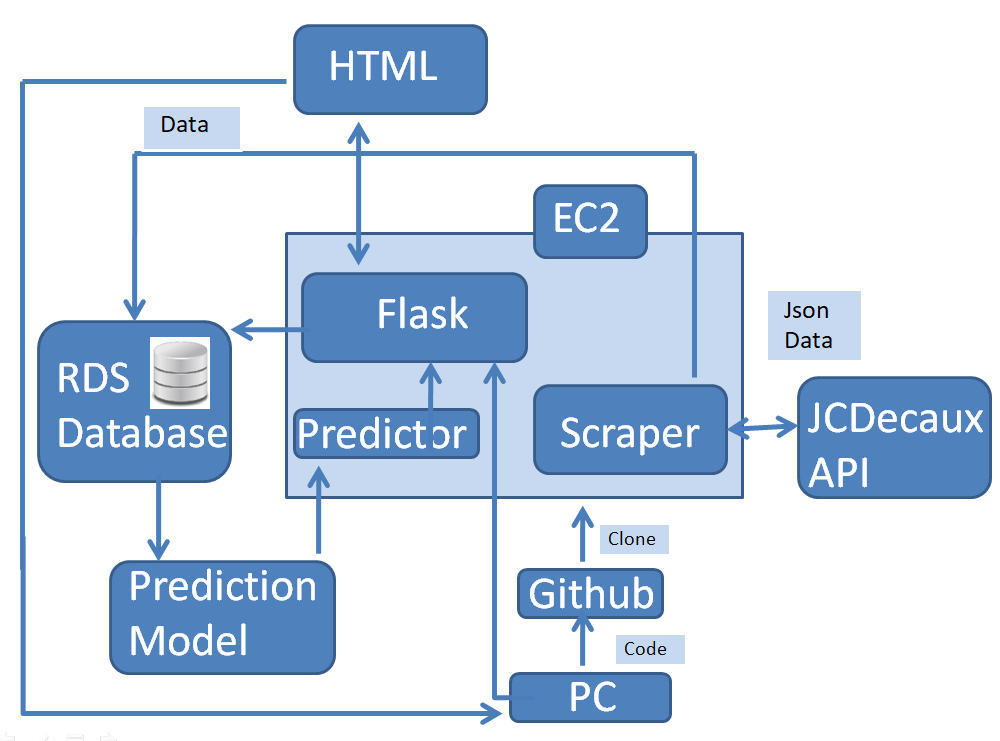


1. **Retrospective**

(问题放在sprint里面还是单列出来)

1. **Architecture**





1. **Analytics**
2. **Design**

软件部署结构加演变

数据结构设计

Analysis设计

预测设计

1. **Future Work**

More quick interaction

Ngix

Javascript

Anjourd os

1. **Reference**

Ec2网站

Github

Shimo 日记

Trello 1,2,3,4

1. **Appendix**

四张backlog

日记待选

手画图

Ui设计手画图

文件归类手画图

预测手画图

