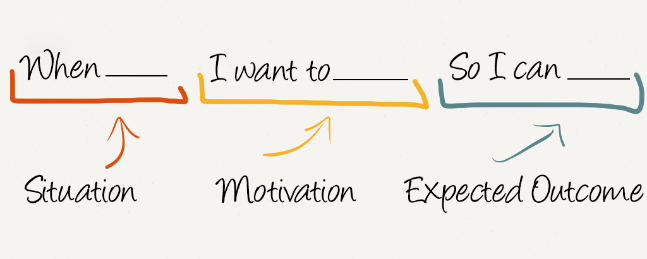
## Sorting a backlog

Break the requirements into tasks. Keep tasks small. Estimate all tasks in hours. Estimate each task as a team. Identify stretch tasks, these are over-committed tasks and should be identified as such. Adds scope in your Sprint Backlog, over and above what we think can be achieved. This is important in order to have something ready if the team delivers early, as the Sprint should ideally remain a fixed length

## 5 Backlog

* **Design**
  + Develop user stories for entire site. – group – 2 hours
  + Develop backlog for scrums 2 and 3.
  + Paper and pen visual design of user stories – 1 hour prep. 30mins agreement
  + Outline expectations for charting. – 2 hours (includes research and itemised requirements for graphing)
  + HTML/CSS of site.
    - Tables/divs for bike and weather info + chart.
    - Responsive design – 1 hour
  + Stretch cards?
* **Development**
  + Merge individual scraping for master branch – 1 hour
  + – 2 hours
  + Parse in data into database – dynamic data, weather data, static data – 3 hours. Done?
  + Display weather info on map- 1 hour
  + Gain access to AWS on windows- 1 hour
  + Use Google Charts to display graph – 3 hours.
  + Password for database?
* **Unit Testing and Error Handling**
  + Parsing in static data/dynamic data – 30 mins
  + Testing connections (database, html page, AWS) – 30 mins
  + Review code and use try, except error handling techniques – 3 hours.
* **Documentation**
  + Learning Journals
  + How to documentation on shared tasks. (in word form and uploaded to git).

## 5 User Stories



1. When I open the webpage, I want to see a map of all the stations for Dublin bikes, so I can click on a station for information.
2. When I click the station, I want to see the available bikes so I can decide to go to a station.
3. When I click more details, I want to see time availability information and influence of weather data so that I can plan a future trip.

Backup the database on git-hub? What security aspects needed?

**Deliverables**

* Data collection through api.
* Data management/storage
* Display bike stations on map
* Occupancy information.
* Weather information.
* Must be interactive (click)
* Error Handling

Github

1. Log of daily standups.
2. Notes from sprint meetings.
3. Retrospectives
4. Feature selection/Product backlog.
5. BurnoutCharts

* What did you do since the last standup?
* What are you doing until the next standup?
* What is stopping you geIng on with your work?

1. During the meeting, the funcGonality that was created during the sprint is demonstrated to the product owner (demonstrator)

**product backlog** - requirements for the project expressed as a priorised list of backlog items cannot be changed until the next sprint planning meeting

* **Design**
* **Development**
* **Unit Testing & Error Handling**
* **Management and Administration**

<http://www.yodiz.com/blog/product-backlog-vs-sprint-backlog-difference-in-agile-methodology/>

**sprint backlog** - subset of product backlog items that are defined as part of the work for a particular sprint usually internal to the team (does not include the product owner)

xzxz

**Product Backlog**

* **Development**
  + Debugging of web scraping framework – 5 hours TAO
  + Domain for website? – 2 - TAO
  + Create setup.py files and install scraping on AWS for live data grabbing – 2 - TAO
  + Multilevel access to AWS Account- 2 - All
  + Research data analytic features for dynamic data – 2 - All
  + Research data analytic features for weekly data – 2- All
* **Unit Testing and Error Handling - Tao**
  + Parsing in static data/dynamic data – 1 hour – 2 -all
  + Query testing – 1 hour - all
  + Testing connections (database, html page, AWS) – 1 hour – 2 – all
  + Review code and use try, except error handling techniques – 3 hours - all
  + Test security and load balancing- 2 hours - all
* **Management and Administration**

**Items to clarify**

1. **Data storage.** Do what is the best approach to storing large amounts of data. Do we only add to the sql database if it’s changing from the previous scrape? If so, wouldn’t that reduce performance because of the constant looping needed?
2. **Daily Standups.** How do we handle these when they’re not necessarily daily?
3. **Error Handling**. What advice can we get on try/except/errorhandling?
4. **Hosting the domain.**