# Requirement

Task 1: Add a REST API endpoint to allow a user to add a rating to a beer.   
  
This endpoint should accept an id parameter and JSON request body which includes the following properties: username, rating, comments.  
Add validation to ensure the id parameter is a valid beer id by querying the Punk API.  
Add validation to ensure that the rating is a valid value in the range of 1 to 5.  
If any of the validations fail an applicable error should be returned to the user.  
All valid requests should append the JSON from the request body to a file called database.json stored within a solution folder of your choosing.  
  
Task 2: Add a REST API endpoint to retrieve a list of beers.  
  
This endpoint should accept one parameter in the query string of the request. The purpose of this parameter is to denote the name of the beer to search for.  
The implementation of your REST endpoint will use the public available Punk API to retrieve all beers matching the search parameter described above.  
After retrieving the search result, load the contents of the database.json file into an in-memory collection and write a Linq query to project an API response that follows the structure of the following example:  
[

{  
 "id":"1",  
 "name":"Buzz",  
 "description":"A light, crisp and bitter IPA brewed with English and American hops. A small batch brewed only once.",  
 "userRatings":[

{  
 "username":"john.doe@fictitious.com",  
 "rating":3,  
 "comments":"Lorem ipsum dolor sit amet, consectetur adipiscing elit"

},  
 {

"username":"max.power&#64;fictitious.com",  
 "rating":5,  
 "comments":"sed do eiusmod tempor incididunt ut labore"

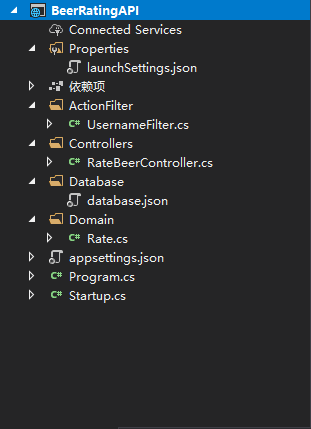
}]

}

]  
  
Task 3: Add a custom Web API Action Filter Attribute  
  
This action filter attribute is for the purpose of validating the username supplied in the JSON body of the REST operation described in task 1. The validation should use a regular expression to ensure the username value is a valid formatted email address. If the username is not valid an applicable error should be returned by the API.  
  
Task 4: Add unit tests  
  
Add applicable Unit tests to a routine of your choosing that formed part of the implementation of one of the above tasks.

# Implementation

Structure



Task 1: Add a REST API endpoint to allow a user to add a rating to a beer.   
Endpoint URL (POST request): <http://localhost:50932/api/RateBeer/{beer_id}>

1. First Endpoint will use filter attribute to validate username
   1. If not, an error will be return
2. Then the method will talk to punk API to validate the beer id
   1. If not, an error will be return
3. After that the method will check the rate value is with in 1 -- 5
   1. If not, an error will be return
4. When all of above step is ready and checked, the method will create the domain object and save the Json format data in to database.json

Task 2: Add a REST API endpoint to retrieve a list of beers.  
Endpoint URL (GET request): [http://localhost:50932/api/RateBeer/{beer\_name}](http://localhost:50932/api/RateBeer/{beer_id})

1. First the endpoint will talk to the punk API to get a list of beer base on the beer name in the request parameter
   1. If the search result is empty then the endpoint will return empty json array (no other information will be return, just empty json array)
2. Then the method will read from databse.json file to load all rate data in to memory
3. After that the method will combine the data from the two steps above and form the final Json array with required format

Task 3: Add a custom Web API Action Filter Attribute  
The Filter attribute extend from “Attribute” and “IAuthorizationFilter”

Override OnAuthorization method and using syncIOFeature from IHttpBodyControlFeature to read form request body. The method will use regular expression to check the username follow the proper email format. If not status code 406 not acceptable will be return.

Task 4: Add unit tests

Unit test is done though a C# ASP.NET console app. It only cover task one.

Has complete test on expected vale, boundary value and unexpected value.

* Valid beer id and request body
* Valid information with rate value be 1 and 5
* Not valid beer id
* Not valid rate value
* Not valid username

If all test passed “All test passed” will show on the console

Others

The API and the test are in different folder under one Repo.

The API has test beer data in it. DO NOT modify the data in the database.json file(the data in the database.json file dose not have the correct format, but this is intentional).If want to reset the rating data just remove the database.json file. The API has the ability to recreate it.