Business Case # 1	**Add a REST API endpoint to allow a user to add a rating to a beer**
Description	1. This endpoint should accept an id parameter and JSON request body which includes the following properties: username, rating, comments. 2. Add validation to ensure the id parameter is a valid beer id by querying the Punk API. 3. Add validation to ensure that the rating is a valid value in the range of 1 to 5. 4. If any of the validations fail an applicable error should be returned to the user. 5. 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.
Operation Type	POST
API URL	**http://localhost:54716/api/ratings/BeerID** **example - http://localhost:54716/api/ratings/7**

Input Parameters	• ID: number (Int32)
	• UserRatings (New Object):
	  ○ Username: string
	  ○ Rating: number [1-5]
	  ○ Comments: string

Output parameters	• Status Code • Success/Error Message • Rating details

Validations	• ID
	  ○ Is Not NULL
	  ○ Number
	  ○ Validate id in Punk API - https://api.punkapi.com/v2/beers/1
	• UserRatings
	  ○ Username:
	  ▪ NOT NULL
	  ○ Rating
	  ▪ NOT NULL
	  ▪ Number
	  ▪ Range 1-5
	  ○ Comments
	  ▪ N/A

Business Logic	• Validations
	  ○ Fail → Status code BAD REQUEST with error messages
	  ○ Pass → Append request to database.json JSON and return ratings object to

Dependent URLs & Operations	https://api.punkapi.com/v2/beers/1

Testcases	N/A

Business Case # 2	**Add a REST API endpoint to retrieve a list of beers**

Description	1. 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.
	2. The implementation of your REST endpoint will use the public available Punk API to retrieve all beers matching the search parameter described above.
	3. 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@fictitious.com",
	"rating":5,
	"comments":"sed do eiusmod tempor incididunt ut labore"}]
	}]

Operation Type	GET
API URL	**http://localhost:54716/api/beers?name=inputSearchParameter** **example: http://localhost:54716/api/beers?name=AB:**
Input Parameters	• name: string • Optional Parameter (result default limit = 25) and user can customize

Output parameters	• Status Code
	• Success/Error Message
	• BeersWithReviews
	  ○ Id
	  ○ Name
	  ○ Description
	  ○ UserRatings (List)
	  ▪ Username
	  ▪ Rating
	  ▪ Comment

Validations	• Name ○ Is Not NULL

Business Assumptions	• Result set returns max of 25 records • In next release will implement customizable results size

Business Logic	• Validations
	  ○ Fail → Status code BAD REQUEST with error messages
	  ○ Pass → Append request to database.json JSON

Dependent URLs & Operations	https://api.punkapi.com/v2/beers?beer_name=AB:
Testcases	N/A

Business Case # 3	**Add a custom Web API Action Filter Attribute**
Description	1. 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.
Operation Type	Action Filter that executes before Action is executed
API URL	**http://localhost:54716/api/ratings/BeerID** **example - http://localhost:54716/api/ratings/7**

Input Parameters	• ID: number (Int32)
	• UserRatings (New Object):
	  ○ Username: string
	  ○ Rating: number [1-5]
	  ○ Comments: string

Output parameters	• Status Code • Success/Error Message ▪ Rating details
Validations	• UserRatings.Username ○ Must be a formatted email
Business Assumptions	• None

Business Logic	• Validations
	  ○ Fail → Add error messages to ModelState messages
	  ○ Pass → Proceed to Action execution

Dependent URLs & Operations	https://api.punkapi.com/v2/beers/1
Testcases	N/A