CODE MASTER

This document describes Code Master Final Round that includes challenges for 2020-2021 period.

OBJECTIVE

Code Master Final Round has 5 tasks which 3 of them are provided for the first part and other 2 for the second. Those tasks require integrations with JIRA Rest APIs. The objective is to develop solutions that fulfills the requirements described for each task.

Developed solution should meet all requirements listed below.

- Should provide a Restful webservice for public use
- Determined response should be in JSON format
- codemaster.obss.io/jira/ should be used as JIRA Endpoint
- bit.ly/2SYuW5g should be used as JIRA Rest API documentation.
- Provided service URLs should start with http://localhost:8080/api/v1. Tasks are detailed at Tasks page. If you have completed first three tasks, you can continue with fourth one.
- Developed service should be available publicly. No authentication should be requirement.
- Provided JIRA API doesn't require authentication.
- Provided solution doesn't require any user interface.
- A bitbucket repository will be provided to each candidate. **After the given time is over, commits will be blocked. It will not be possible to push any other source code.**
- The response which is in JSON format, should have "*application/json*" content-type in http response header except Echo Service task.

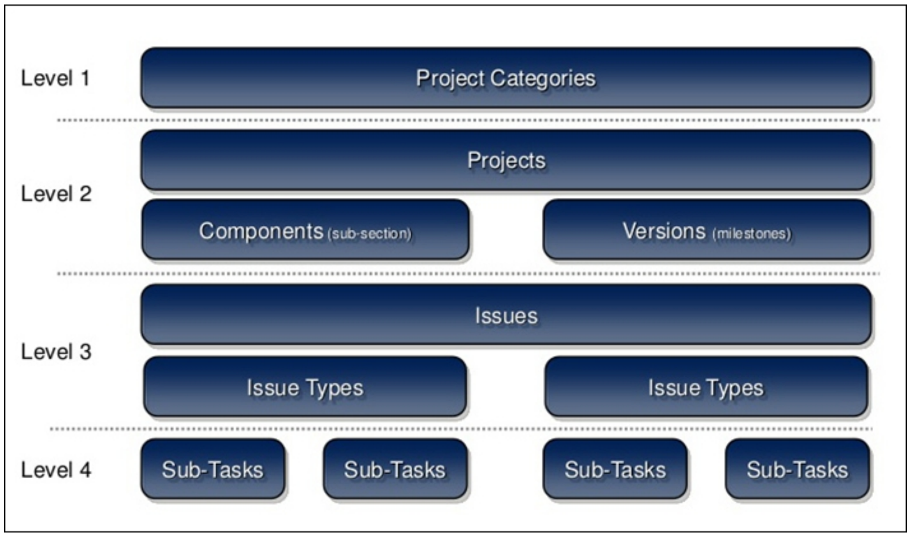
ABOUT JIRA

"Jira is a family of products built to help all types of teams manage their work. Jira offers several products and deployment options that are purpose-built for Software, IT, Business, Ops teams, and more."

atlassian.com

Jira is used for issue tracking and project management. The platform leverages all kinds of project management skills, including software development, Agile project management, bug tracking, scrum management, content management, marketing, professional service management, and so much more.

JIRA CONCEPTS



- **Project Categories**: When there are too many projects in JIRA, it becomes important to segregate them into various categories. JIRA will let you create several categories that could represent the business units, clients, or teams in your company.

- **Projects**: A JIRA project is a collection of issues. Your team can use a JIRA project to coordinate the development of a product, track a project, manage a help desk, and so on, depending on your requirements.

- **Components**: Components are subsections of a project. They are used to group issues within a project to smaller parts.

- **Versions**: Versions are point-in-time for a project. They help you schedule and organize your releases.

- **Issue Types**: JIRA will let you create more than one issue types that are different from each other in terms of what kind of information they store. JIRA comes with default issue types, such as bug, task, and subtask, but you can create more issue types that can follow their own workflow as well as have different set of fields.

- **Sub-Tasks**: Issue types are of two types: standard and subtasks, which are children of a standard task. For instance, you can have test campaign as standard issue type and test cases as subtasks.

TASKS

1. Echo Service

Should provide a Restful web service which returns back what you sent as an input. Response should be in body as raw type.

Service URL	http://localhost:8080/api/v1/echo/{**queryText**}
HTTP Method	GET
HTTP Return Code	200
Expected Response Body	{queryText}

2. Issue Type Lists

Should provide a Restful web service which returns **list of issue types** defined on JIRA server.

Service URL	http://localhost:8080/api/v1/issuetypes?jiraUrl={jiraUrl}
Inputs	"**jiraUrl**" will be provided as query parameter which contains Jira Api URL
HTTP Method	GET
HTTP Return Code	200
Response Body Type	JSON
Content-Type	application/json
Expected Response Body (Example)	see below

Expected Response Body (Example)

```
[
    …,
    {
        "id": "18910",
        "description": "Task description text",
        "name": "Improvement",
        "subtask": false
    },
    …
]
```

3. Subtask Type Issues in Selected Project

Should provide a Restful web service which returns **issues which are subtask type in selected project** defined on JIRA Server.

Service URL	http://localhost:8080/api/v1/issues/subtasks?jiraUrl={**jiraUrl**}&projectId={**projectId**}
Inputs	"**jiraUrl**" will be provided as query parameter which contains Jira Api URL
	"**projectId**" will be provided as query parameter which contains project id on Jira Server
HTTP Method	GET
HTTP Return Code	200
Response Body Type	JSON
Content-Type	application/json

Expected Response Body (Example)	
	```
	{
	  "issues": [
	    …,
	    {
	      "id": "1469109",
	      "key": "I18N-2997",
	      "fields": {
	        "assignee": {
	          "name": "mheighway@atlassian.com",
	          "key": "mheighway@atlassian.com",
	          "emailAddress": “mheighway@atlassian.com”,
	          "displayName": "Melanie Heighway"
	        },
	        "reporter": {
	          "name": "mheighway@atlassian.com",
	          "key": "mheighway@atlassian.com",
	          "emailAddress": “mheighway@atlassian.com”,
	          "displayName": "Melanie Heighway"
	        },
	        "project": {
	          "id": "11460",
	          "key": "I18N",
	          "name": "Atlassian Translations"
	        }
	      }
	    },
	    …
	  ]
	}
	```

4. Find top N Users

Should provide a Restful web service which returns **top n issue assigned users (assignee) sorted by issue count on given projects** defined on Jira server. **N indicates the number of users that response should contain.**

Service URL	http://localhost:8080/api/v1/users/find-top-n-users?jiraUrl={**jiraUrl**}&topn={**topn**}
Inputs	“**jiraUrl**” will be provided as query parameter which contains Jira Api URL
	“**topn**” will be provided as query parameter which contains the most N user count on Jira Server
	List of Project Ids will be provided on request body
Request Body Type	JSON
Expected Request Body	`[ "ACCESS", “I18N” ]`
HTTP Method	POST

HTTP Return Code	200
Response Body Type	JSON
Content-Type	application/json

Expected Response Body (Example)

```
[
    …,
    {
        "name": "rmacalinao",
        "key": "rmacalinao",
        "emailAddress": null,
        "displayName": "Ramon Macalinao",
        "issueCount": 27
    },

    …
]
```

5. Find Top M Projects Min N Issues

Should provide a Restful web service which returns **top m projects (sorted by number of issues) that given users are the assignee of n issues at minimum** defined on Jira Server. **N is the number of issues assigned to given users and m indicates number of the projects response should contain.**

Service URL	http://localhost:8080/api/v1/projects/find-min-n-issues?jiraUrl={**jiraUrl**}&minn={**minn**}&topm={**topm**}
Inputs	“**jiraUrl**” will be provided as query parameter which contains Jira Api URL
	“**minn**” will be provided as query parameter which contains the min N user count on Jira Server *This parameter is **optional**, **default value** is “5”*
	“**topm**” will be provided as query parameter which contains the top M projects on Jira Server *This parameter is **optional**, **default value** is “10”*
	User list will be provided on request body
Request Body Type	JSON

Expected Request Body

```
[
   "mheighway@atlassian.com",
   "franz",
   "serge"
]
```

HTTP Method	POST
HTTP Return Code	200
Response Body Type	JSON
Content-Type	application/json

Expected Response Body (Example)

```
[
    …,
    {
        "id": "11460",
        "key": "I18N",
        "name": "Atlassian Translations",
        "issueCount": 16
    },
    …
]
```

Addition to requirements in above, your solution should handle some exceptions detailed at below

Exception #	**Definition**	**Expected Response Body**	**Http Code**
Exception# 1	User information is empty. There is just empty array on request body.	`{ "timestamp": "2020-02-11", "status": 500, "errors": "users not found" }`	500
Exception# 2	User information is missing. There is no data on request body.	`{ "timestamp": "2020-02-11", "status": 400, "errors": "Required request body is missing: }`	400
Exception# 3	jiraUrl is missing	`{ "timestamp": "2020-02-11", "status": 400, "errors": "Required String parameter 'jiraUrl' is not present" }`	400
Exception# 4	Jira is not available	`{ "timestamp": "2020-02-11", "status": 500, "errors": "Jira is not available" }`	500
Exception# 5	Request method 'GET' not supported"	`{ "timestamp": "2020-02-11", "status": 405, "errors": "Request method 'GET' not supported" }`	405
Exception# 6	Request method 'PUT' not supported"	`{ "timestamp": "2020-02-11", "status": 405, "errors": "Request method 'PUT' not supported" }`	405

CONSTRAINTS

Responsiveness of the service is relevant. Service should not be unresponsive or down, even if remote system fails to respond.

EVALUATION

Provided solutions will be evaluated according to the following criteria:

- Correctness
- Performance
- Robustness of the solution
- General code quality and reliability

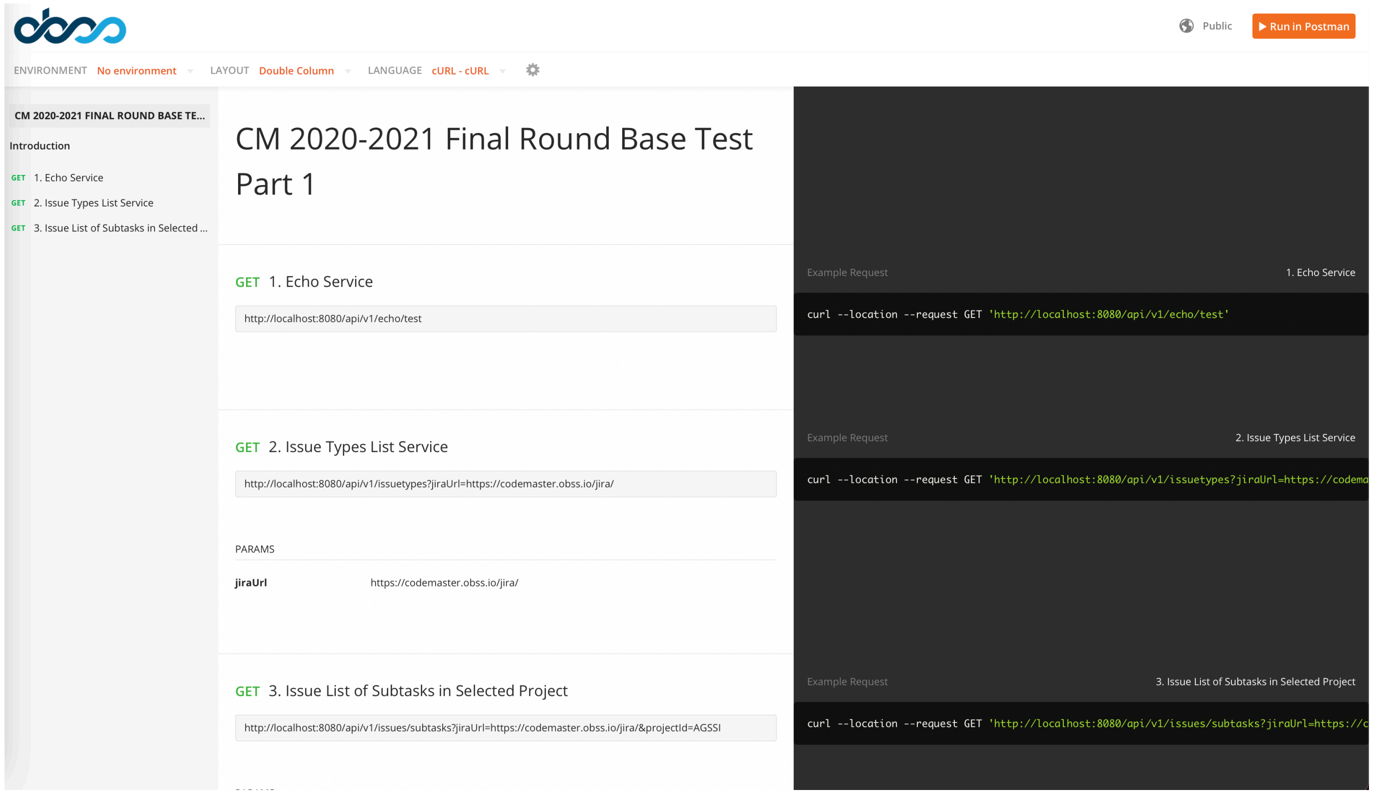
SUBMISSION

Source code and related documents should be submitted to Git repositories before specified time limit exceeds. Otherwise, the solutions will not be assessed.

TESTING

Postman Client will be used for testing for your solution. Therefore, you should follow the directions at below to setup your test environment.

1. Download latest Postman Client from https://www.postman.com/downloads/
2. Visit bit.ly/2SwPNjD to download test collection prepared for postman app. Web page similar to the image shown below should be opened


Public
Run in Postman
ENVIRONMENT No environment
LAYOUT Double Column
LANGUAGE cURL - cURL
CM 2020-2021 FINAL ROUND BASE TE...
Introduction
GET 1. Echo Service
GET 2. Issue Types List Service
GET 3. Issue List of Subtasks in Selected ...

CM 2020-2021 Final Round Base Test Part 1

GET 1. Echo Service

http://localhost:8080/api/v1/echo/test

Example Request — 1. Echo Service

```
curl --location --request GET 'http://localhost:8080/api/v1/echo/test'
```

GET 2. Issue Types List Service

http://localhost:8080/api/v1/issuetypes?jiraUrl=https://codemaster.obss.io/jira/

PARAMS

jiraUrl	https://codemaster.obss.io/jira/

Example Request — 2. Issue Types List Service

```
curl --location --request GET 'http://localhost:8080/api/v1/issuetypes?jiraUrl=https://codema
```

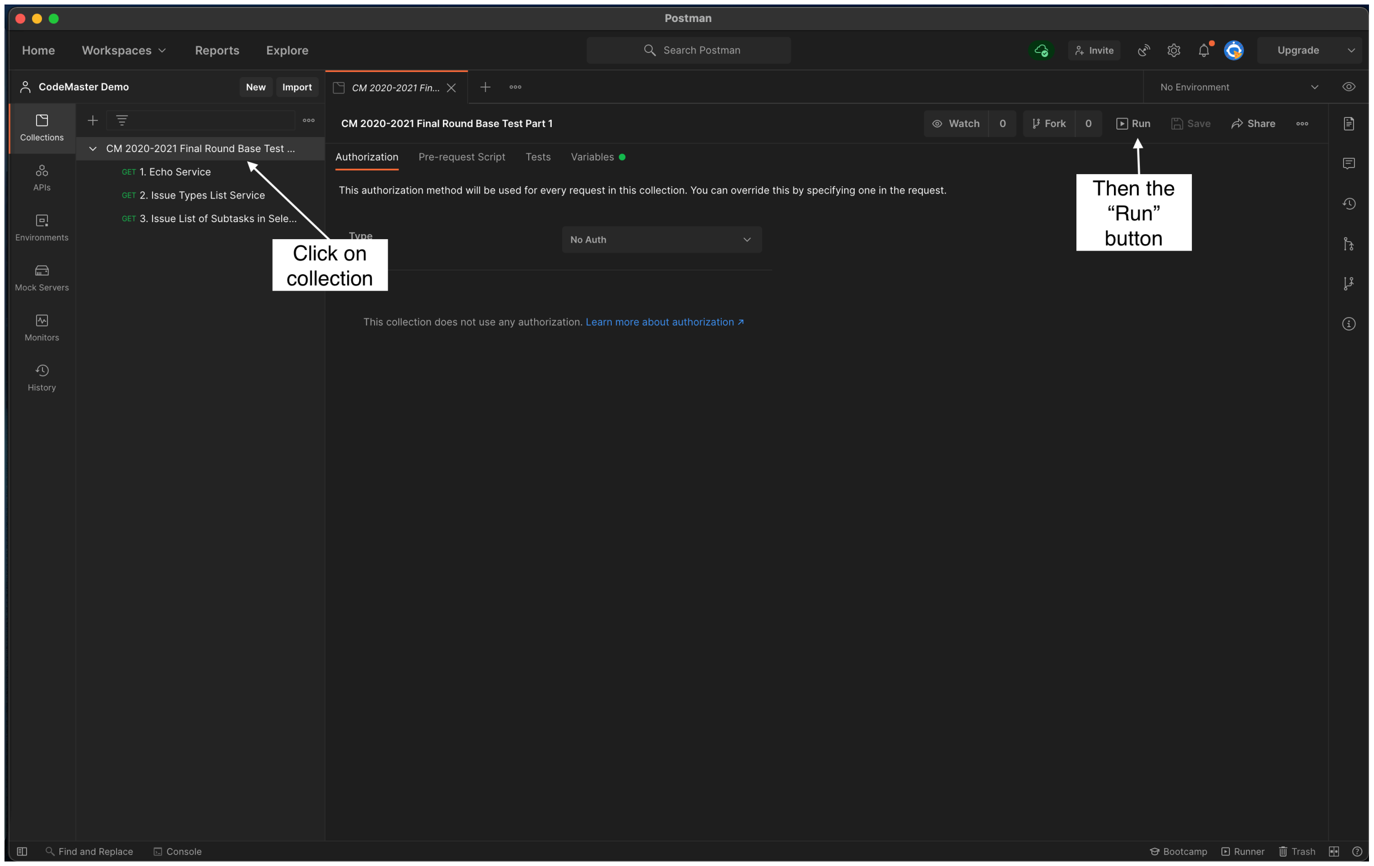
GET 3. Issue List of Subtasks in Selected Project

http://localhost:8080/api/v1/issues/subtasks?jiraUrl=https://codemaster.obss.io/jira/&projectId=AGSSI

Example Request — 3. Issue List of Subtasks in Selected Project

```
curl --location --request GET 'http://localhost:8080/api/v1/issues/subtasks?jiraUrl=https://c
```

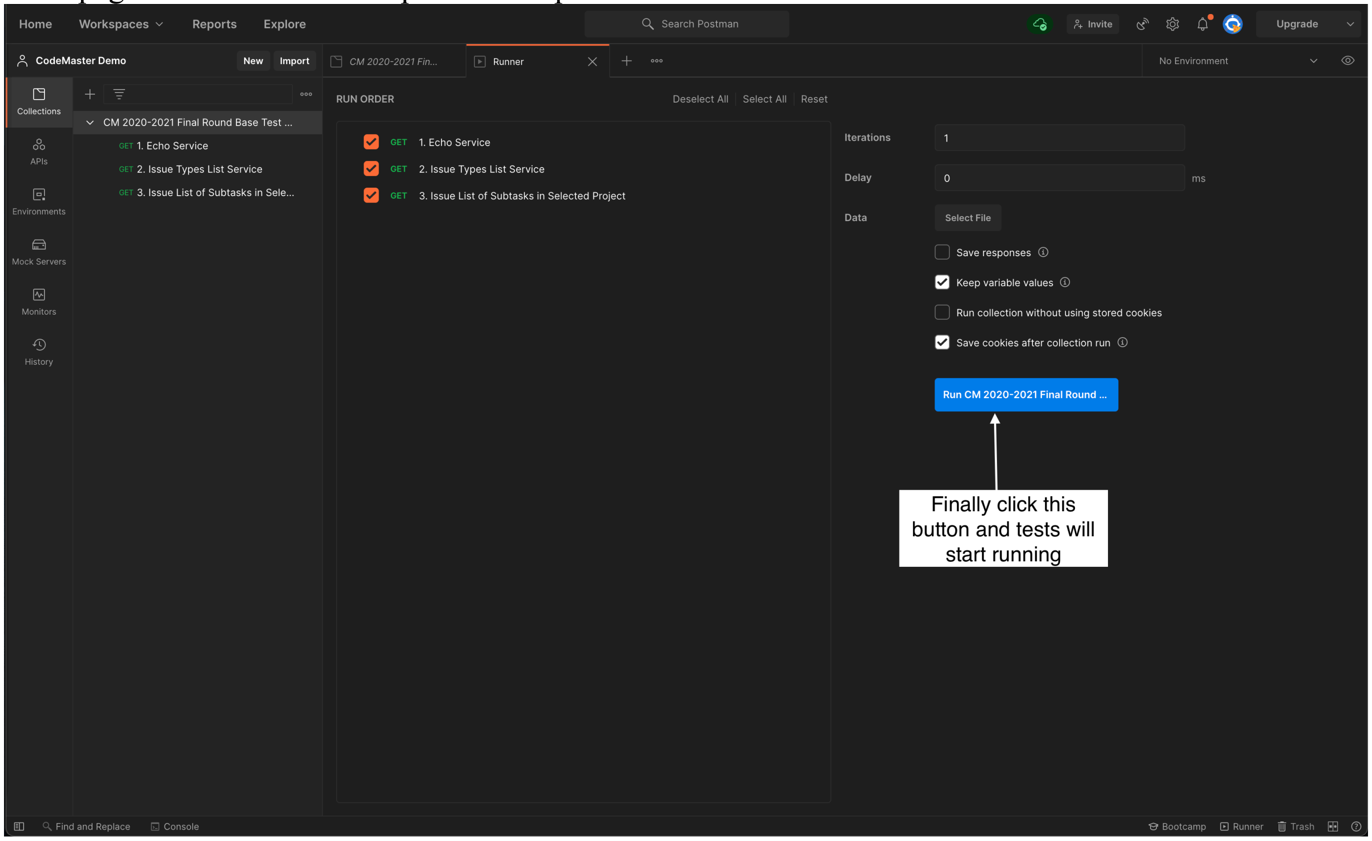
3. Click "Run in Postman" button at right corner on the top, to import test collection to local workspace on postman
4. After import of the test collection, click on the collection and then "run" button.



This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

This collection does not use any authorization. Learn more about authorization ↗

5. Final page will be shown after previous step.



CM 2020-2021 Final Round Base Test ...
- GET 1. Echo Service
- GET 2. Issue Types List Service
- GET 3. Issue List of Subtasks in Sele...

RUN ORDER — Deselect All | Select All | Reset

- [x] GET 1. Echo Service
- [x] GET 2. Issue Types List Service
- [x] GET 3. Issue List of Subtasks in Selected Project

Iterations: 1

Delay: 0 ms

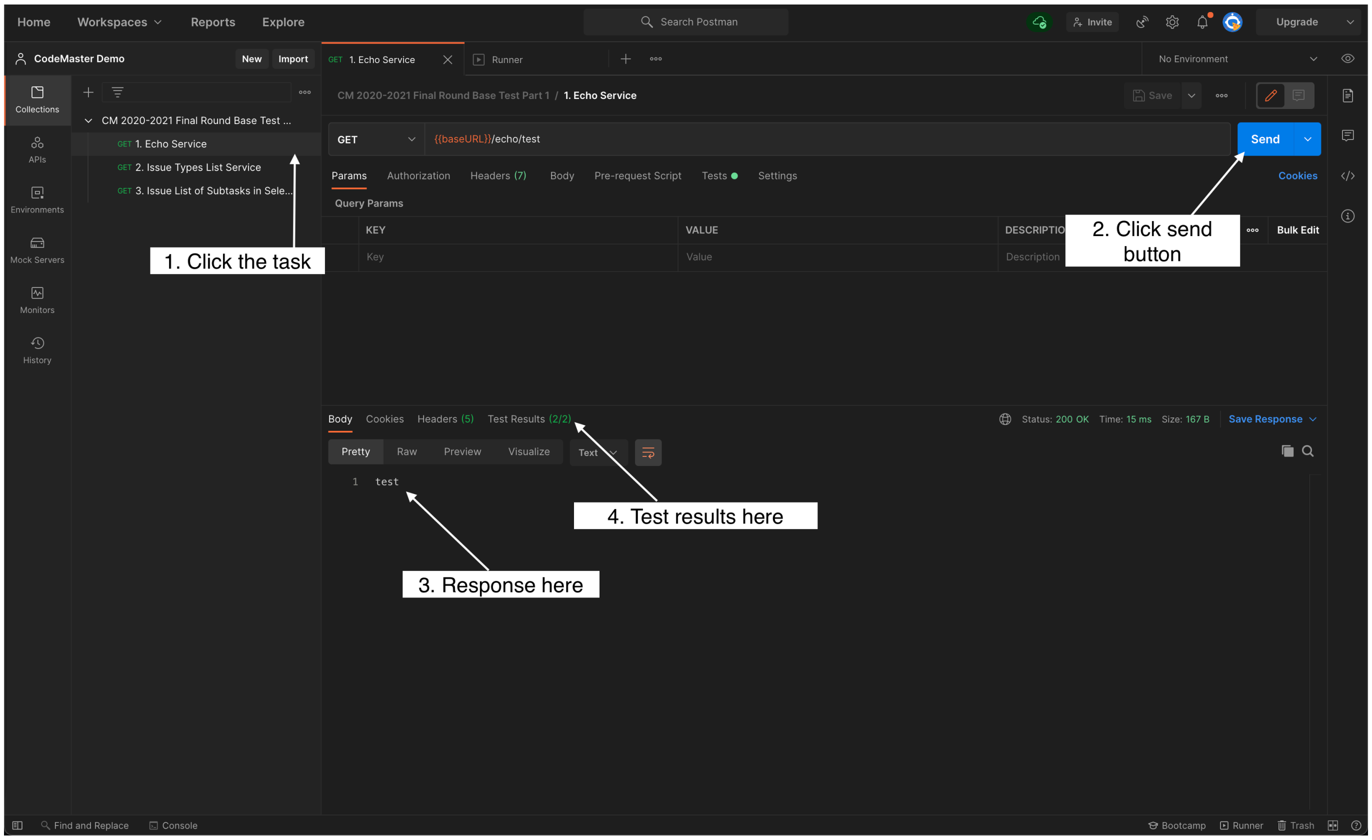
Data: Select File

- [ ] Save responses
- [x] Keep variable values
- [ ] Run collection without using stored cookies
- [x] Save cookies after collection run

Run CM 2020-2021 Final Round ...

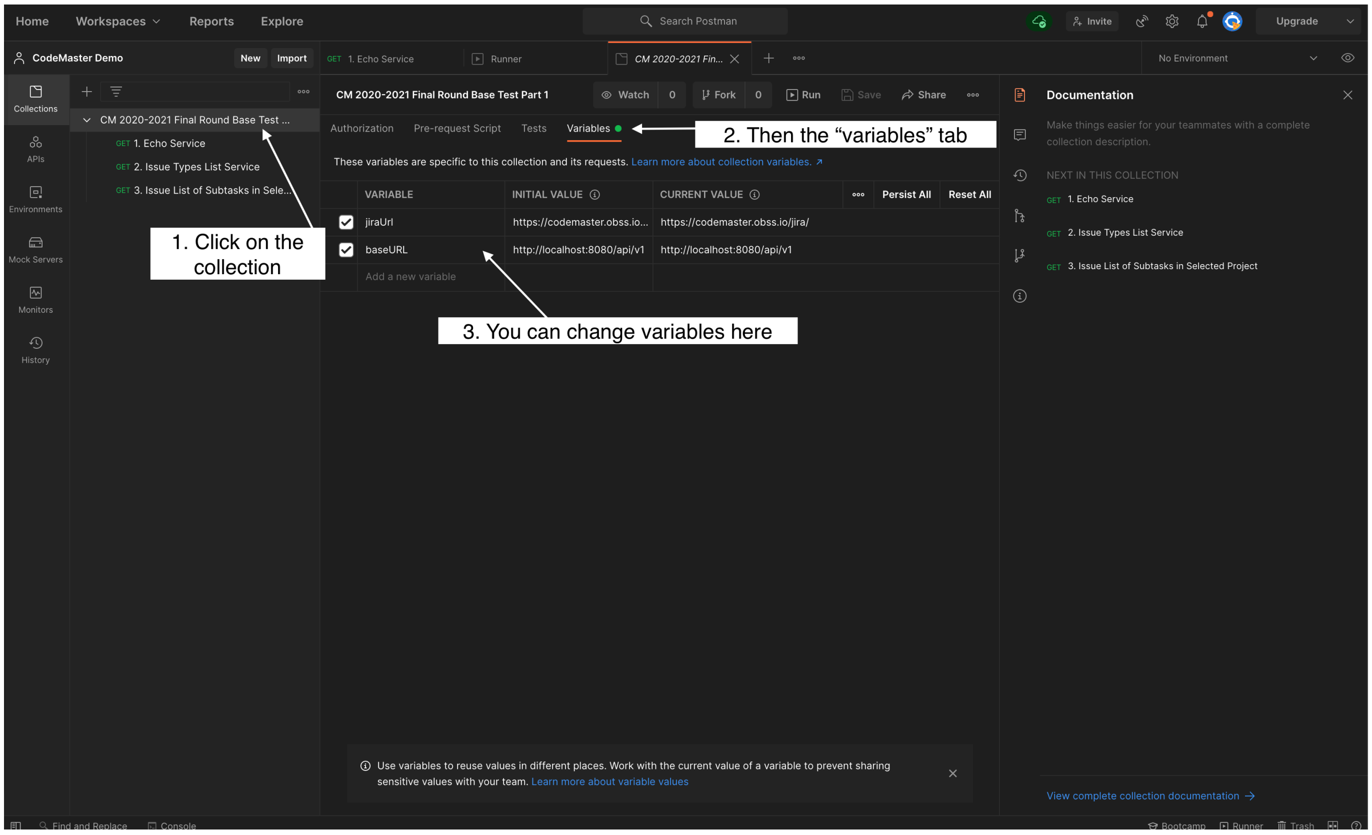
Finally click this button and tests will start running

6. You can also run tests task by task. It is described below.



```
test
```

7. You can change environment variables. It is described below.



These variables are specific to this collection and its requests. Learn more about collection variables. ↗

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	jiraUrl	https://codemaster.obss.io...	https://codemaster.obss.io/jira/
✓	baseURL	http://localhost:8080/api/v1	http://localhost:8080/api/v1
	Add a new variable		

Make things easier for your teammates with a complete collection description.

Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values

baseURL: http://localhost:8080/api/v1
jiraUrl: https://codemaster.obss.io/jira/