1. **Overview**

Purpose of this document is to provide information regarding the development task given to me as Programming Example Task document. Some of the line items are still pending, but can be provided subsequently.

* 1. Rest services, /math/add (GET), /math/add (POST), /time/now (GET) services are completed. Please refer to technical specification section for more details on Overview & Usage, Process flow diagrams and Error Handling.

1.2 Services are hosted on Google cloud service can be access using host name as **www.thiyon.ca** or using ip address **35.197.42.183.**

Eg: <http://www.thiyon.ca/math/add?p1=100&p2=20.7>, <http://www.thiyon.ca/time/now/YYC>

1.3 Testing – There are 2 Postman (JSON) files are attached into my email. Since the Eclipse project also attached into the same email, if you would like to use a local deployment (local host on port 8080), please use postman file “**Math Add.postman\_collection.json**” for your testing. Otherwise, you can use “**Math Add (Thiyon).postman\_collection.json**” to test on cloud.

1. **Development Tools & Environments**
   1. IDE : Eclipse (Neon)
   2. Java : JDK 1.8\_0.73
   3. Resource dependency management tool : Maven
   4. Unit test tool : Postman
   5. Cloud environment : Google cloud services (External IP : 35.197.42.183)
2. **Q&A**
   1. Data caching – In spring, data caching can be enabled with @Cacheable() annotation with app level annotation @EnableCaching. @EnableCaching annotation will cause the application to search through public methods with @Cacheable annotation and start intercepting during method call and provide cached information.

Pros – In case where requesting data set is mostly static information and need DB querying or external resource access (which adds latency), it is a good idea to cache such information within the local system to improve system performance. Another advantage of caching is to provide accessibility to information when initial source is not accessible. As an example, you may have already notices that our demo application return errors as error codes, in case if you want to convert such error codes into proper error messages (may or may not be based on language preference), such list of error messages can stored in a DB but can be cached within app for later mapping.

Cons – Cached information can be stale over time. In such case, appropriate processes need to be in place for cache to be refreshed.

3.4 Security Choices – Our demo services are not using https (secure socket layer). As a result, information transfers between the client (browser/postman) and server can be extracted and viewed if required. To enable https on tomcat we need to configure a SSL connector and SSL certificate. Passwords/keys (x-apikeys) should be encrypted before store in DBs. Encryption can be done using Spring Security framework.

1. **Development task exceptions**
   1. Accessing [www.developer.aero](http://www.developer.aero) – In order to get an x-apikey for /time/now API development. I did create an account with SITA (UID : swoop.test). As per SITA instruction, I need to create an application entry within SITA for me to obtain a x-apikey, which required to contact [ashish.kapoor@sita.aero](mailto:ashish.kapoor@sita.aero). Did send out an email but there is no reply yet. Meanwhile, based on the success and un-successful samples JSONs provided in SITA site. I did create a local rest API within same project and redirecting external API call to itself.
   2. GitHub – I do not own a GitHub account, as a result source code and Eclipse package will be included in my email.
   3. Docker file – I could not get a chance to work on the docker file due to time limitation.

1. **Technical Specifications**
   1. **REST Service GET - /math/add**
      1. **Overview and Usage**

Purpose of this service is to add 2 decimal numbers using REST API as a GET operation. Client application must provide 2 numerical values as parameters for this API to return sum of given 2 values. 2 values must be specified as path variables and variable names must be labeled as p1 & p2. Following is a sample API call.

URL - http://localhost:8080/math/add?p1=5.7867886&p2=12.6756765

In a success scenario API will return the response with sum of above 2 values similar following JSON document.

{

"total": "18.4624651",

"operation": "ADD",

"error\_codes": [],

"status": "STS\_SUCCESS",

"errorFound": false,

"parameter\_2": "12.6756765",

"parameter\_1": "5.7867886"

}

As highlighted above, there are no errors found during the execution and status is a “STS\_SUCCESS” and sum of parameter 1 & 2 is provided as total.

API execution failures can happen due to multiple reasons such as missing parameters, invalid numbers, etc. Following a sample JSON document which illustrate a fail scenario. In this case the client did not provide the second parameter,

URL - http://localhost:8080/math/add?p1=5.7867886

API JSON response

{

"total": null,

"operation": "ADD",

"error\_codes": [

"ERR\_P2\_002"

],

"status": "STS\_FAILED",

"errorFound": true,

"parameter\_2": null,

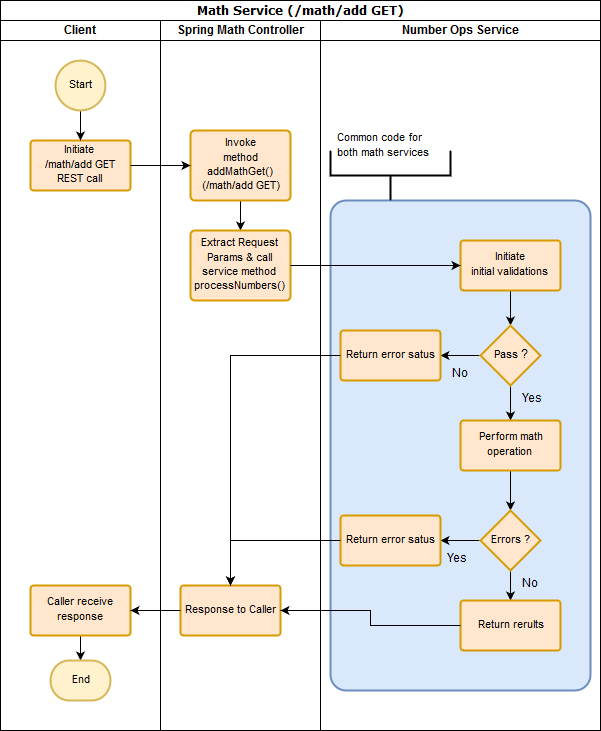
"parameter\_1": "5.7867886"

}

Error codes are an array and can provide more than 1. Please refer to Error handling section for all possible error codes.

* + 1. **Process flow**

Following diagram illustrate the process flow of /math/add (GET) API. Section in blue is common code for both /math/add GET and POST APIs.



* + 1. **Error Handling**

Following is a list of possible error status codes and error codes can be generated by the API. API will not provide the description for the message. Error code to description conversion need to happen separately with user language preference.

|  |  |  |
| --- | --- | --- |
| Status Code | Error Code | Description |
| STS\_FAILED | ERR\_NULL\_OBJ | Null math operation requested |
| STS\_FAILED | ERR\_P1\_001 | Parameter 1 is missing |
| STS\_FAILED | ERR\_P2\_001 | Parameter 2 is missing |
| STS\_FAILED | ERR\_P1\_002 | Parameter 1 is invalid number |
| STS\_FAILED | ERR\_P2\_002 | Parameter 2 is invalid number |
| STS\_FAILED | ERR\_BAD\_OP | Unspecified match operation (\* for future) |

* 1. **REST Service POST - /math/add**
     1. **Overview and Usage**

Purpose of this service is to add 2 decimal numbers using REST API as a POST operation. Client application must provide 2 numerical values as part of request body for this API to return sum of given 2 values. 2 values must be specified as request body as a JSON and variable names must be labeled as parameter\_1 & parameter\_2 and operation code must set as ADD. Following is a sample API call.

URL - http://localhost:8080/math/add

Request Body (parameter 1 & 2 can be specified as a numerical or non-numerical form),

{

"operation": "ADD",

"parameter\_1": 23.45645646,

"parameter\_2": "45.678678"

}

Same as in /math/add GET operation, in a success scenario API will return the response with sum of above 2 values similar following JSON document format.

{

"total": "69.13513445999999",

"operation": "ADD",

"error\_codes": [],

"status": "STS\_SUCCESS",

"errorFound": false,

"parameter\_2": "45.678678",

"parameter\_1": "23.45645646"

}

As highlighted above, there are no errors found during the execution and status is a “STS\_SUCCESS” and sum of parameter 1 & 2 is provided as total.

API execution failures can happen due to multiple reasons such as missing parameters, invalid numbers, etc. Following a sample JSON document which illustrate a fail scenario. In this case the client did not provide non numerical values as param 1 & 2,

URL - http://localhost:8080/math/add

{

"operation": "ADD",

"parameter\_1": "23.4564wer5646",

"parameter\_2": "45.67wer8678"

}

API JSON response

{

"total": null,

"operation": "ADD",

"error\_codes": [

"ERR\_P1\_002",

"ERR\_P2\_002"

],

"status": "STS\_FAILED",

"errorFound": true,

"parameter\_2": "45.67wer8678",

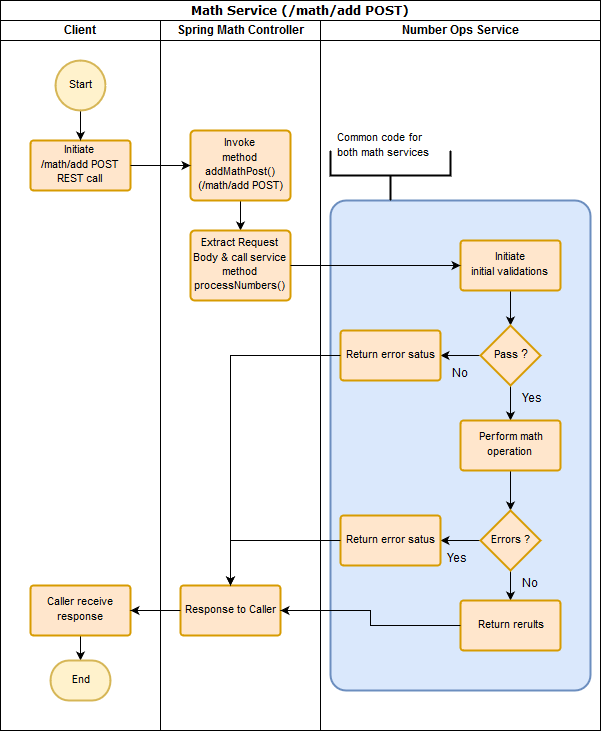
"parameter\_1": "23.4564wer5646"

}

Error codes are an array and can provide more than 1. Please refer to Error handling section for all possible error codes.

* + 1. **Process flow**

Following diagram illustrate the process flow of /math/add (GET) API. Section in blue is common code for both /math/add POST and GET APIs.



* + 1. **Error Handling**

Following is a list of possible error status codes and error codes can be generated by the API. API will not provide the description for the message. Error code to description conversion need to happen separately with user language preference.

|  |  |  |
| --- | --- | --- |
| Status Code | Error Code | Description |
| STS\_FAILED | ERR\_NULL\_OBJ | Null math operation requested |
| STS\_FAILED | ERR\_P1\_001 | Parameter 1 is missing |
| STS\_FAILED | ERR\_P2\_001 | Parameter 2 is missing |
| STS\_FAILED | ERR\_P1\_002 | Parameter 1 is invalid number |
| STS\_FAILED | ERR\_P2\_002 | Parameter 2 is invalid number |
| STS\_FAILED | ERR\_BAD\_OP | Unspecified match operation (\* for future) |

* 1. **REST Service GET - /time/now/{airport\_code}**
     1. **Overview**

Purpose of this service is to get local time at given airport with time zone using a REST API as a GET operation. Client application must provide airport code value as path variable for this API to return local time and time zone. API will call external API (GET operation) to retrieve data.

https://www.developer.aero/waittime/v1/current/{airportCode}

Response from external call will also be embedded within JSON response.

Following is a sample API call.

URL - http://localhost:8080/time/now/YYC

In a success scenario API will return the response with time and time zone values similar to following JSON document.

{

"status": "STS\_SUCCESS",

"zone": "-07:00",

"airportCode": "YYC",

"currentTime": "2017-11-07T00:05:15.000Z",

"localTime": "2017-11-07T10:15:00.000-07:00",

"sourceResponse": {

"success": true,

"current": [

{

"airportCode": "YYC",

"airportName": "Calgary International Airport",

"queueId": "4dbe38e82f245f3ce8c347d6d29184df",

"queueName": "South General",

"projectedWaitTime": 960,

"projectedMinWaitMinutes": 14,

"projectedMaxWaitMinutes": 18,

"localTime": "2017-11-07T10:15:00.000-07:00",

"time": "2017-11-07T00:05:15.000Z"

}

]

},

"errorType": null

}

As highlighted above, there are no errors found during the execution and status is a “STS\_SUCCESS”

API execution failures can happen due to multiple reasons such as missing parameters, communication errors, etc. Following a sample JSON document which illustrate a fail scenario. In this case external API service was not available,

URL - http://localhost:8080/time/now/YYC

API JSON response

{

"zone": null,

"status": "STS\_FAILED",

"currentTime": null,

"airportCode": "YYC",

"localTime": null,

"errorType": "ERR\_TYPE\_SITA\_EX",

"sourceResponse": {

"errors": {

"error": [

{

"description": "no service available",

"code": "1003"

}

]

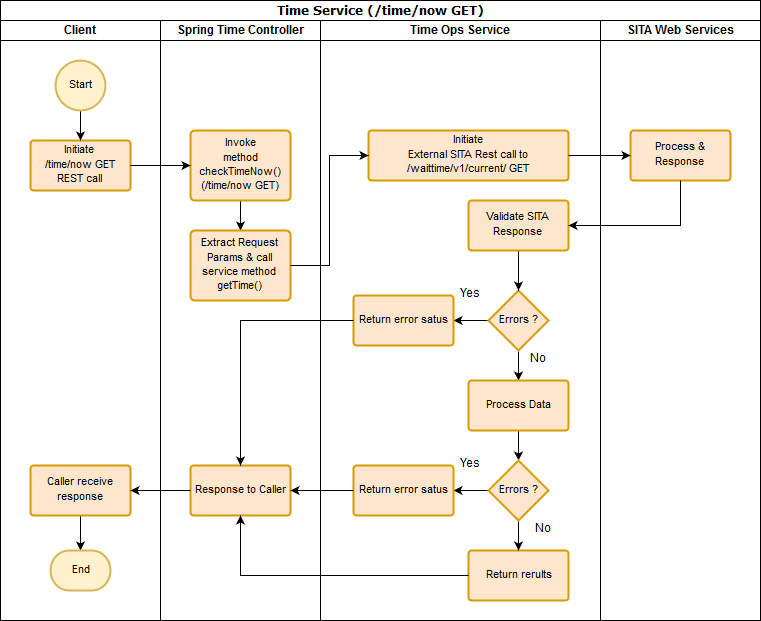
},

"success": false

}

}

* + 1. **Process flow**



* + 1. **Error Handling**

Following is a list of possible error status codes and error type codes can be generated by the API. API will not provide the description for the message. Error type code to description conversion need to happen separately with user language preference.

|  |  |  |
| --- | --- | --- |
| Status Code | Error Type | Description |
| STS\_FAILED | ERR\_TYPE\_COMM\_EX | Sample Msg on sourceResponse:  {     "status":"failed",     "httperrorCode":"404" } |
| STS\_FAILED | ERR\_TYPE\_SITA\_EX | Sample Msg on sourceResponse:  {  "errors": {  "error": [  {  "description": "authentication failed",  "code": 9998  }  ],  "statusCode": 403  },  "success": false  } |
| STS\_FAILED | ERR\_TYPE\_SITA\_EX | Java exception message will be indicated on sourceResponce |