POSTMAN- 10.6.7

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Workspace – Empty space to add projects

Collections in postman- To group API together and hold it in collection.

We need to ask API Contract to developers before starting API testing.

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

5 types of variables Scopes/

Global

Collection

Environment



Data

Local

Local variable has higher preference than global.

**Order of execution of postman requests**

Postman has inbuilt node.js we can do JS code inside it.

Pre Test scripts-> Request-> Response-> Test Scripts (for assertion and validation.)

The pm object- To call any API functionality to access request, response data of anything of api.

Pm.test takes 2 arguments- Description and Function()

// assertions and validations

pm.test("To check status code is 200", *function*()

{

})

// assertions and validations

*const* jsondata = pm.response.json();

pm.test("To check status code is 200", *function*()

{

  pm.response.to.have.status(200);

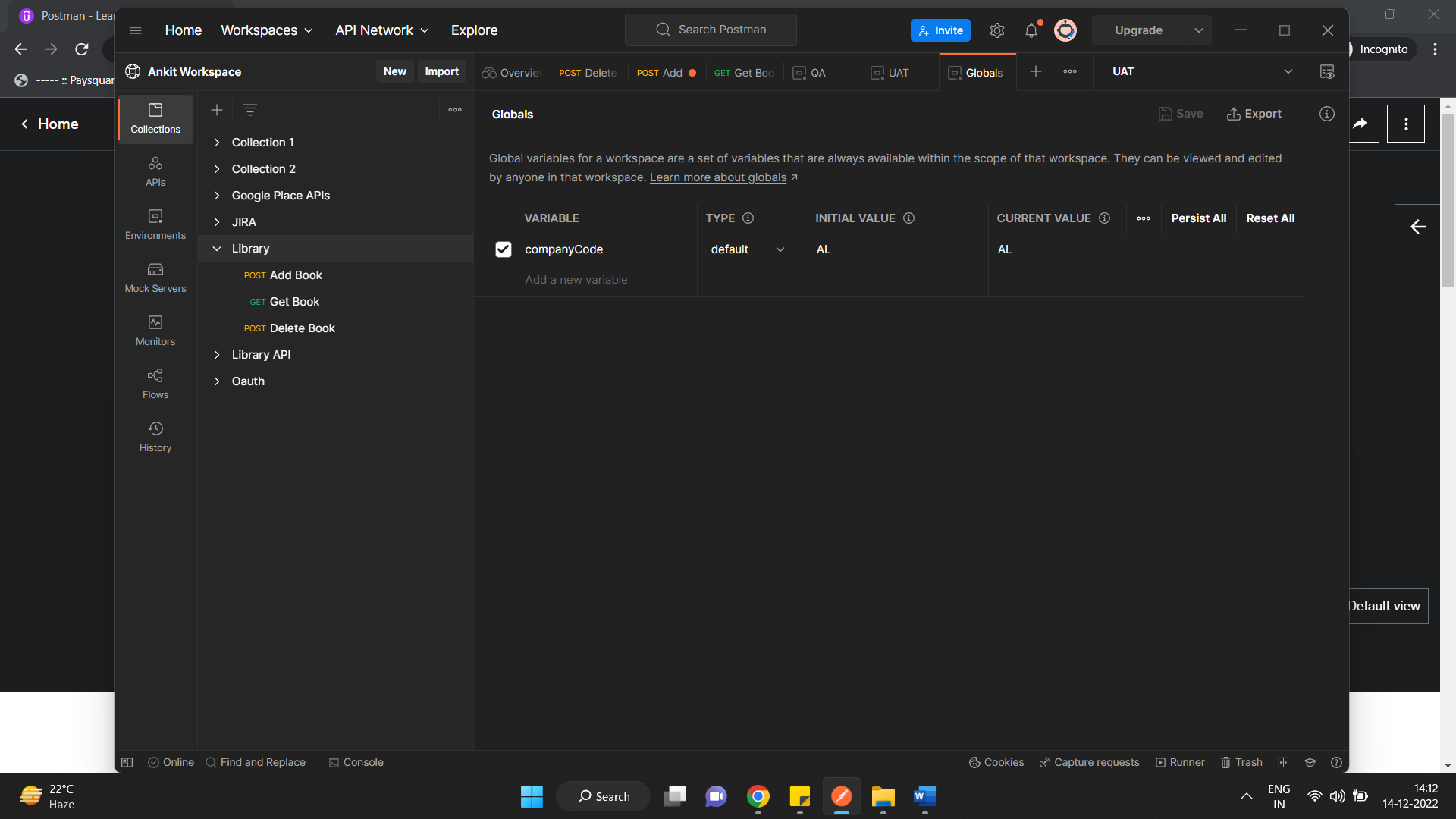
  pm.expect(jsondata).have.property("Msg");

  pm.expect(jsondata.Msg).to.equal("successfully added");

})

How to dynamically generate values from variables into input Payload requests

In pre request script we will create dynamic payload



So here go to env level and set a global variable e.g company code and call this in pre scripts to generate dynamic body

// RS23445 TO CREATE ISBN DYNAMICALLY into body

*console*.log(pm.globals.get("companyCode"));

*const* code = pm.globals.get("companyCode");

*const* val = pm.variables.replaceIn('{{$randomInt}}');

pm.collectionVariables.set("isbn", code + val);

{

"name":"Learn Appium Automation with Java",

"isbn":"*{{isbn}}*",

"aisle":"222483",

"author":"John foe"

}

Chai Assertions in Postman and how to use that in Scripting window

pm.test("Headers cookies response validation", *function*()

{

  pm.response.to.have.header("Content-Type");

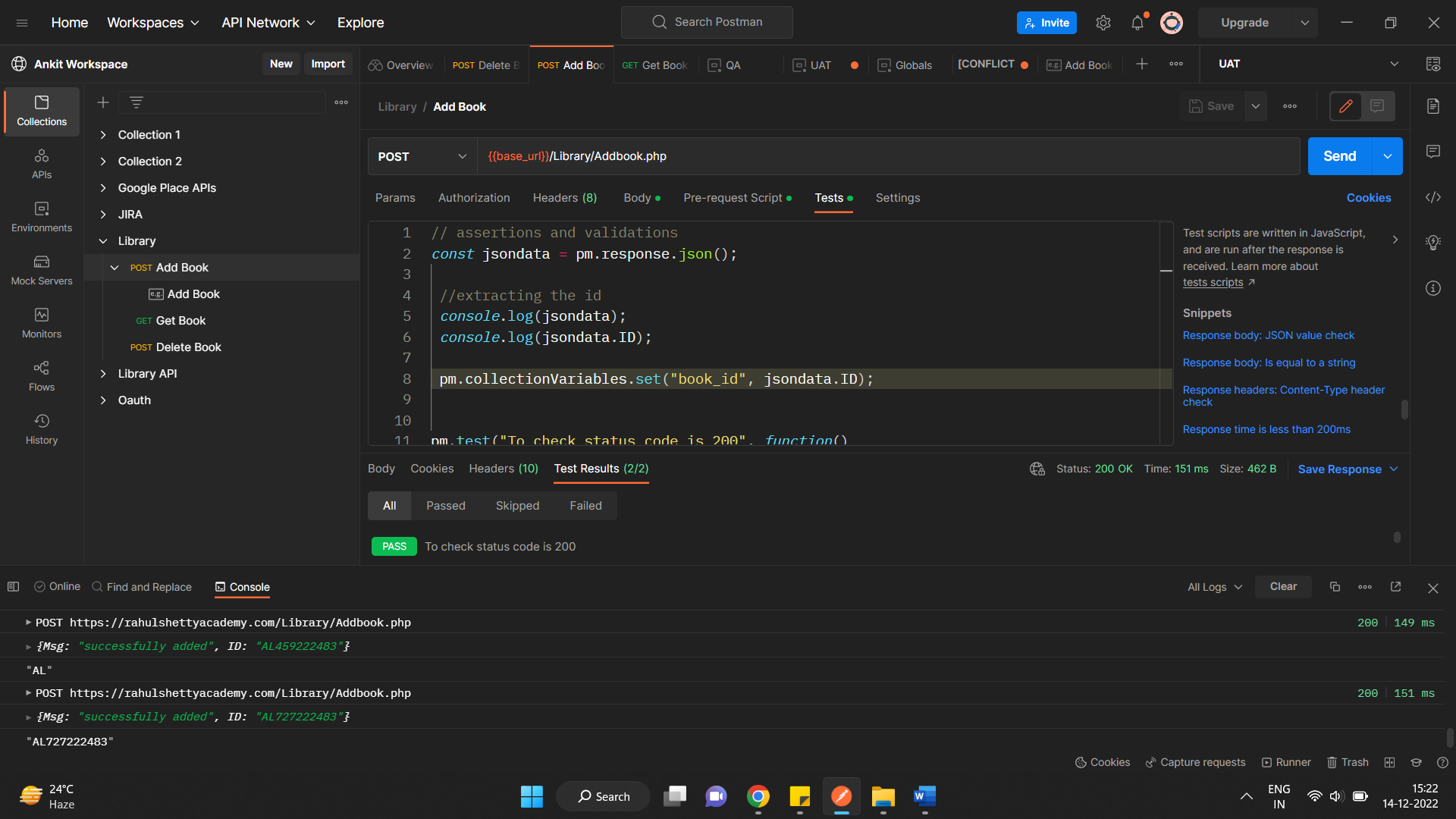
  pm.expect(pm.response.headers.get("Content-Type")).to.equal("application/json;charset=UTF-8");

  pm.expect(pm.response.responseTime).to.be.below(2000);

  pm.expect(pm.response.text()).to.include("successfully added");

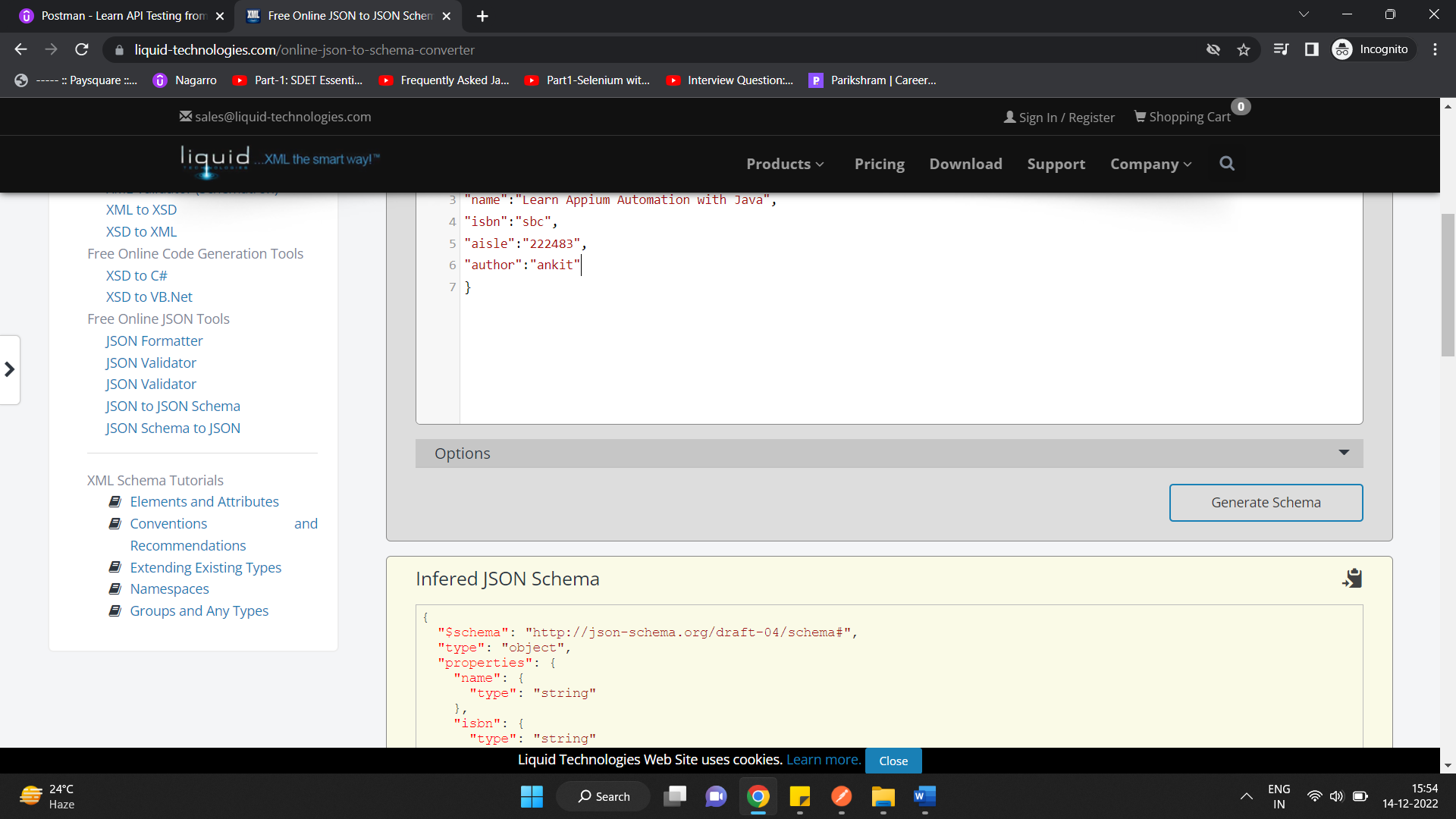
});

Running set of tests together in the Collection as automated runner



To validate the Get response with the body send in post in Tests

https://www.liquid-technologies.com/online-json-to-schema-converter



A screenshot of a computer

Description automatically generated with medium confidence

Validating JSON SCHEMA

*console*.log(pm.response.json());

*const* bookdata= pm.response.json()

*const* schema= {

  "type": "array",

  "items": [

    {

      "type": "object",

      "properties": {

        "name": {

          "type": "string"

        },

        "isbn": {

          "type": "string"

        },

        "aisle": {

          "type": "string"

        },

        "author": {

          "type": "string"

        }

      },

      "required": [

        "name",

        "isbn",

        "aisle",

        "author"

      ]

    }

  ]

};

pm.test("validating Json response logic", *function*()

{

    pm.expect(bookdata[0].author).to.equal(pm.collectionVariables.get("author\_name"));

});

pm.test("validating Json Schema logic", *function*()

{

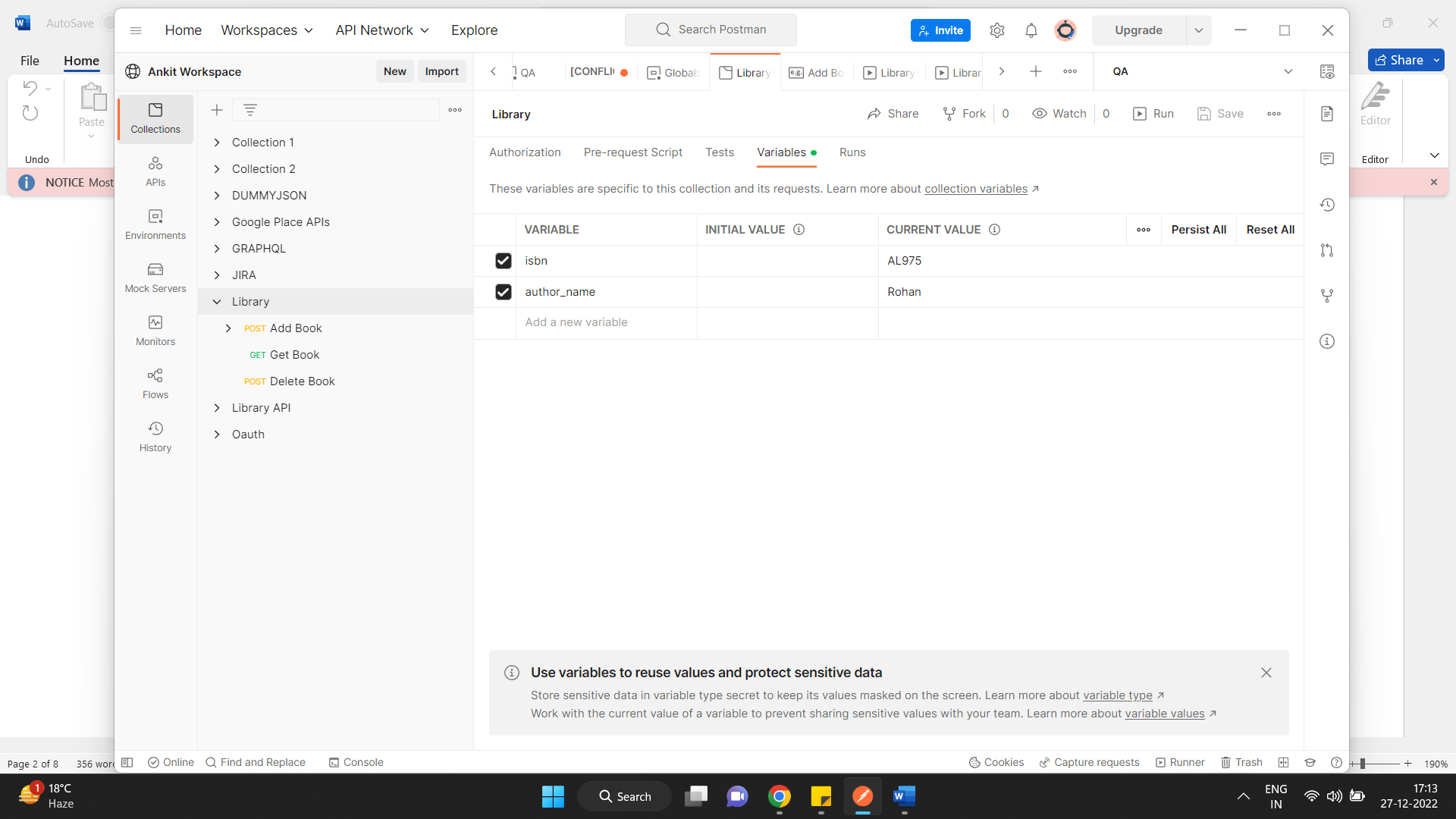
 pm.response.to.have.status(200);

 pm.response.to.have.jsonSchema(schema);

});

**DATADRIVEN TESTING USING POSTMAN**

**Accessing request body to validate/assert. As postman do not have request.json() function to retrieve json data from request just like we had in for response.json() we will extract raw from body and parse it using JSON Parser**



// To validate contactination of ID and aisle and assert it with response.

pm.**test**("ID Logic validation", **function**()

{

 //How to get isbn

const isbn\_val **=** pm.collectionVariables.**get**("isbn");

var request\_id **=**JSON.**parse**(pm.request.body.raw);

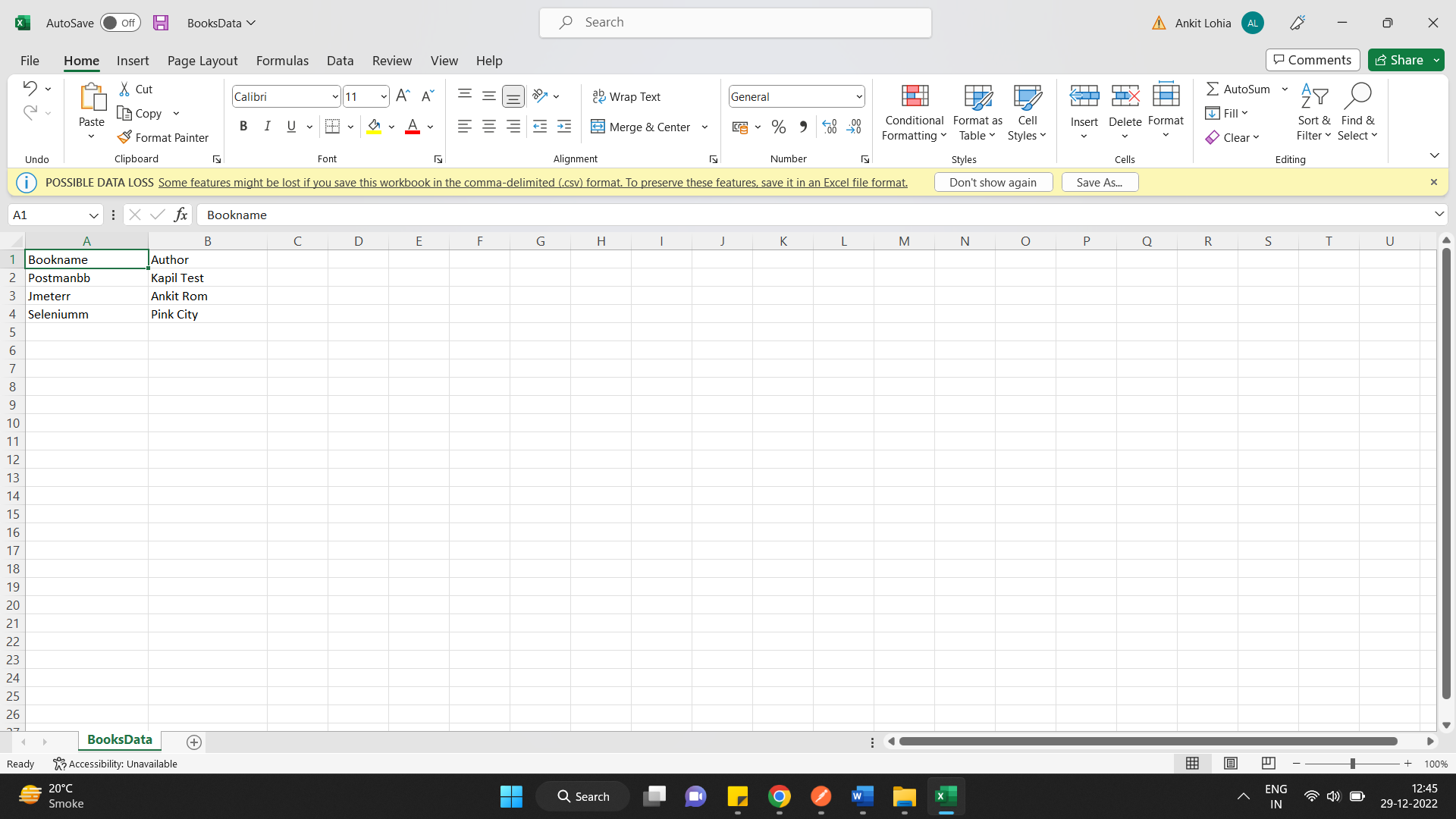
const aisle\_val**=** request\_id.aisle;

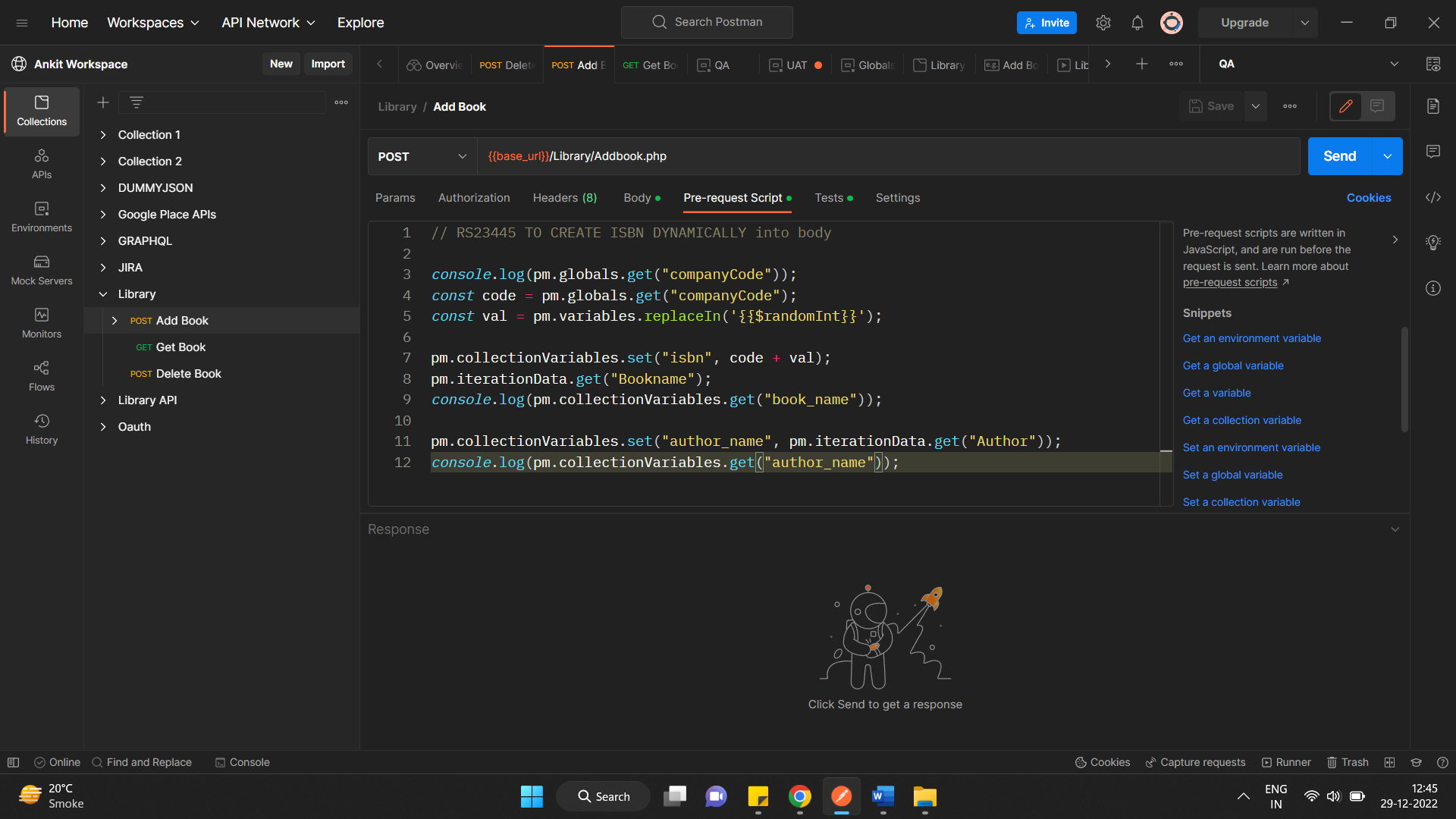
const expected\_id **=** isbn\_val **+** aisle\_val;

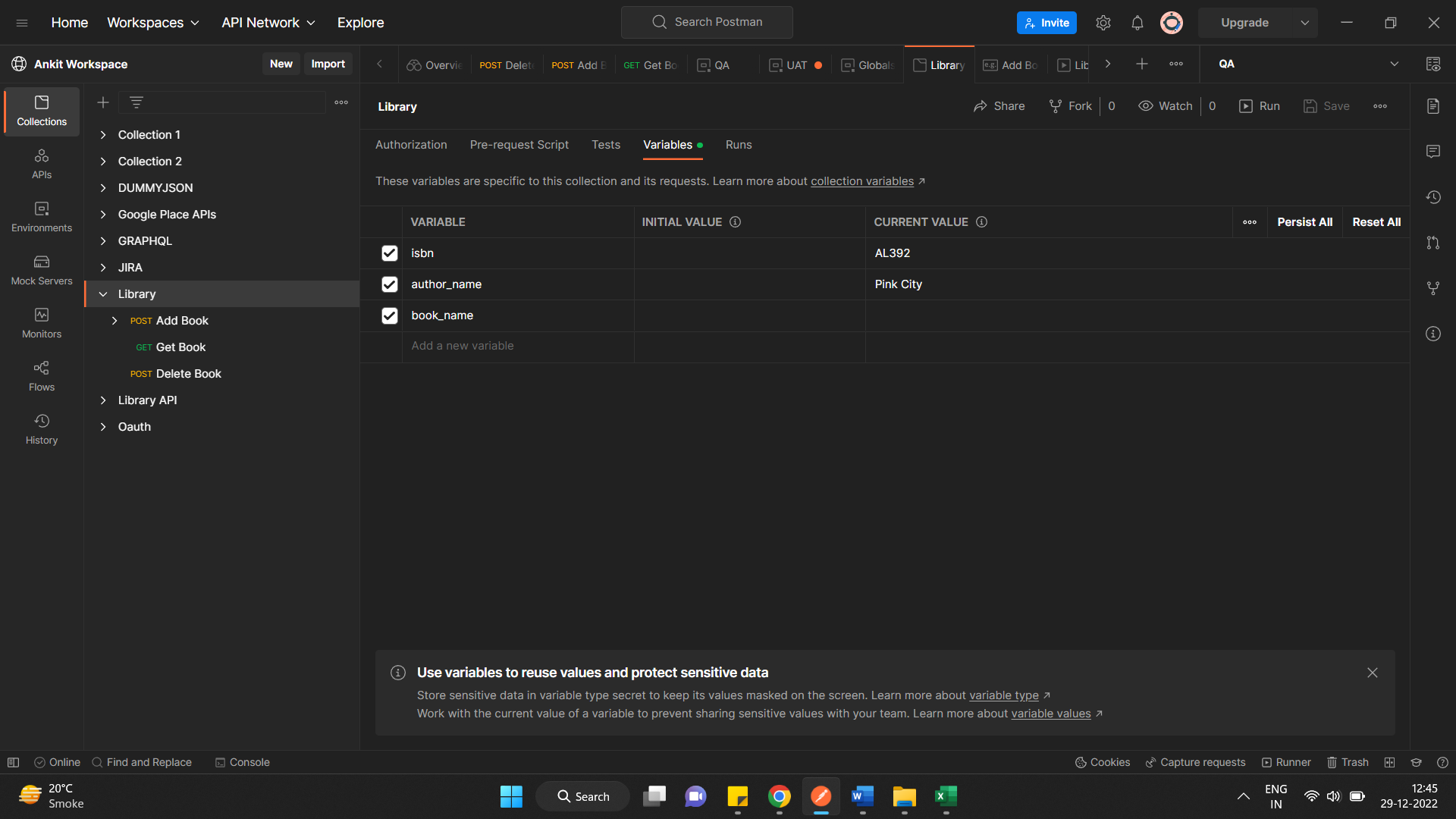
pm.expect(expected\_id).to.equal(book\_id\_val);

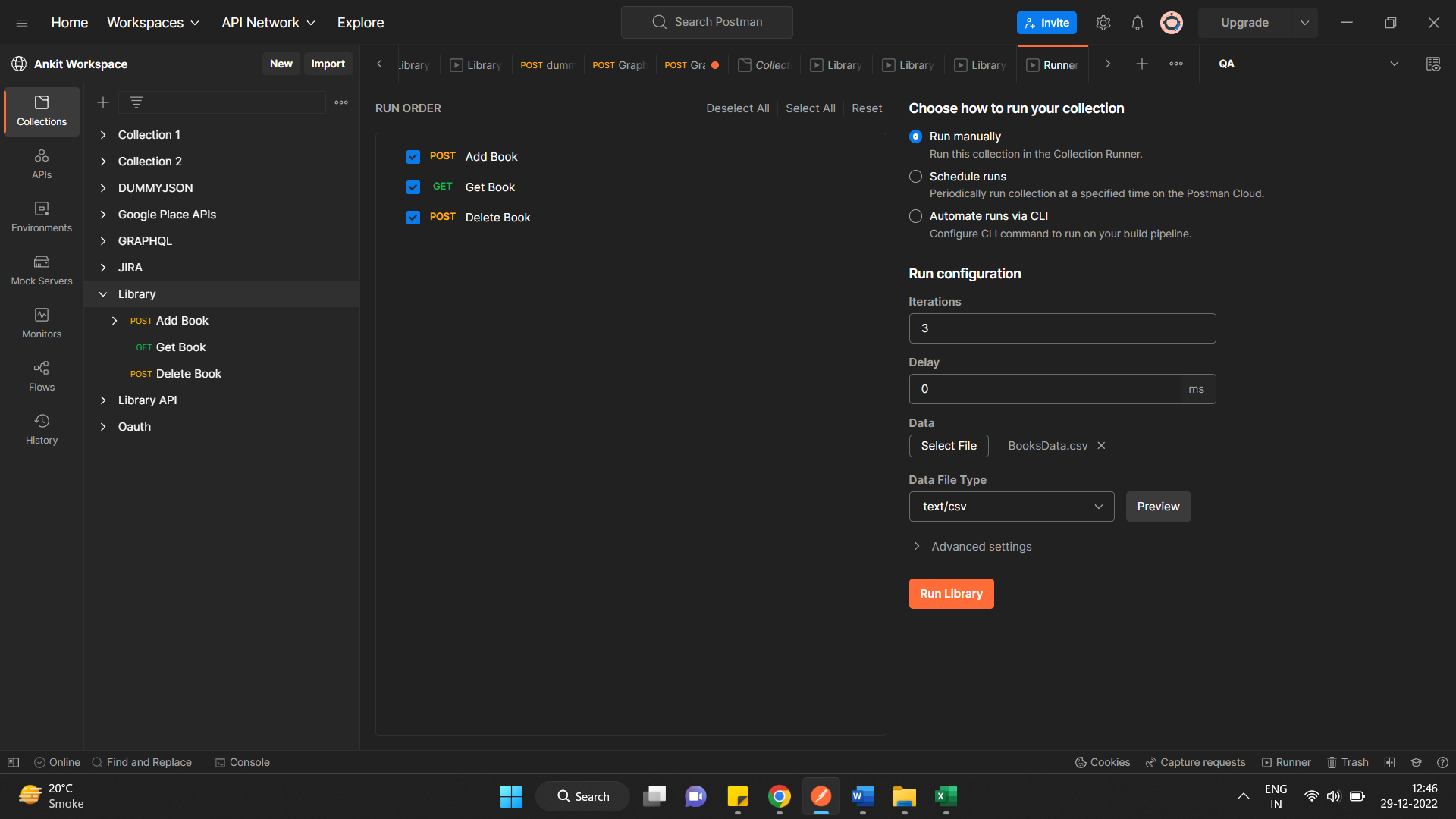
});

How to perform Data driven testing from CSV to Postman requests Scripts

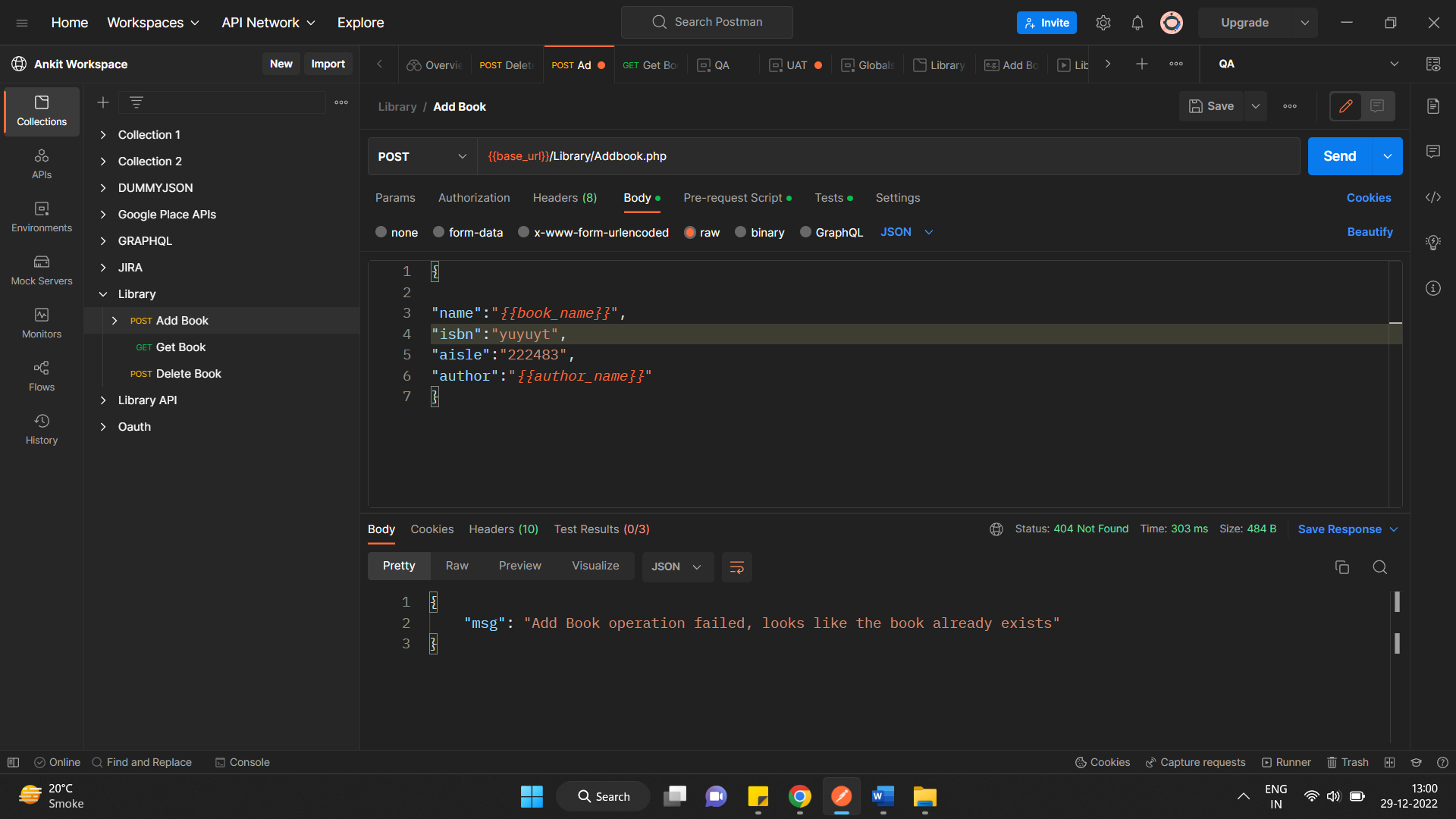


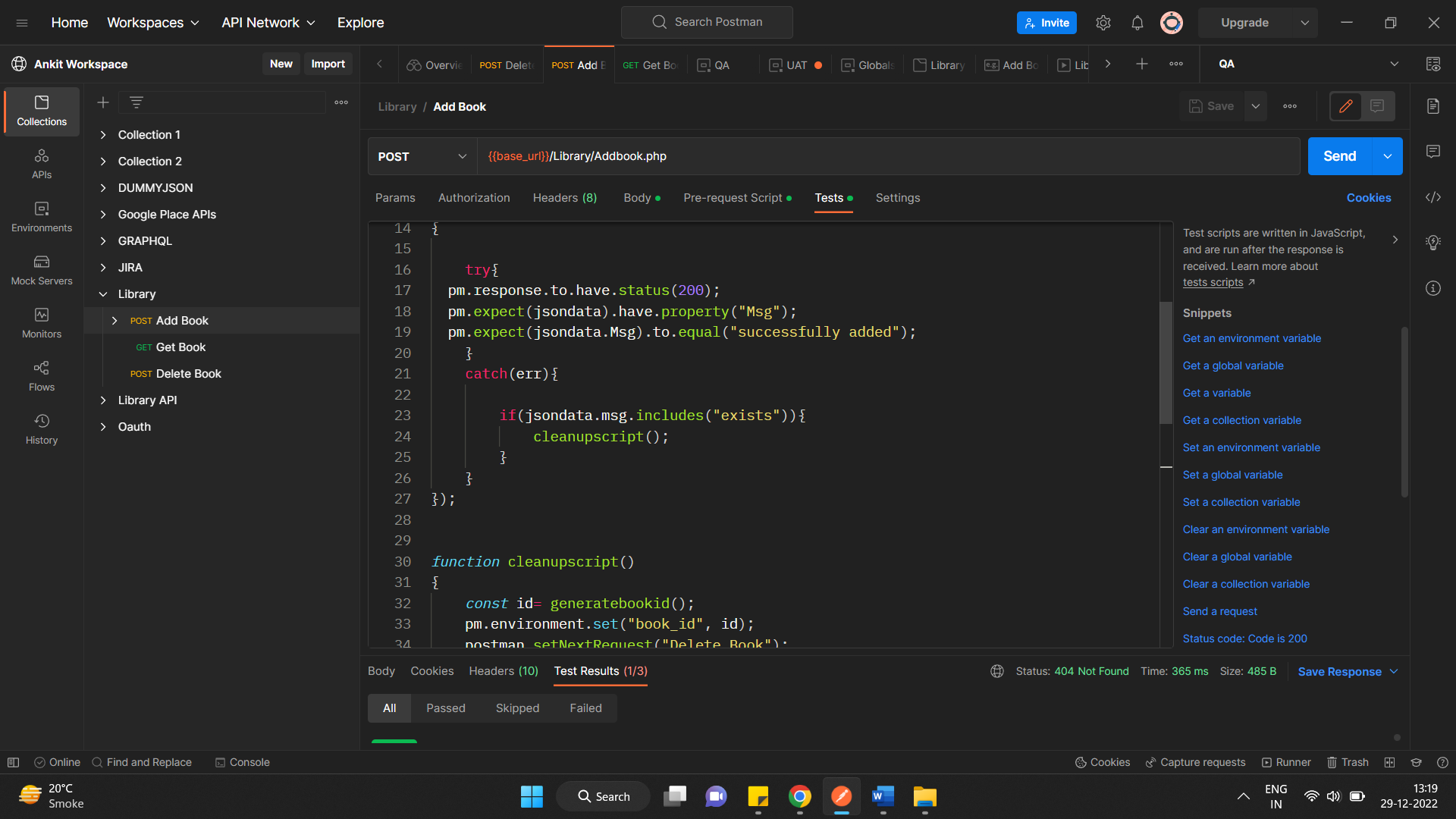






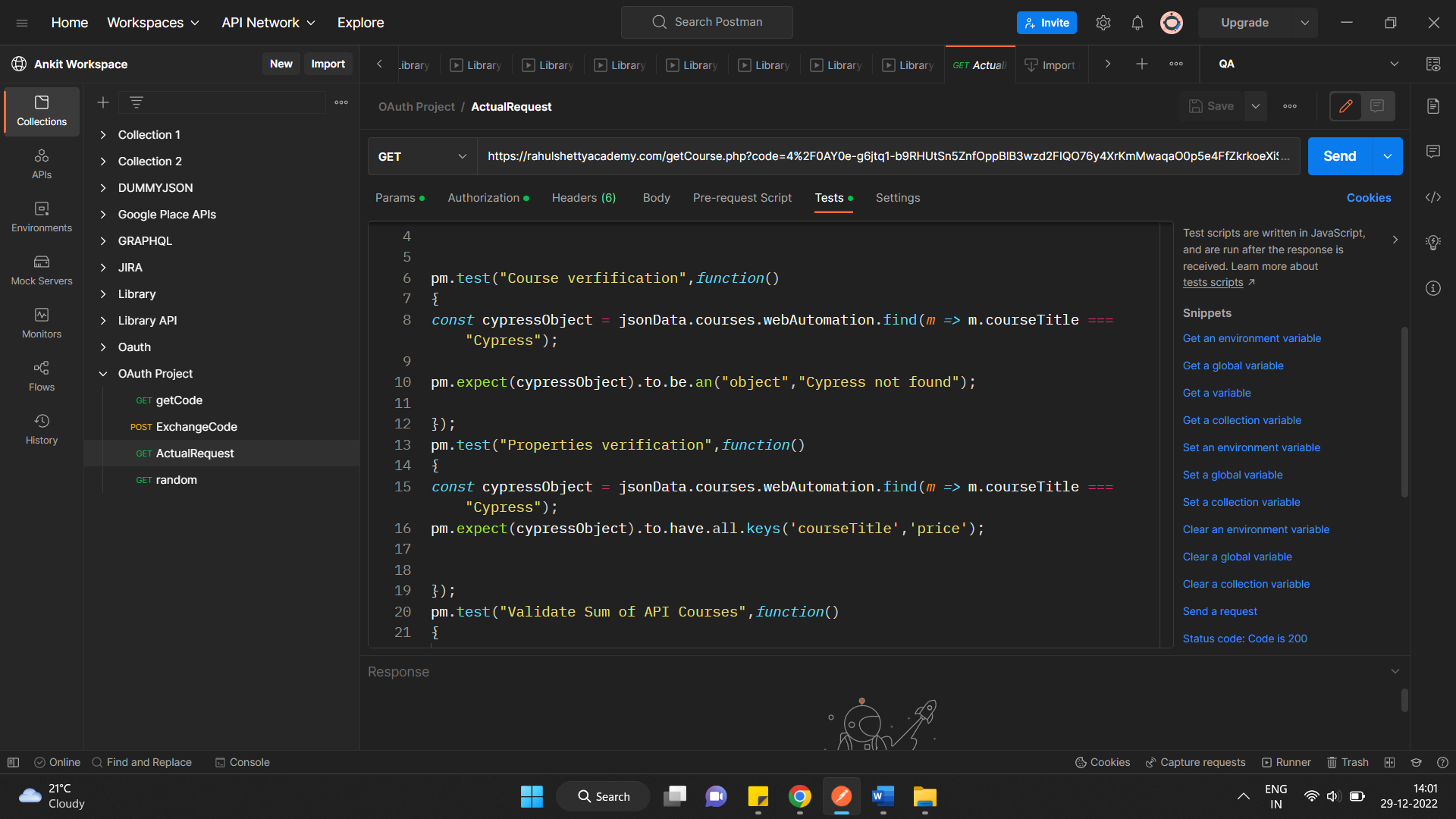
How to handle Error handling Scenarios in API Tests with Postman







. How to parse Complex Nested Json response through Json object



**To check keys in nested json visa postman**

pm.expect(cypressObject).to.have.all.keys('courseTitle','price');

Here course title and price are keys

**How to push elements in the array**

**Use Push keyword**

*const* expectedCourses = ["Selenium Webdriver Java","Cypress","Protractor"];

*const* actualCourses = [];

pm.test("Validate Web Course Titles",*function*()

{

    jsonData.courses.webAutomation.map(*m=>* actualCourses.push(m.courseTitle));

*console*.log(actualCourses);

    pm.expect(expectedCourses).to.be.eql(actualCourses);

});

**JSON SCHEMA VALIDATIONS USING POSTMAN**

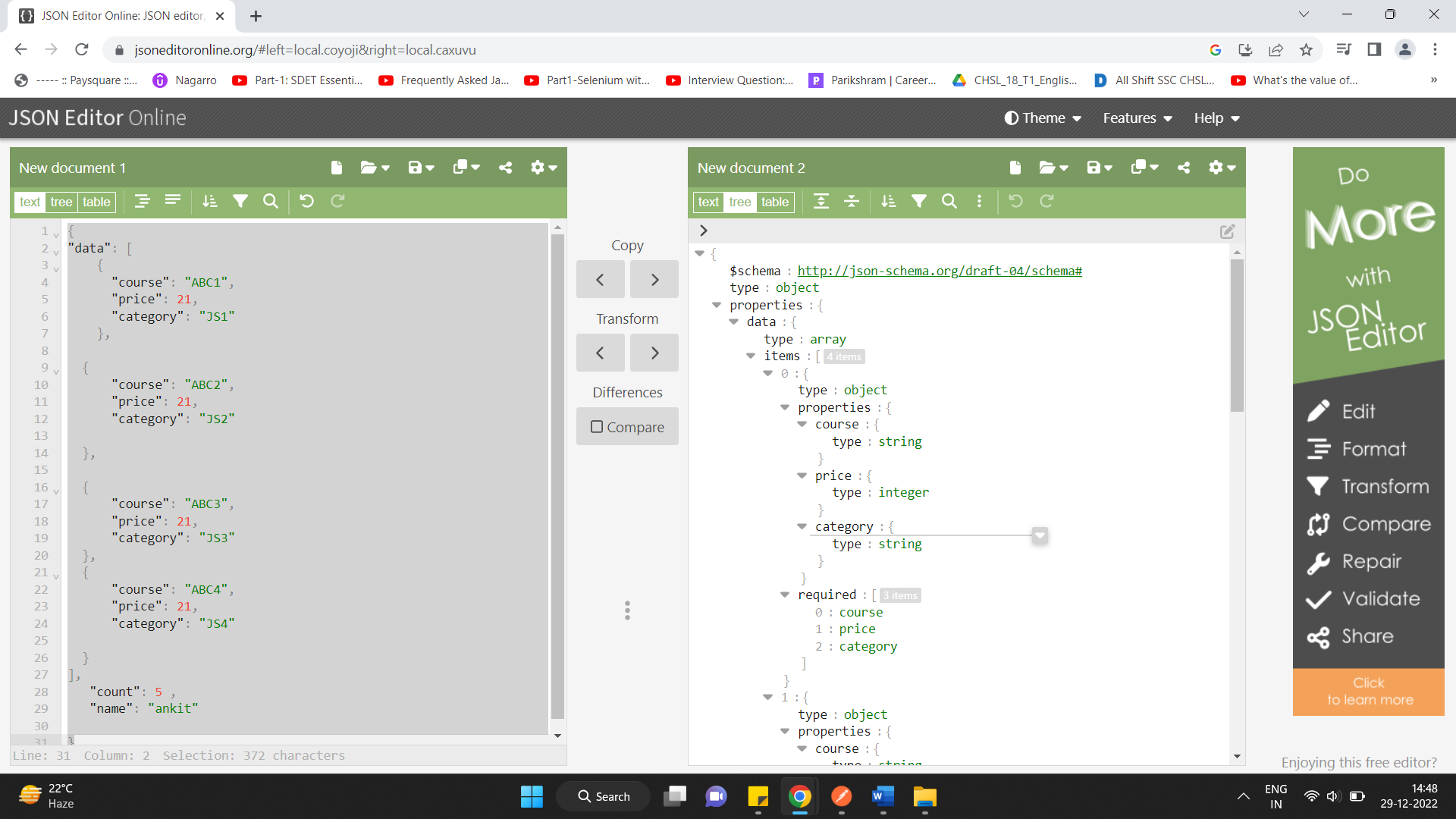
**Json schema is raw structure where we have types in values like strung , int while api development is in progress**

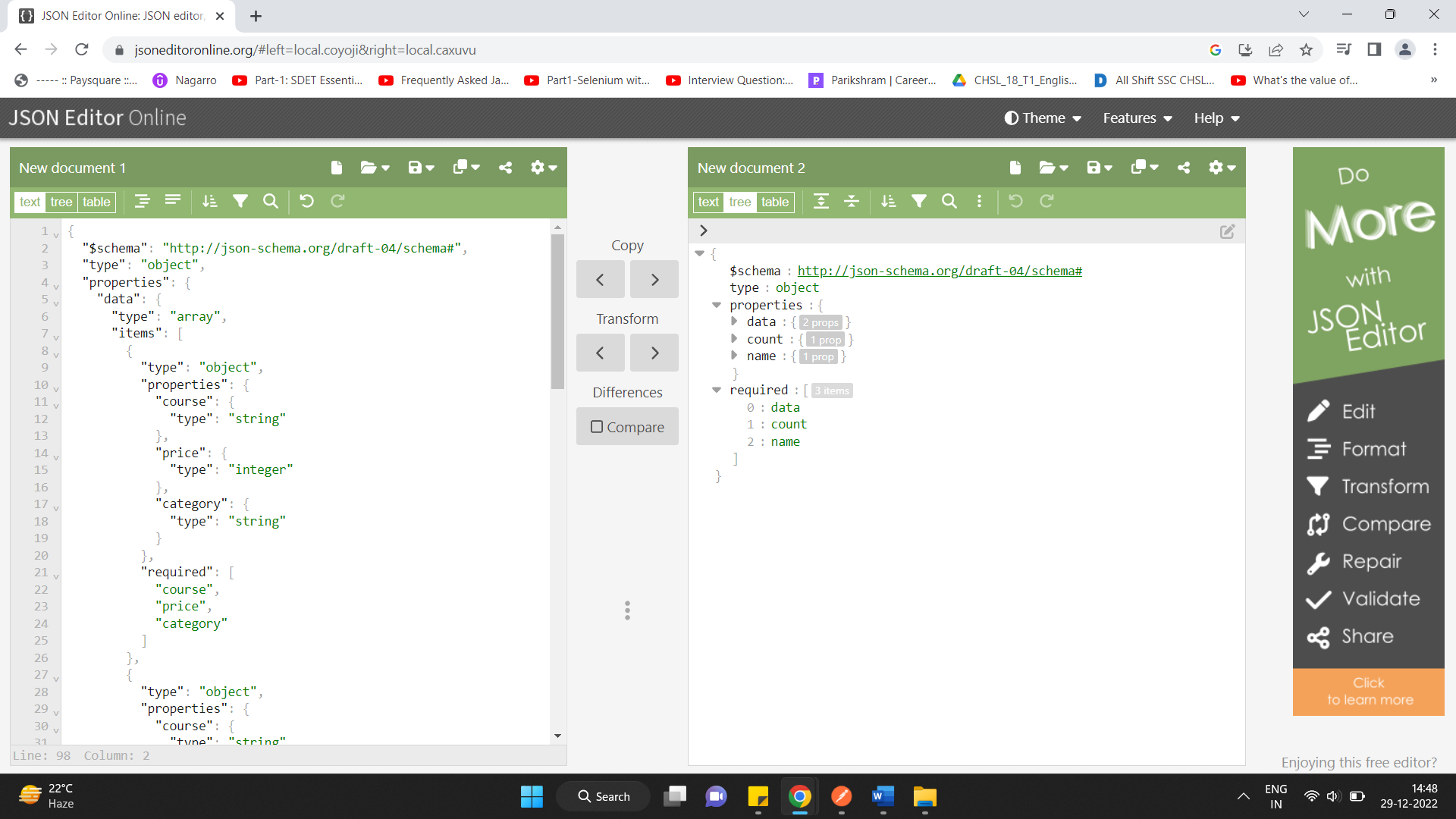
[**https://www.liquid-technologies.com/online-json-to-schema-converter**](https://www.liquid-technologies.com/online-json-to-schema-converter)

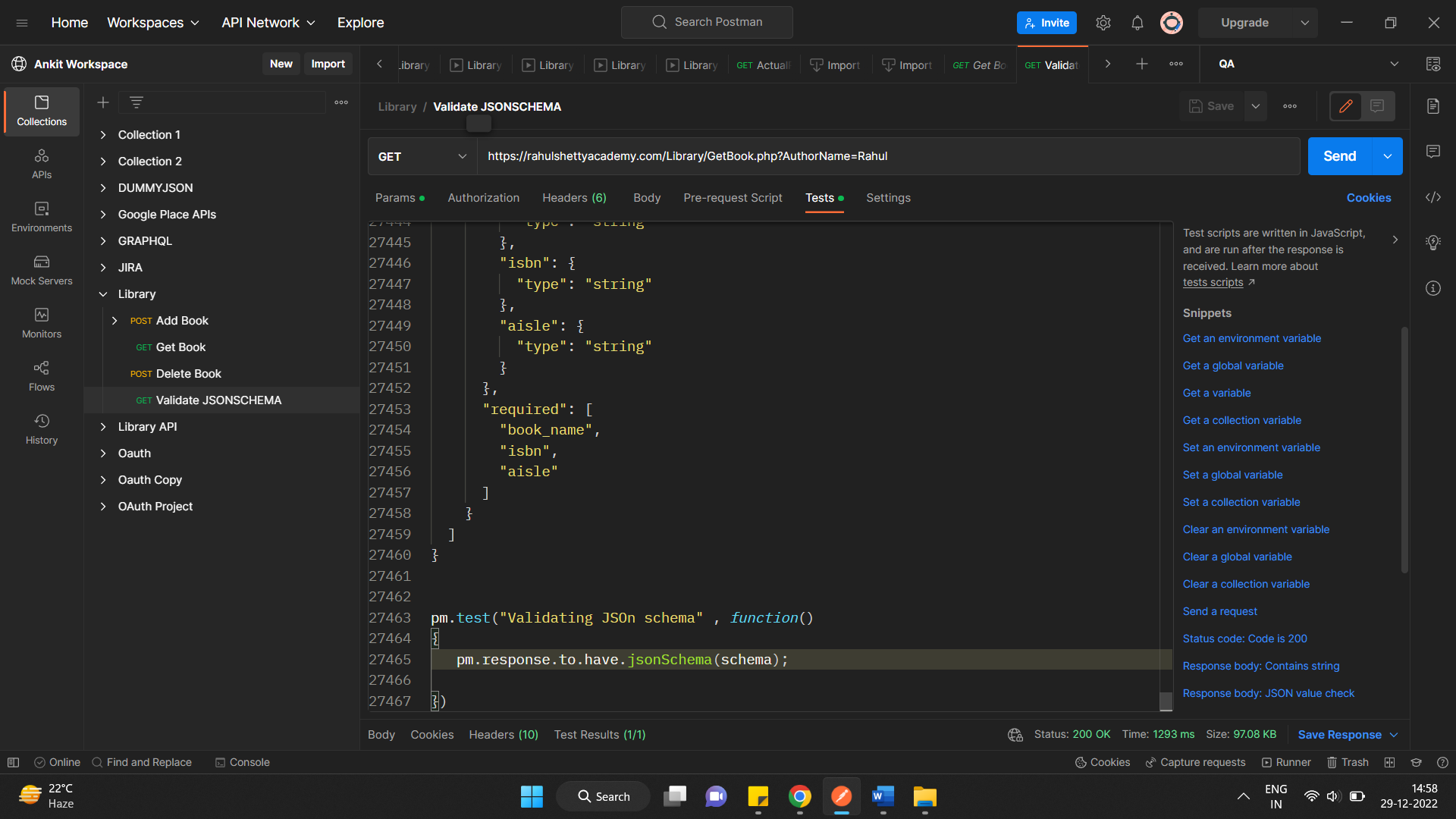
**Use above website to convert JSONSCHEMA TO JSON OR vice versa**

**OR USE**

[**https://jsoneditoronline.org/#left=local.coyoji&right=local.caxuvu**](https://jsoneditoronline.org/#left=local.coyoji&right=local.caxuvu)

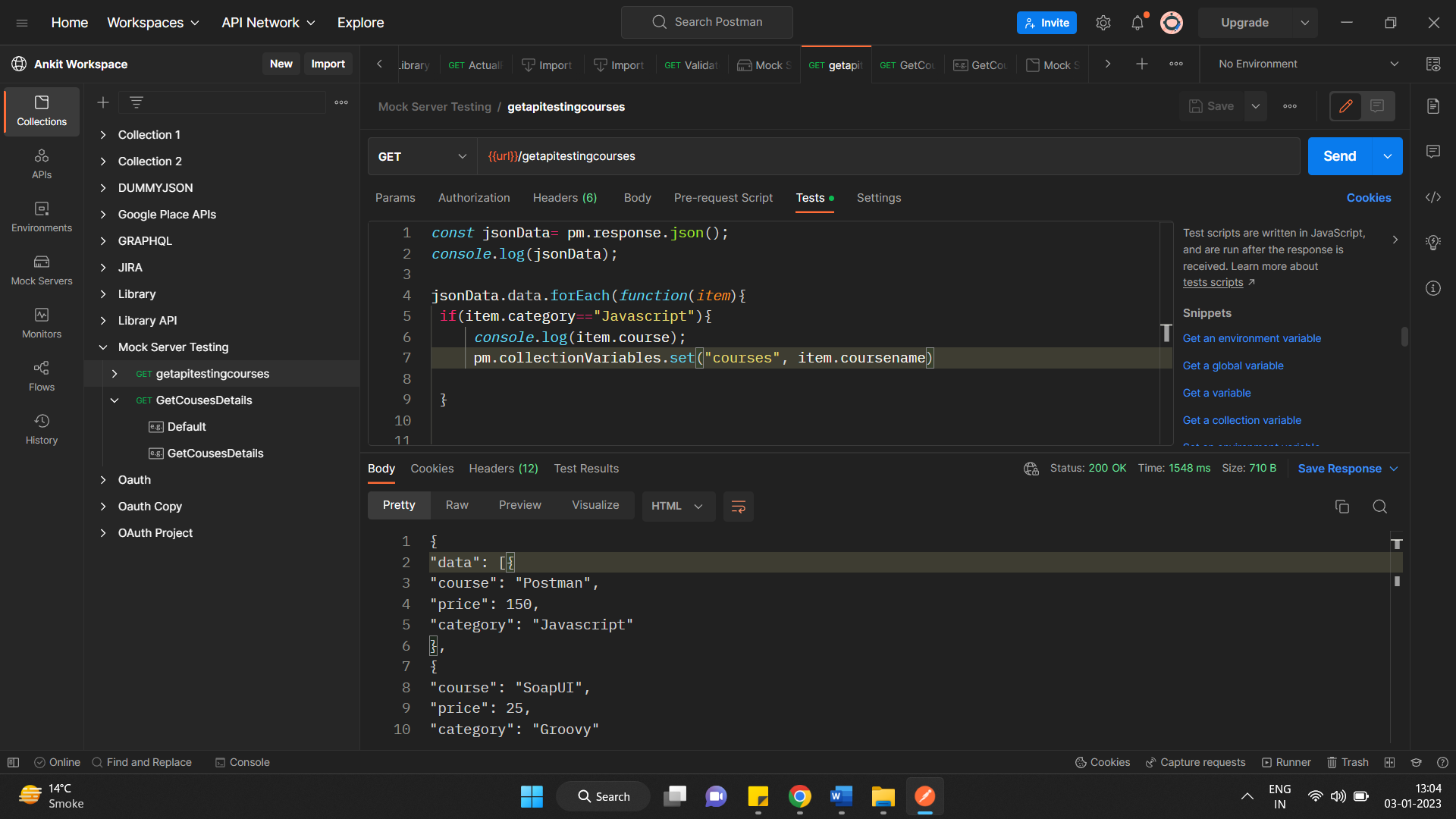


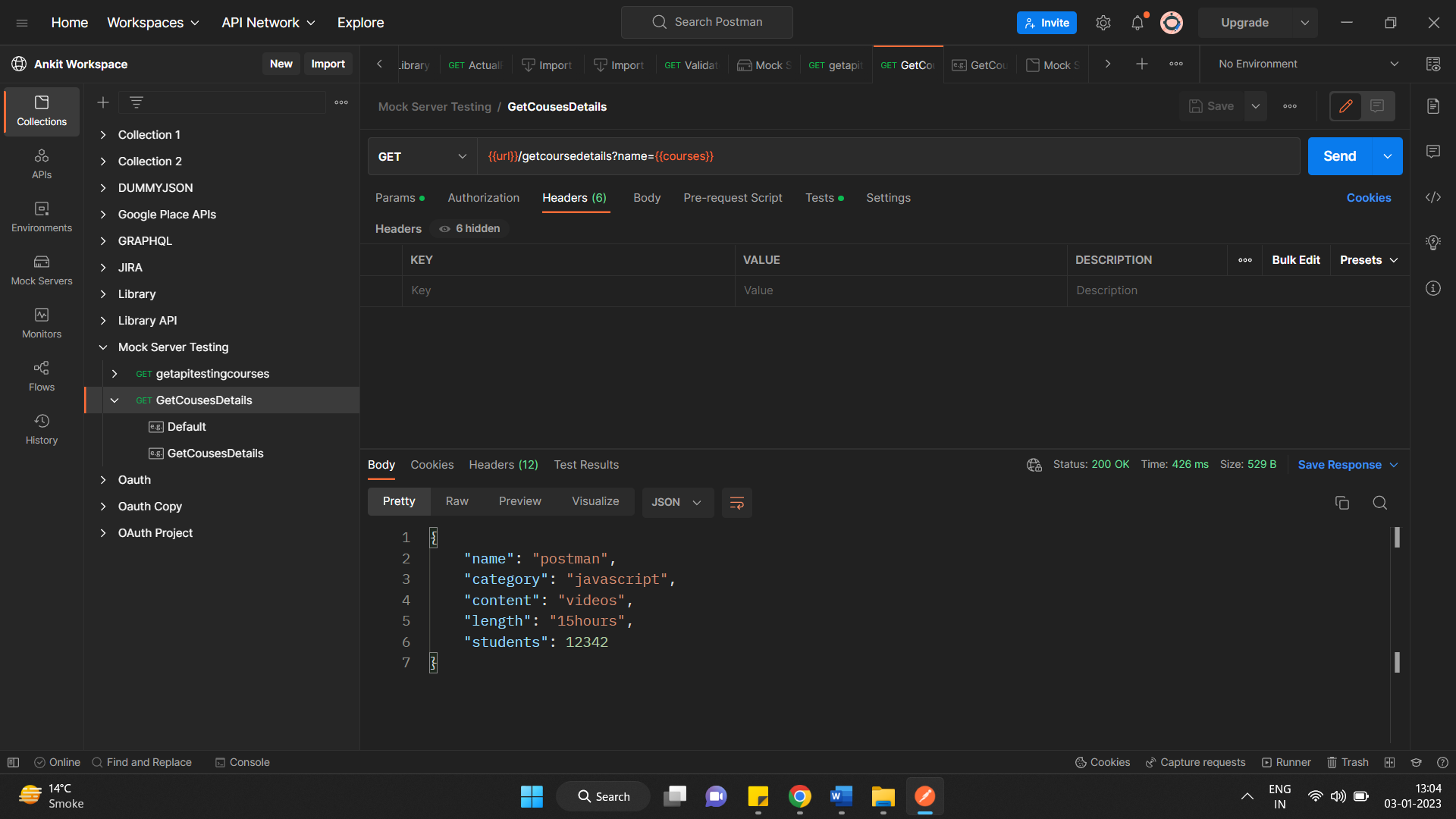




**MOCK SERVERS and how to use in postman**

**Follow mock server doc**





**NEWMAN**

**Collection runner through command line or terminal**

**It allows us to do CI CD integration and reporting**

**Node.js needed a prerequisite to install newman**

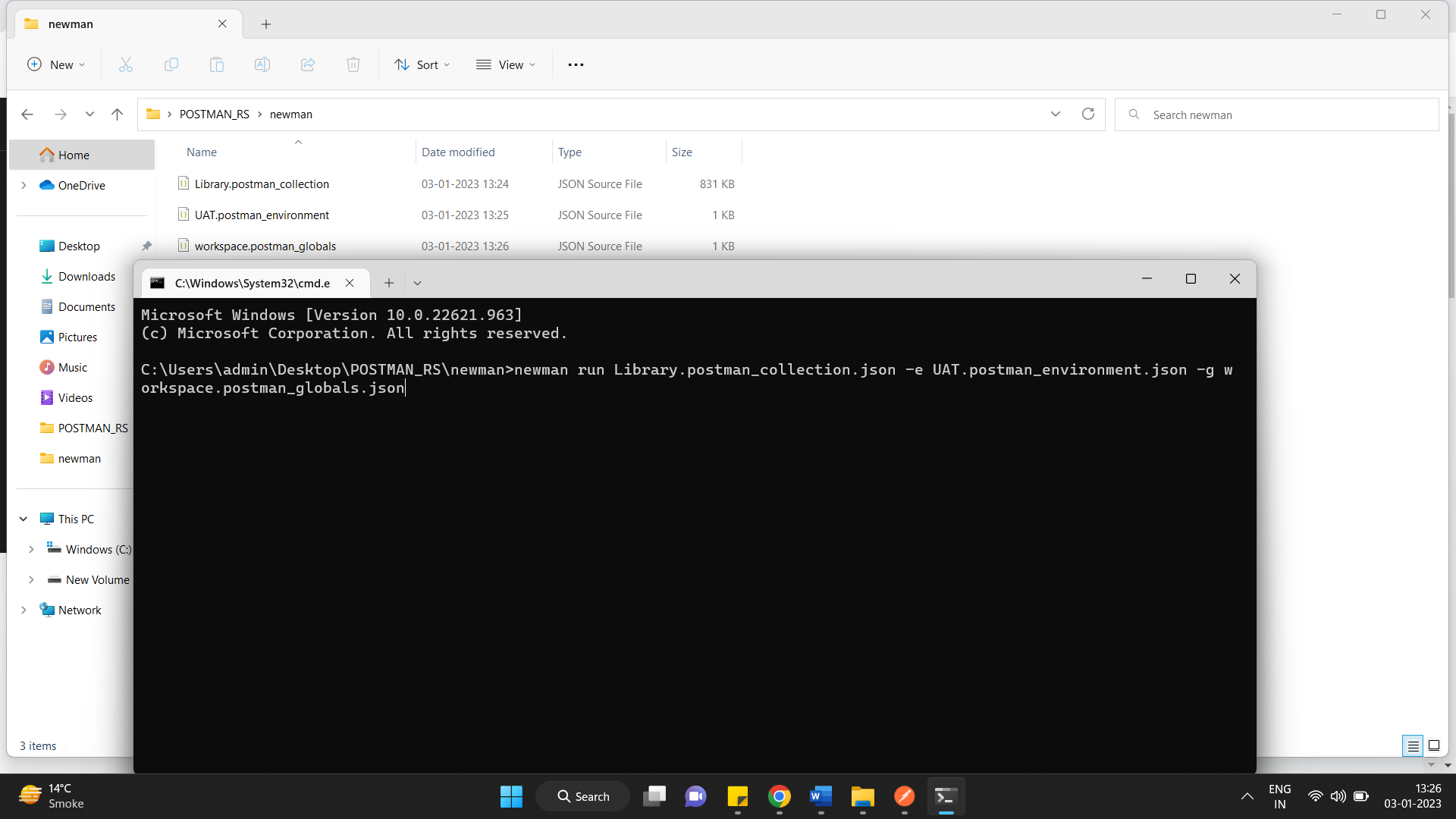
**npm install -h newman**

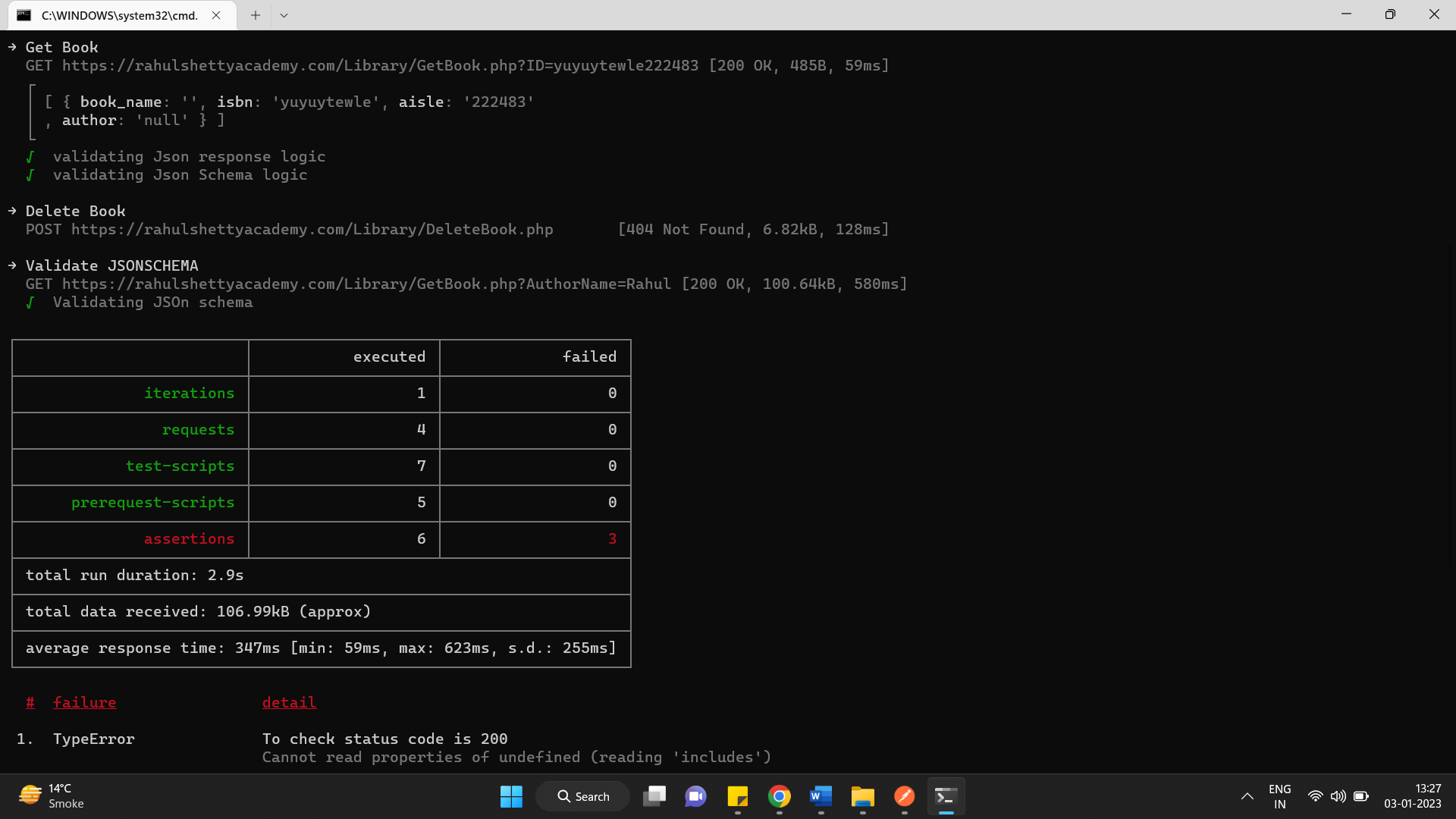
**How to run collection from new man**

**$ newman run <collectionfile>**

**newman run Library.postman\_collection.json -e UAT.postman\_environment.json -g workspace.postman\_globals.json**

**newman**



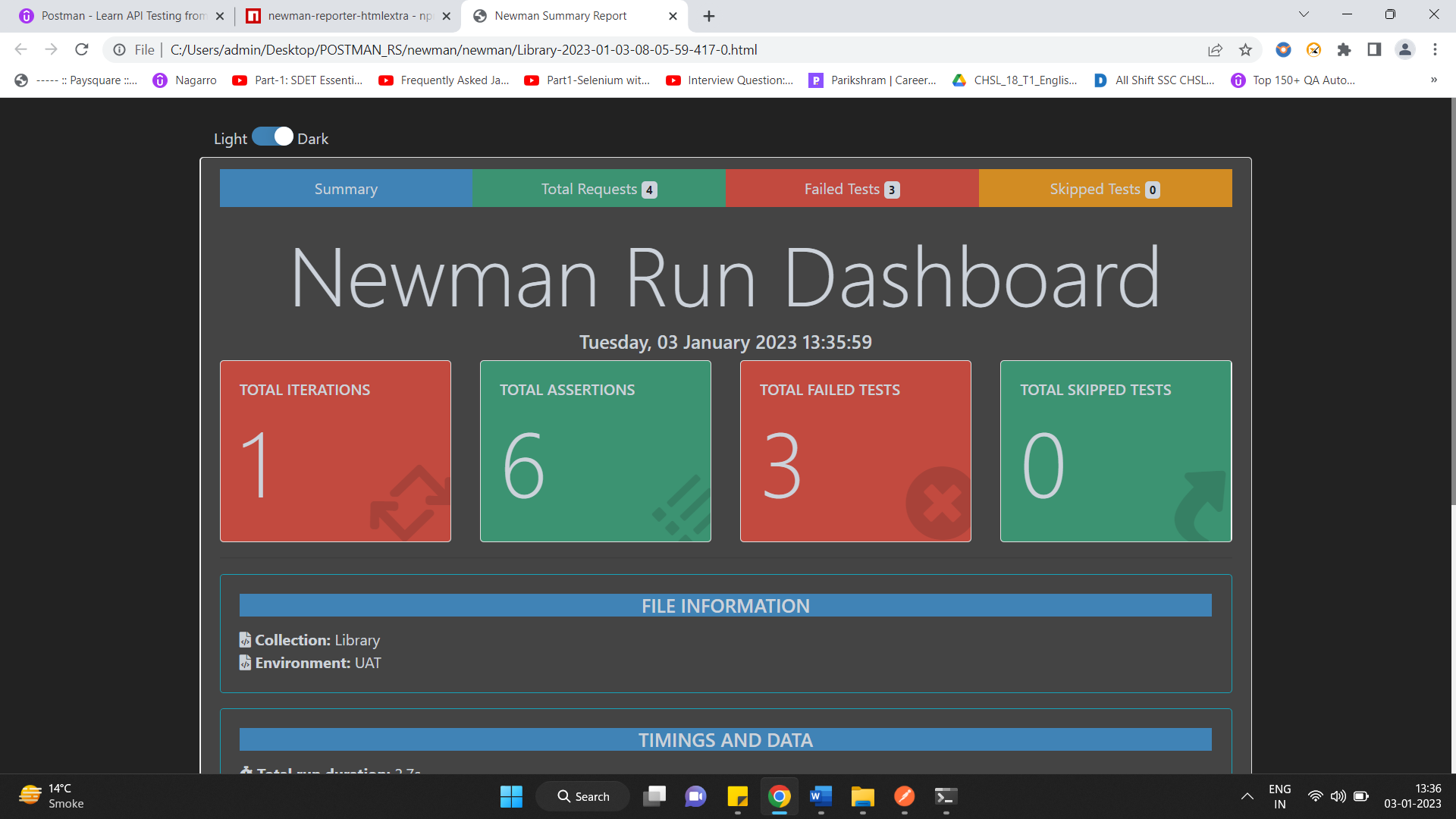


**Generate HTML reports by newman**

npm install -g newman-reporter-htmlextra

**newman run Library.postman\_collection.json -e UAT.postman\_environment.json -g workspace.postman\_globals.json**

**Newman -r hrmlextra**



**JENKINS INTEGRATION**

**Download Jenkins.war**

**Start Jenkins**

**Java -jar Jenkins.war -httpPort=9090**

**Create a new project in Jenkins and copy paste above newman command to BUILD section in job and click on build now**

**cd C:\Users\admin\Desktop\POSTMAN\_RS\newman newman run Library.postman\_collection.json -e UAT.postman\_environment.json -g workspace.postman\_globals.json**

**Newman -r hrmlextra**