

INTRODUCTION TO SOFTWARE TESTING QA / QE



Note:

This is compressed version of a 14+ hour course

Compressed to under 2 hours.

FREE Full course

<https://www.supersqa.com/learn-software-testing>

Welcome

- Outcome of the course
- Table of content
- Support information

Instructor

- Name = **Admas Kinfu**
- Creds
 - Software Development Engineer in Test (SDET) - **11+ years**
 - Teaching QA Automation/SDET - **9 years**
- Education
 - Masters & Bachelors in Mechanical Engineering - Grad 2012

Why QA

- Easiest way to start a tech career
- Great \$\$\$
- Doors to unlimited career path and growth
- Good work life balance

Outcome Of The course

- Understand what software testing is
- Understand the role of a tester
- Know the difference between “Manual” & “Automation” testing
- Get familiar with test cases and how to document them
- Understand some technical things we do like SQL and API Testing

Table Of Content (TOC)

- What Is Software Testing
- Manual Testing Vs. Automation
- What Is A Test Case
- Test Case Management Tools
- SDLC (Software Development Life Cycle)
- Intro To Database & SQL
- Intro to Backend/API Testing

Support

- Discord Private Community (& other Socials)
[SuperSQA.com/social](https://supersqa.com/social)
- Email:
 - admas@supersqa.com

Resources

- YouTube - Lots of tutorials educational material

[YouTube.com/supersqa](https://www.youtube.com/supersqa)

- Community Discussion:
 - Discord (**[SuperSQA.com/social](https://supersqa.com/social)**)





WHAT IS SOFTWARE TESTING

What is Software Testing?

Software Testing is the **process** of evaluating a software application to ensure that it meets its requirements, behaves as expected, and is free of defects.

What is Software Testing?

- Detect **bugs** and **defects** before software goes live
- Validate **requirements**
- Improve **quality**

What is Software Testing?

- At high level testers job is “**writing tests**”
- But this course is **much more** than that
- Must know how things **work** to test them effectively
- **Start** with simple examples, but we will get back to it

What is Software Testing?

* We will use different apps as examples *

* Action Is Better Than Words *

* Not A Fan Of Slides *

examples





MANUAL VS. AUTOMATION

Manual Vs Automation

Manual Testing

A process where **human testers** execute test cases **manually** without the help of scripts or automation tools.

Automation Testing

The **use of specialized software tools** or scripts to execute test cases and compare actual outcomes with expected results **automatically**.

Demo

Manual Vs Automation



supersqa.com

Manual Testing - Pros & Cons

Pros

- **Intuitive:** Human eye catches UI/UX issues, aesthetics, or anomalies that tools might miss
- **Flexible:** Quick to set up, no special framework or tooling needed
- **Exploratory:** Great for testing new or undefined features

Cons

- **Time-Consuming:** Repeated tests become tedious
- **Prone to Human Error:** Consistency can vary
- **Not Scalable** for large or complex test suites

Automation Testing - Pros & Cons

Pros

- **Speed & Efficiency:** Fast execution for multiple test iterations
- **Repeatability:** Minimizes human error; consistent results
- **Scalability:** Supports large test suites and complex scenarios
- **Integration:** Fits well with CI/CD pipelines for continuous testing

Cons

- **High Upfront Effort:** Requires scripting, setup, and tool investment
- **Maintenance Overhead:** Test scripts must be updated as the product changes
- **Limited Exploratory Capabilities:** Automated scripts follow predefined paths, missing the nuances of human exploration

When to use which?

Manual Testing

- **Exploratory Testing:** Ideal when requirements are not fully clear or for early feedback
- **Usability & UX:** Subjective areas that require human perception
- **Ad Hoc / One-off Testing:** Quick checks where automation overhead isn't justified

Automation Testing

- **Regression Testing:** Frequently run tests on mature, stable features
- **Performance/Load Testing:** Large-scale scenarios that would be impractical to do manually
- **High Repetition:** Tests that need repeated execution across multiple environments or builds





WHAT IS SDLC

SOFTWARE DEVELOPMENT LIFE CYCLE

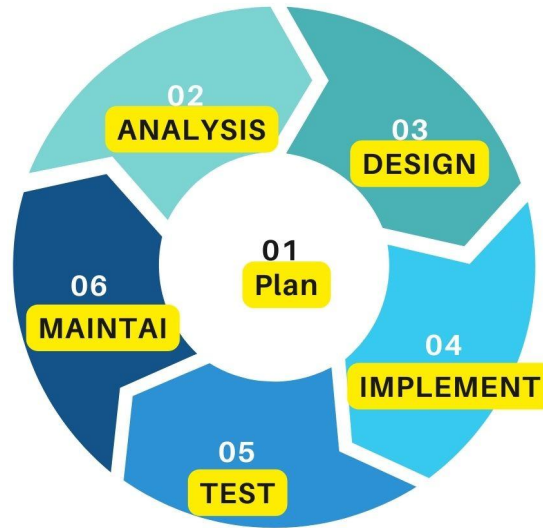
SDLC

Software Development Life Cycle

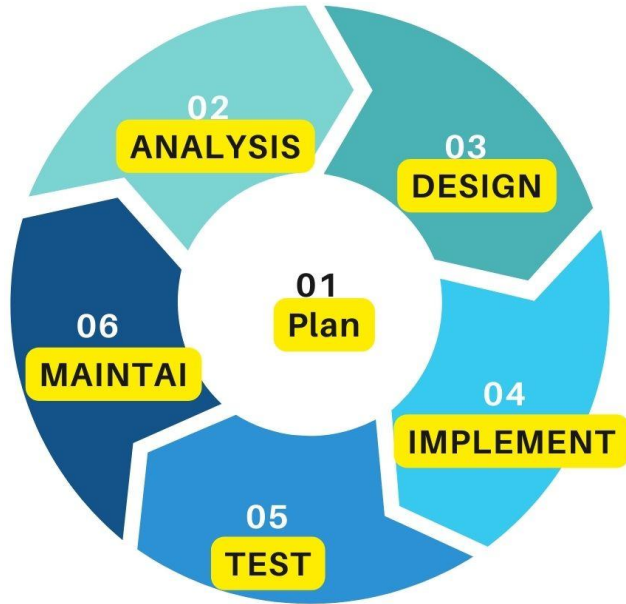
- A **structured** approach to software development.
- It defines **stages** and processes to ensure **high-quality** software is delivered efficiently.

SDLC

Software Development Life Cycle



SDLC



Phases Of SDLC

1. Planning
2. Requirements Analysis
3. System Design
4. Implementation (Coding)
5. Testing
6. Deployment
7. Maintenance

In Modern Companies QA Is Part Of Every Phase

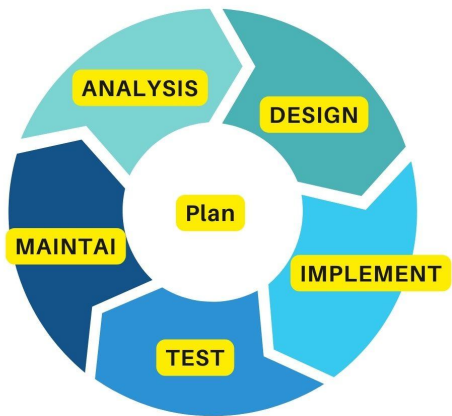
Agile Vs. Waterfall

“Software development methodologies”

Agile Vs. Waterfall

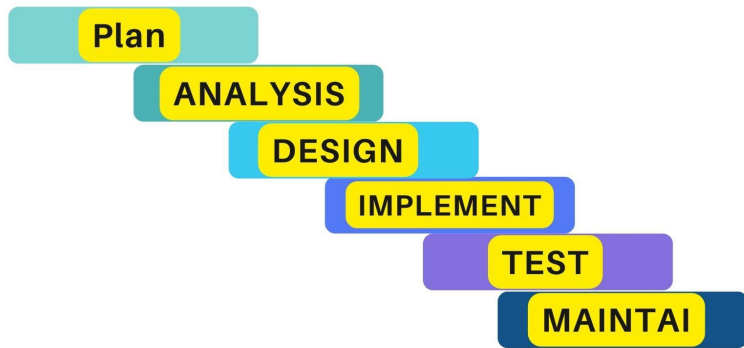
Agile:

Agile is an iterative approach to software development where requirements and solutions evolve through collaboration.



Waterfall:

A linear, sequential approach where each phase of development must be completed before moving to the next.



Agile

Characteristics:

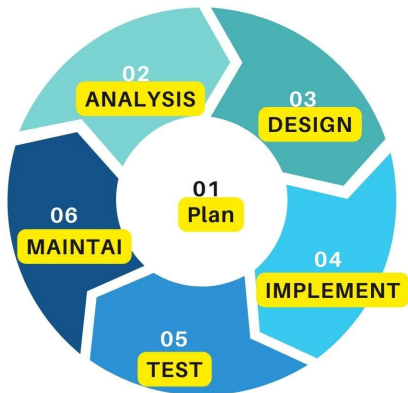
- Iterative and incremental.
- Encourages adaptability and quick responses to changes.
- Collaboration between teams and stakeholders.
- Frequent deliveries of working software.

Strengths:

- Great for projects with evolving requirements.
- Faster time-to-market with smaller, functional deliverables.
- Strong customer involvement.

Weaknesses:

- Requires high-level collaboration and communication.
- Difficult to implement without a mature team.



Waterfall

Characteristics:

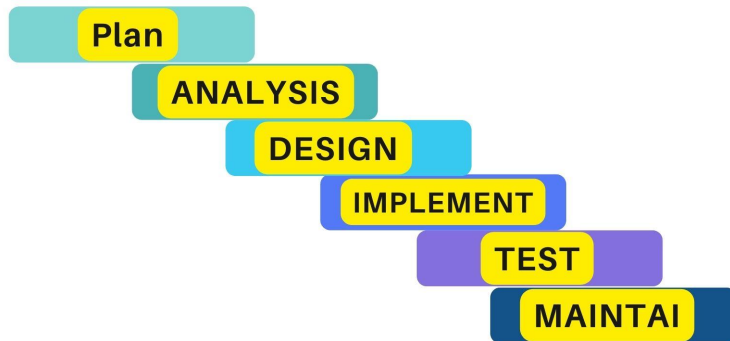
- Phases: Planning → Requirements → Design → Implementation → Testing → Deployment → Maintenance.
- Fixed requirements from the start.

Strengths:

- Simple and easy to manage for small, well-defined projects.
- Clear milestones and deliverables.
- Easier to measure progress.

Weaknesses:

- Not flexible; poor fit for projects with changing requirements. g happens late, increasing the cost of fixing issues.



Scrum! What is Scrum!

- Scrum is a **framework** for implementing Agile practices
- A **structured** framework that provides a **process** to manage Agile projects
- It **defines** roles, events, and artifacts to help teams deliver iterative work efficiently

Scrum!

Roles:

- **Product Owner:** Defines and prioritizes the work.
- **Scrum Master:** Ensures the team follows Scrum practices.
- **Development Team:** Builds and delivers the product.

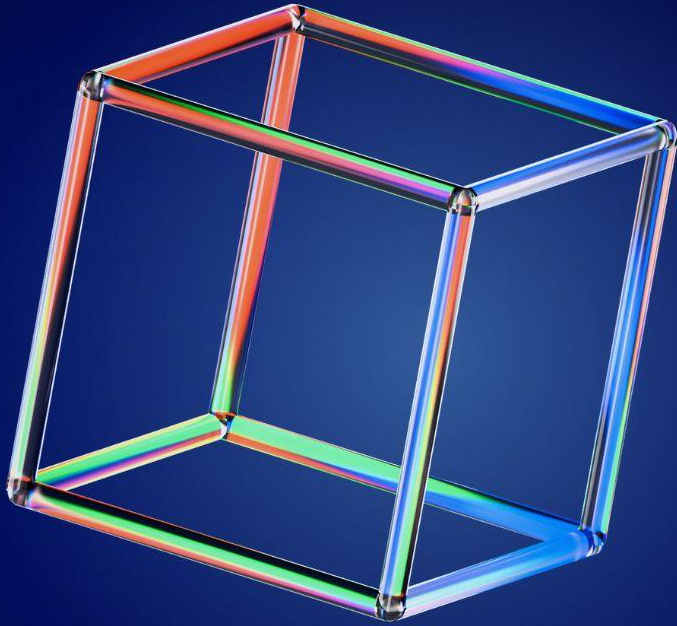
Events (Ceremonies):

- **Sprint Planning:** Plan the work for the Sprint.
- **Daily Scrum (Standup):** Brief daily meetings for progress updates.
- **Sprint Review:** Demonstrate the work completed.
- **Sprint Retrospective:** Reflect and improve the process.

Artifacts:

- **Product Backlog:** List of prioritized tasks.
- **Sprint Backlog:** Tasks selected for the Sprint.
- **Increment:** The working product delivered at the end of the Sprint.





WHAT IS A TEST CASE

What is a “Test Case”

A **test case** is a **documented set of actions, inputs, and expected results** used to verify a specific aspect of software functionality.

- Guide **systematic testing** by outlining precise steps
- **Validate requirements** and expected behavior
- Ensure **repeatability** and consistency in testing

What is a “Test Case”

Why Test Cases Matter

- **Clarity & Organization:** Captures exactly what and how to test
- **Risk Mitigation:** Identifies defects early by methodically testing features
- **Traceability:** Links directly to requirements or user stories
- **Reuse:** Facilitates regression testing on future software versions

Anatomy of a Test Case

1. Test Case ID

Unique identifier for reference and tracking

2. Title/Name

Brief description of what the test covers

3. Objective / Purpose

Why this test is needed and what requirement it fulfills

4. Preconditions

Any setup, configurations, or user states required before testing

5. Steps / Actions

Sequential instructions on how to execute the test

6. Test Data

Specific input values, files, or scenarios needed

7. Expected Result

Anticipated outcome if the software behaves correctly

8. Actual Result

The outcome observed when the test is executed

9. Pass/Fail Criteria

How to decide if the test is successful or not

Writing Effective Test Cases

- **Be Clear & Concise:** Keep steps easy to follow; avoid ambiguity.
- **Single Purpose:** Ensure each test case covers one feature or scenario.
- **Use Realistic Data:** Provide inputs that reflect real-world usage.
- **Be Independent:** Tests shouldn't rely heavily on each other's outcomes.
- **Cover Edge Cases:** Include boundary values, negative scenarios, and error conditions.

Example Test Case

Test Case ID: TC-001

Title: Valid User Login

Preconditions:

- User has a valid account
- Browser is open and website is accessible

Steps

1. Navigate to the login page
2. Enter a valid username and password
3. Click on "Login"

Expected Result:

- User is successfully logged in and taken to the dashboard

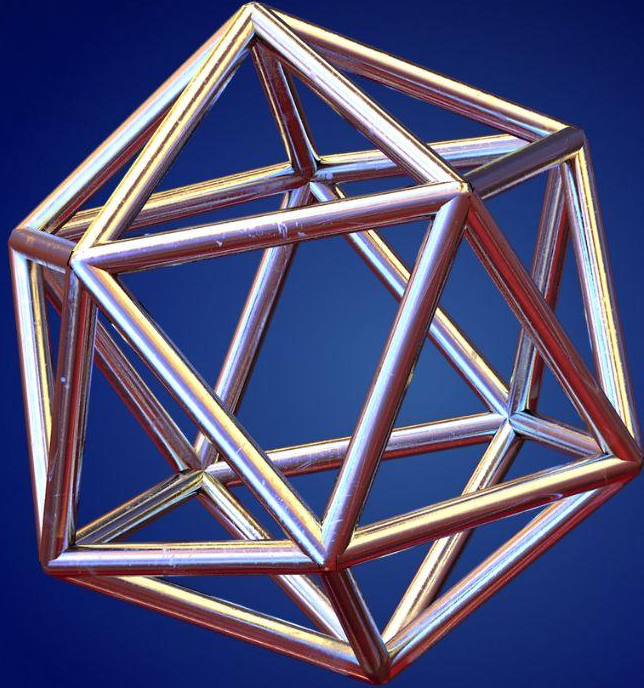
Actual Result:

- [To be filled in upon execution]

Pass/Fail Criteria:

- Pass if user lands on the dashboard with no errors





TEST CASE MANAGEMENT

Test Case Management

Test Case Management refers to the process of documenting, organizing, tracking, and maintaining test cases to ensure efficient testing.

It helps teams **track** test progress, identify issues, and ensure thorough testing of all functionalities.

Test Case Documented Example

Test Case ID	Test Case Name	Preconditions	Test Steps	Expected Result
TC001	Valid Login Credentials	User has a registered account.	1. Open the login page. 2. Enter a valid username and password. 3. Click the "Login" button.	User is successfully logged in and redirected to the homepage or dashboard.
TC002	Invalid Password	User has a registered account with a known password.	1. Open the login page. 2. Enter a valid username but an incorrect password. 3. Click the "Login" button.	Error message is displayed: "Invalid username or password." Login should fail, and the user remains on the login page.

Why Do We Need It?

- **Avoids duplication of effort by tracking existing test cases**
- **Ensures test coverage for all requirements and functionalities**
- **Helps in identifying which tests pass, fail, or need re-testing**
- **Provides a centralized repository for all test-related information**
- **Improves collaboration within the team**

Key Features of a Test Case Management System

- **Organized Storage:** Centralized repository for all test cases.
- **Tracking Progress / Test Runs:** Monitor test case status (e.g., passed, failed, blocked).
- **Team Collaboration:** Assign and share test cases with team members.
- **Integration:** Often integrates with tools like Jira or CI/CD pipelines.
- **Reporting:** Detailed insights on test execution and results.

Common Test Case Management Tools

- **qase.io**: Simple, modern tool for managing test cases and test plans.
- **TestLink**: Open-source tool for managing and tracking test cases.
- **TestRail**: Comprehensive tool for large teams.
- **Zephyr**: A popular plugin for Jira.
- **Spreadsheet**: good old Excel/Google Sheets

Qase.io

Create A Free Account



SQL & DATABASE INTRO





In Database/SQL Section

- ✓ What is Database
- ✓ What Is SQL
- ✓ Real Example Of SQL In Testing



What is database?

- | A database is an organized collection of data that is stored and managed by a computer system.

- | Databases are used to store and retrieve data in a structured and efficient manner.



What is database?

- | There are many different types of databases, including relational databases, NoSQL databases, and object-oriented databases.
- | Relational databases are the most common type of database and use tables to organize and store data.
- | NoSQL databases, are non-relational and are designed to handle large amounts of unstructured or semi-structured data.



Relational Databases

MySQL

- Most popular
- Commonly used in web applications like WordPress

PostgreSQL

- Complex data scenarios
- Known for its advanced features and complex data environments

SQL Server

- Microsoft
- Enterprise level & Integrates well with Microsoft technologies



NoSQL Databases

Unstructured databases

MongoDB

- Used in applications with dynamic and evolving schemas
- Known for its advanced features and complex data environments

Casandaa

- Distributed Database
- High scalability in large scale application

In This Course

we use: MySQL

- Most popular
- Free and open-source
- Easy
- Cross-platform (Windows/Mac/Linux)
- **Relational** database



SQL =

Structured Query Language

Relational Database

Data is stored as tables

Result Grid							
Filter Rows: <input type="text" value="Search"/>							
Edit:							
Export/Import:							
Fetch rows:							
emp_no	birth_date	first_name	last_name	gender	hire_date		
1222	1952-12-24	Lillian	FOO	M	1993-08-03		
10011	1953-11-07	Mary	Sluis	F	1990-01-22		
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18		
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20		
10014	1956-02-12	Berni	Genin	M	1987-03-11		
10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02		
10016	1961-05-02	Kazuhito	Cappelletti	M	1995-01-27		
10017	1958-07-08	Cristinel	Bouloucos	F	1993-08-03		
			Peha	F	1987-04-03		
			Haddadi	M	1999-04-30		

Result Grid							
Filter Rows: <input type="text" value="Search"/>							
Edit:							
Export/Import:							
supplier_ids	id	product_code	product_name	description	standard_cost	list_price	
4	1	NWTB-1	Northwind Traders Chai	NULL	13.5000	18.0000	
10	3	NWTCO-3	Northwind Traders Syrup	NULL	7.5000	10.0000	
10	4	NWTCO-4	Northwind Traders Cajun Seasoning	NULL	16.5000	22.0000	
10	5	NWTO-5	Northwind Traders Olive Oil	NULL	16.0125	21.3500	
2;6	6	NWTJP-6	Northwind Traders Boysenberry Spread	NULL	18.7500	25.0000	
2	7	NWTDFN-7	Northwind Traders Dried Pears	NULL	22.5000	30.0000	
8	8	NWTS-8	Northwind Traders Curry Sauce	NULL	30.0000	40.0000	
2;6	14	NWTDFN-14	Northwind Traders Walnuts	NULL	17.4375	23.2500	
6	17	NWTCFV-17	Northwind Traders Fruit Cocktail	NULL	29.2500	39.0000	
1	19	NWTBGM-19	Northwind Traders Chocolate Biscuits Mix	NULL	6.9000	9.2000	

products 1

SQL

SELECT * FROM <table name>;

SELECT * FROM <database name>.<table name>;

Example:

SELECT * FROM employees;

SELECT * FROM my_store.product_prices;

INSERT INTO city (Name, CountryCode, District, Population)
VALUES ('SampleCity', 'USA', 'SampleDistrict', 123456);

DELETE FROM city **WHERE** CountryCode = 'USA' AND Population < 10000;

DEMO



SQL IN TESTING

Why SQL In Testing?

- Most application rely on databases
- Testers often need to verify data integrity, correctness, and consistency

Why SQL In Testing?

- Frontend & Backend Testing
- The frontend says something happened correctly, **but did it?**

We should verify in DB

Example SQL In Testing

- Verify user creation
 - Create the user in the UI (the browser)
 - The UI says success user is created
 - Verify the database captured everything



BACKEND / API TESTING



Introduction To BE Testing!

- Backend is the stuff we do not see
- “Behind-the-scenes”
- Logic, servers, databases,....
- Every* Frontend has a backend
- More jobs for backend testing (my opinion)

* “Every” is a strong word

Why BE Testing Is Important

- **Reliability & Stability - Core logic of the application**
- **Data Integrity**
- **Security**
- **Performance**
- **Cost**

Common Types Of BE Testing

- **API Testing**

- Verifying endpoints, request/response formats, and status codes
- Checking data validation and error handling

- **Integration Testing**

- Testing how different modules and services communicate
- Ensuring the entire system works cohesively

- **Database Testing**

- Validating data integrity and consistency
- Checking performance under load or large datasets

- **Performance & Load Testing**

- Assessing server responsiveness under various traffic levels
- Ensuring stability and scalability

- **Security Testing**

Overview Of APIs

- APIs are used for one application to get data from another application
- An app can use functionalities of another application without implementing its own logic
- An app can use data from another app
- Someone creates functionality and others can use that functionality
- Very easy to implement existing tech into new tech

Examples

- **Google Map API**

- Not everyone will build tech to create map
- But they embed map functionality to their app using “**Google Map API**”
- For example:
 - Zillow (Real estate companies) - use map in their app but didn't invent their own mapping.
 - Uber - map is their bread & butter but they just use Google Map API
 - Airbnb -

Examples

- **Stripe API (Payment API)**
 - Not everyone will build tech to handle payments
 - But they can accept payments on their app using “**Stripe API**”, “**PayPal API**” or others ...
 - Any app that accepts payments can simply use API like this

Examples

- **Qase.io API**
 - Create test cases using API
 - Get list of test cases
 - Get reports
 - Update test runs from automation (CI/CD) tools

Example API Call

```
curl --location 'http://127.0.0.1:8888/localdemostore/wp-json/wc/v3/products' \
--header 'Content-Type: application/json' \
--data '{
    "name": "Admas hat",
    "type": "simple",
    "regular_price": "21.99",
    "description": "adafdkfjl;adkf a;dlskfjasdf eqwiodfiohadf adfadfadf",
    "short_description": "adfadfasdf"
}'
```



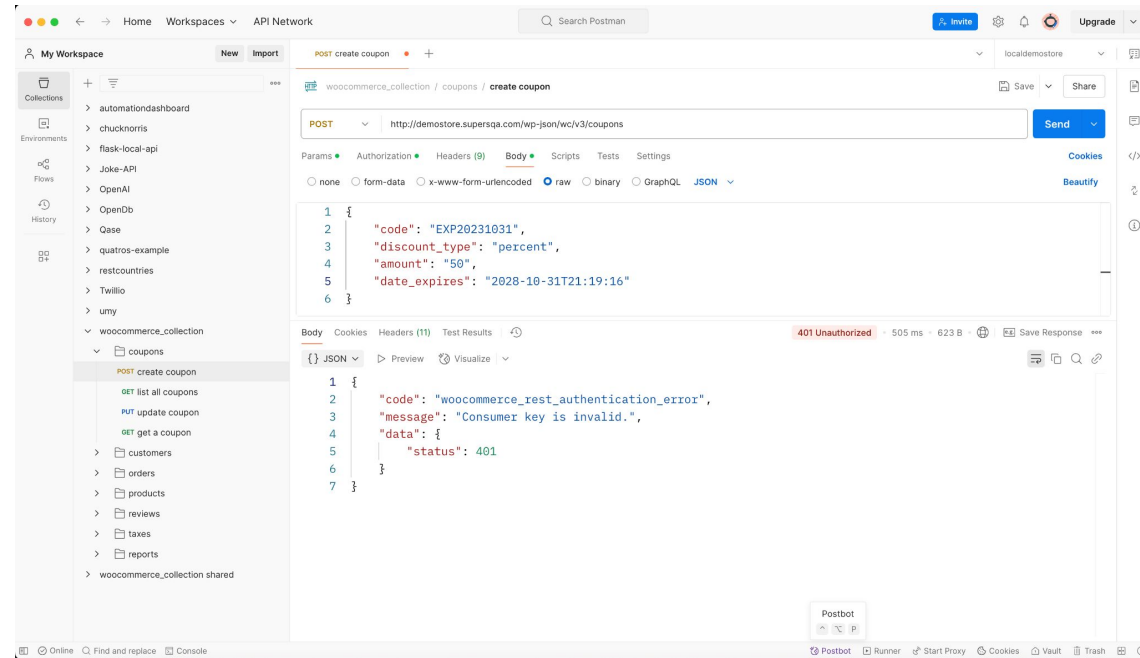
REST API

- **REST - Representational State Transfer**
- **HTTP Protocol**
- **HTTP Methods:**
 - **GET**
 - **POST**
 - **PUT**
 - **DELETE**

REST API - Key Components

- Method (GET, POST, PUT, PATCH, DELETE, ...)
- Base URL (<https://api.example.com>)
- Endpoint (/users)
- URL (<https://api.example.com/users>)
- Request
 - Headers ({“Content-Type”: “application/json”})
 - Body ({“email”: “foo@supersqa.com”})
 - Query Parameters
(<https://api.example.com/users?status=active&limit=10>)
- Response
 - Status Code (2XX, 3XX, 4XX, 5XX)
 - Body ({“email”: “foo@supersqa.com”})
 - Headers

TOOLS TO WORK WITH REST API





CONCLUSION



CONCLUSION

- Hoping you know more now than you did
- Hoping you are excited about the journey
- Ask questions
- Do research
- Really consider “Automation”

CONCLUSION

- Join the community:

SupersQA.com/social

Keep Learning and Upskilling