# Load Automation Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
|  | Prepared by | Reviewed By | Approved By |
| Name | Madhuri Nannapuraju |  |  |
| Role | QA Automation Lead |  |  |
| Date | 09/10/2024 |  |  |

# Objective:

The primary objective of this Automation Test Plan is to portray the step-by-step process that will enhance the efficiency, reliability and scalability of the testing process. This automation of repetitive, time-consuming and critical test scenarios will improve the product quality and will speed up the delivery cycle.

# Assumptions:

This automation plan can be considered as the initial plan to automate Load test on API Endpoints. Given the allocated resource will completely concentrate on the automation process & the implementation itself.

# Project Overview:

Live interactive streaming app, which is used across all different platforms to upload, watch, comment & like on the live content. This Application has multiple API’s that are used to make the application work. Using the Load Automation process we can verify the performance & scalability of these API’s.

# API Test Life Cycle:

A diagram of a light bulb

Description automatically generated

# Load/Performance Test Life Cycle:

With the increasing interactivity of applications and requests to them from users, the load on the server is growing and requires a high level of service. To be sure of the correct operation of the app in real conditions, software performance testing is imperative. With performance testing, we detect:

* slow loading time
* long response time
* poor scalability issues
* performance bottlenecks
* system reliability issues

As a result, we assess the efficiency and operational [capability of the application](https://qawerk.com/services/mobile-application-testing/) during the design and deployment phases. During the testing process, we determine the number of users simultaneously working with the application and the limits of acceptable performance with increasing load, also examine productivity at high, extreme, and stressful burdens.

A diagram of a speedometer

Description automatically generated

# Test Coverage:

Test Methodolgies that can be covered during this Load Automation test process:

* Sanity/Smoke testing
* Functional testing
* Performance/Load testing
* Stress testing
* Spike testing
* Scalability testing

# API’s In-scope:

**In- Scope:**

* <https://jsonplaceholder.typicode.com/posts>
* <https://jsonplaceholder.typicode.com/comments>
* <https://jsonplaceholder.typicode.com/albums>
* <https://jsonplaceholder.typicode.com/photos>
* <https://jsonplaceholder.typicode.com/todos>
* <https://jsonplaceholder.typicode.com/users>

**Out-Of Scope:**

Any other API’s

# High level API Load Test Plan & Time Estimation:

* API’s accessibility
* API’s Response Validation
* API’s Load testing
* API’s Stress testing
* API’s Spike testing
* API’s Scalability testing

A screenshot of a document

Description automatically generated

PS: Above test automation plan is a prediction, which can vary based on the change of scope, resource availability, Addition of test scenarios & Unresolved/Blocked issues.

# Defect Management:

All the issues identified during the Automation process with respect to the project will be raised manually using the JIRA system. As mentioned above in the Test estimates after the Automation of Status reporting is in place- Status of the execution will be notified through Email, but the Issues/bugs identified during or after the process will be raised manually in JIRA.

# Recommended Tools:

There are few good Load/performance testing tools, out of which JMeter & POSTMAN are one of the best for Performance testing & API testing tools

**JMeter**

[JMeter](https://jmeter.apache.org/) is one of the most popular and free open source load testing tools, which offers several features in comparison to commercial tools. You will be able to record the test scripts and analyze the reports feasibly. The developers look after updating and maintaining the open source tools frequently to support new technologies. JMeter comes with the Swing graphical API, which allows it to run across all workstations or environments which accept JVM.

Key Features:

* Uses protocols such as SOAP, FTP, Java Objects, databases with JDBC, and HTTP/HTTPS.
* Embedded with a nice IDE for recording, debugging, and building performance tests.
* JMeter 3.1 comes with Groovy as the default coding language.
* Configured for testing the performance of mobile applications.

**POSTMAN** desktop app has multiple API testing features, few are stated below which will help us to reach & validate the Project scalability goals.

* Test the scalability of the API’s by increasing up to 300 Virtual users per min for a certain duration
* Test the performance by Ramp up the VU