**Industrial Internship Report on**

**”URL Shortener”**

**Prepared by**

**[Dheeraj Saini]**

|  |
| --- |
| *Executive Summary* |
| This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).  This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time.  My project was (URL Shortener)  This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship. |

**TABLE OF CONTENTS**

[1 Preface 3](#_Toc139702806)

[2 Introduction 4](#_Toc139702807)

[2.1 About UniConverge Technologies Pvt Ltd 4](#_Toc139702808)

[2.2 About upskill Campus 8](#_Toc139702809)

[2.3 Objective 9](#_Toc139702810)

[2.4 Reference 9](#_Toc139702811)

[2.5 Glossary 10](#_Toc139702812)

[3 Problem Statement 11](#_Toc139702813)

[4 Existing and Proposed solution 12](#_Toc139702814)

[5 Proposed Design/ Model 13](#_Toc139702815)

[5.1 High Level Diagram (if applicable) 13](#_Toc139702816)

[5.2 Low Level Diagram (if applicable) 13](#_Toc139702817)

[5.3 Interfaces (if applicable) 13](#_Toc139702818)

[6 Performance Test 14](#_Toc139702819)

[6.1 Test Plan/ Test Cases 14](#_Toc139702820)

[6.2 Test Procedure 14](#_Toc139702821)

[6.3 Performance Outcome 14](#_Toc139702822)

[7 My learnings 15](#_Toc139702823)

[8 Future work scope 16](#_Toc139702824)

# Preface

This whole 6 weeks of the internship have been very useful for me as I have learned a lot of new things during this period of time. During this 6 weeks of the internship I have learnt about the importance of Python programming , gain skills to achieve good heights in the Understanding and implementation of Python Programing , I have also solved the problems during the quiz test which has increased my confidence in the particular field.

Python is a widely used programming language applied in websites, software, data analysis and visualization. If we are looking to build a career in programming as a software developer, then a Python internship can be a valuable addition to our resume. An internship is an official program offered by the organizations to help train and provide work experience to students and recent graduates. The concept of working as an intern began a long time ago but has drastically evolved over the years. Internships first started as a labourer who would take on young individuals and teach them their art or trade. In exchange for being taught a skill, the trainee would agree to work for the labourer for a specific time. Even then, the purpose of an internship or rather an apprenticeship was to gain new skills to be able to obtain future work.

URL Shortener, as the name suggests, is a service to help to reduce the length of the URL so that it can be shared easily on platforms like Twitter, Linkdin etc., where number of characters is an issue. An URL shortener is a website that reduces the length of your URL (Uniform Resource Locator). The idea is to minimize the web page address into something that's easier to remember and track. There are many URL shorteners on the market today, including Bit.ly, Goog. le and Tinyurl.com.

This is a wonderful Opportunity given by USC/UCT for every learner. As python is powerful, easy to understand and fast growing language in many fields. So once again I would like to thanks USC/UCT for providing such a great internship in python along with quizzes.

How Program was planned



My overall experience is so good and unforgettable. I liked everything about this program , I have been consistent throughout on this tenure as an intern. The Skills gained in this phase will surely help in my upbringings and also for my future endeavors.

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the internship. I express my sincere gratitude to Mr. Apurv, Kaushlendr Singh Sisodia and Upskill Campus team for providing me support , cooperation during this internship.

I would like to suggest everyone for joining this organization as I have been a part of this organization and it was truly so fruitful to me and also it’s a great opportunity to every learner.

# Introduction

## About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various**Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end**etc.



1. UCT IoT Platform **(****)**

**UCT Insight** is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

* It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
* It supports both cloud and on-premises deployments.

It has features to  
• Build Your own dashboard  
• Analytics and Reporting  
• Alert and Notification  
• Integration with third party application (Power BI, SAP, ERP)  
• Rule Engine

1. **Smart Factory Platform (****)**

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

* with a scalable solution for their Production and asset monitoring
* OEE and predictive maintenance solution scaling up to digital twin for your assets.
* to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
* A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

1.  based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

1. Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



## About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

<https://www.upskillcampus.com/>

upSkill Campus aiming to upskill 1 million learners in next 5 year



## The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

## Objectives of this Internship program

The objective for this internship program was to

 ☛ get practical experience of working in the industry.

 ☛ to solve real world problems.

 ☛ to have improved job prospects.

 ☛ to have Improved understanding of our field and its applications.

 ☛ to have Personal growth like better communication and problem solving.

## Reference

[1] UniConverge Technologies Pvt. Ltd. from upskill internship portal, uniconvergetech.in

[2] IoT academy from theiotacademy.com

[3] upSkill Campus from learn.upskillcampus.com and their youtube channel.

[4] W3 School

[5] CodeWith Harry.

[6] Java Tutorial Point

[7] Geeks for Geeks

## Glossary

|  |  |
| --- | --- |
| Terms | Acronym |
| Predective | relating to or having the effect of predicting an event or result |
| Insight | Information Systems Integration using Global Hypermedia Technology |
| Hindrance | a person or thing that makes it difficult for you to do something, an obstacle or obstruction |
| Dynamic | active or forceful |
| Optimization | the action of making the best or most effective use of a situation or resource |

# Problem Statement

A URL shortener is a tool that takes a long URL (Uniform Resource Locator) and converts it into a shorter, more manageable URL that redirects to the original long URL. This is often used to share links on social media, in emails, or in other situations where long URLs are impractical.

our task is to create a URL shortener program using Python. The program should be able to take a long URL as input and provide a shortened URL as output. Additionally, the program should be able to handle requests for the shortened URL and redirect users to the original long URL.

**Example:**

User input: **https://www.example.com/very/long/url/to/a/page**

Shortened URL generated: **https://short.url/abc123**

When the shortened URL **https://short.url/abc123** is accessed, the program should redirect the user to **https://www.example.com/very/long/url/to/a/page**.

# Existing and Proposed solution

There are Many URL shortner present in the development world. Many links have been working to reduce the length of the working URL. Many developments are under process to increase the accuracy of the short URL and also to increase the processing time. Some of the Famous URL shortner used in the developing world are :-

* Bitly : Bitly is a widely recognized URL shortening service that offers link analytics, tracking, and customization options. It's commonly used for social media sharing and marketing campaigns.
* TinyURL : One of the earliest URL shorteners, TinyURL provides a simple interface to create short links without the need for an account. It's straightforward and easy to use.
* Ow.ly : Part of Hootsuite's suite of tools, Ow.ly offers link shortening and social media analytics, making it useful for social media managers and businesses.

Etc.

And there are many links which Provide service for the Shortning of the URL , But there are real time issues in the services provided by the shortners. Some of them lack in the accuracy while many of them takes a large ample amount of time to process the webpage through the shorter link , while some of the links will teleport the user to wrong web page through the shorter link.

We are planning to build a URL shortner which can overcome all the issues of the existing shortners. We are trying to achieve Maximum accuracy and loading speed to provide best web service to the users. We are looking forward to the data privacy of the servers and users also so that there is no hindrance of the data when the conversion of the length is taking place.

## Code submission (Github link) :

## https://github.com/Ds10011003/Upskill\_Campus\_pythonProjectReport.git

## Report submission (Github link) :

## https://github.com/Ds10011003/Upskill\_Campus\_pythonProjectReport.git

# Proposed Design/ Model

The model which we have proposed follows a Dynamic process of processing the tasks at various levels of its execution. We have made a model which allows the processing of every stage without the interruption of any outward hindrance. The flow of the control can be easily understood by the Level of diagrams we have created for the better understanding of our model.

## High Level Diagram

This is a High Level Entity – Relationship diagram which shows the Working of an URL shortner.

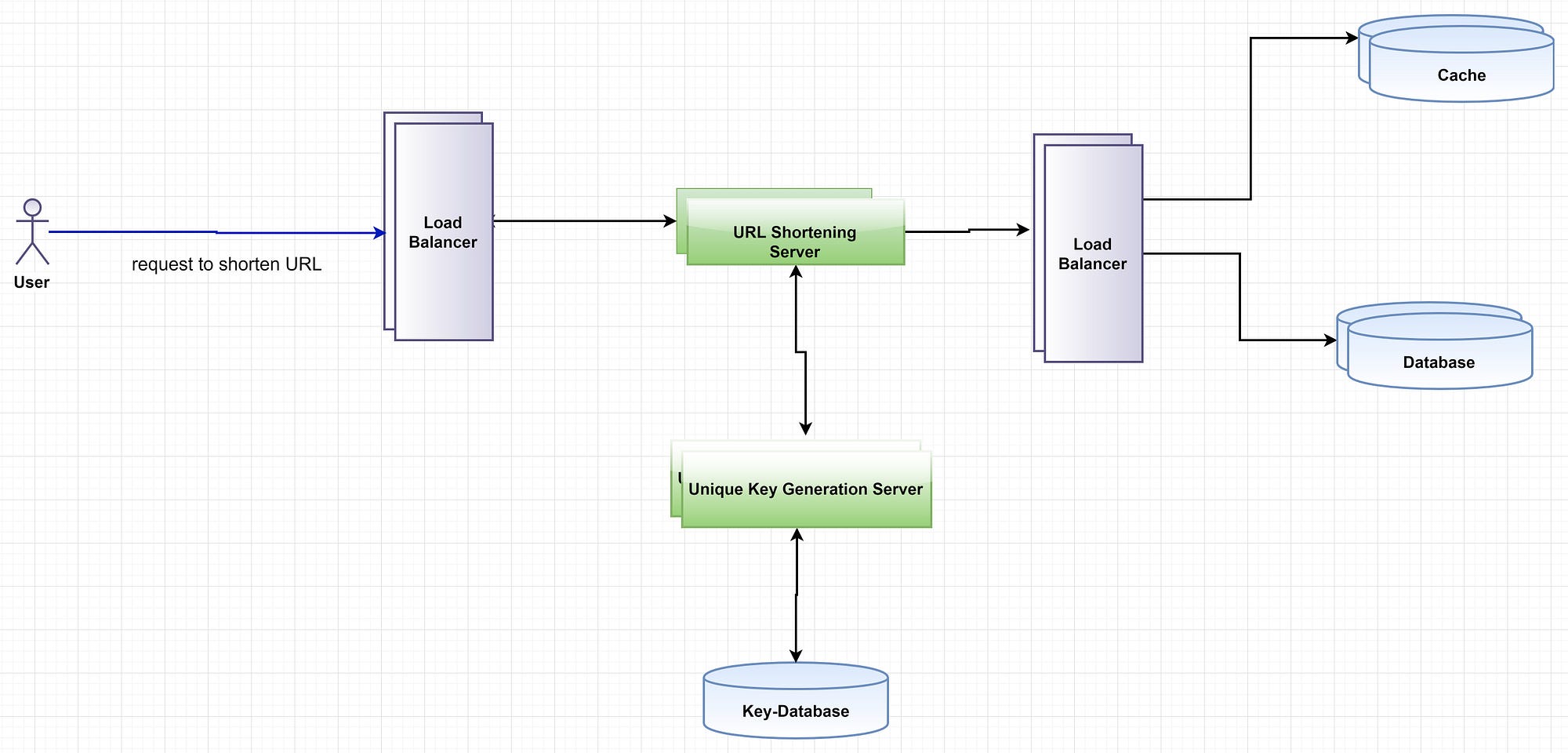
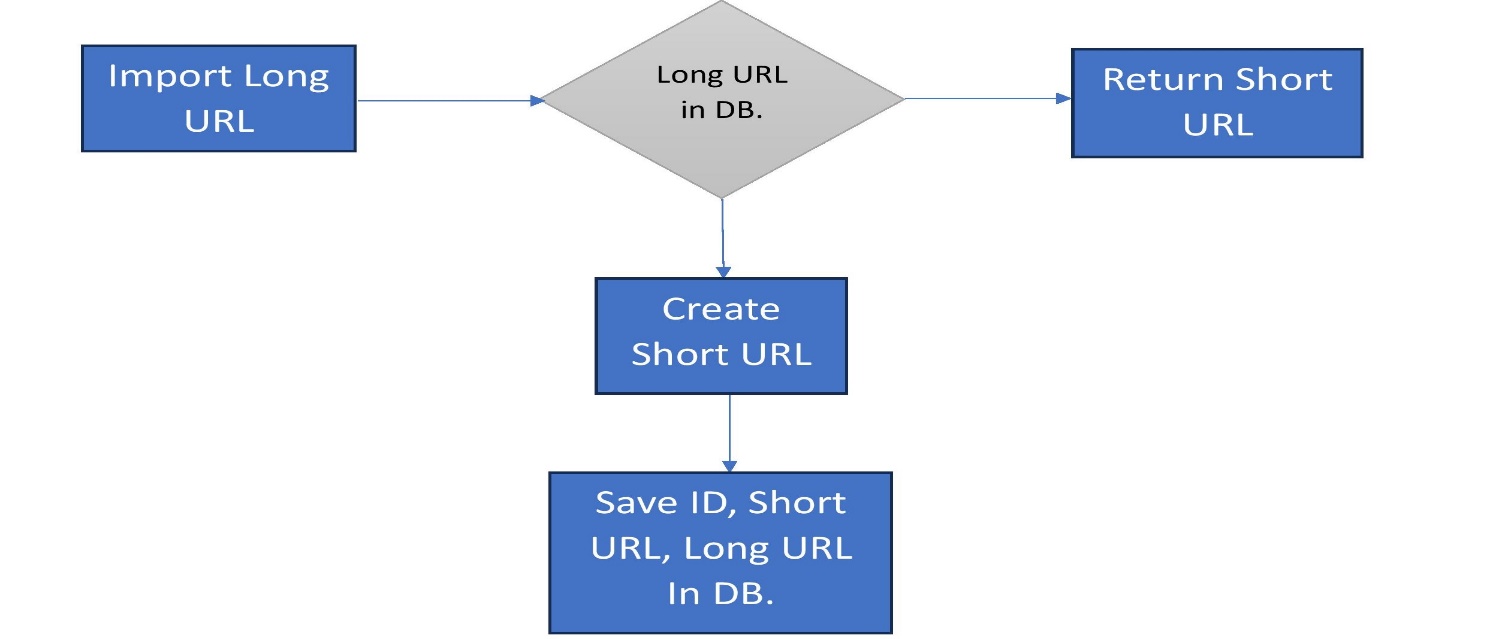


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

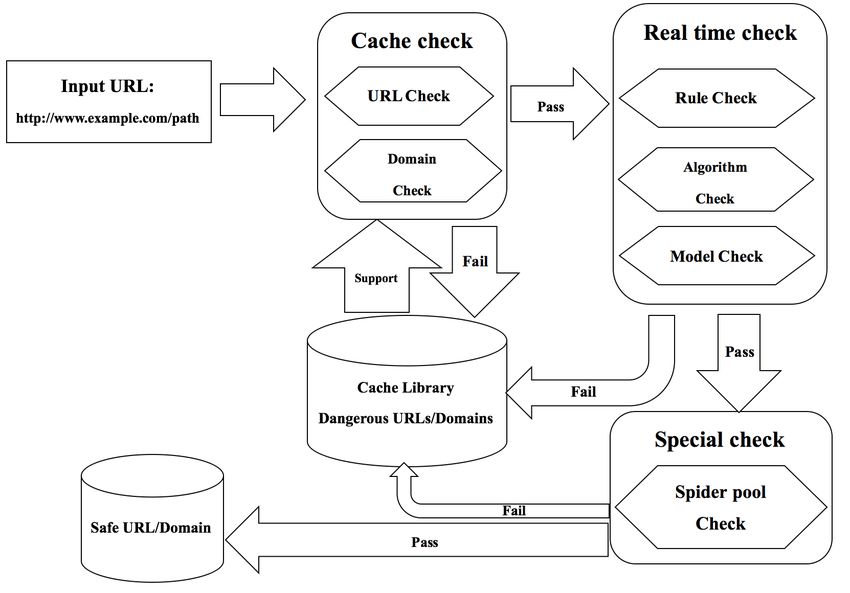
## Low Level Diagram :-

* 1. Low Level Diagram



## Figure 2 : Low Level Diagram of the System

## Interfaces :-



**Figure 3 : Data Flow Diagram**

# Performance Test

The significance of the performance testing and its impact on real industries for a Python-based URL shortener project is profound and goes beyond mere academic exercises. This critical aspect underscores the practical applicability and relevance of the project in real-world scenarios. Here's why this work is essential for real industries:

**User Experience Enhancement:** In today's fast-paced digital landscape, user experience is paramount. A robust and responsive URL shortener ensures that users can quickly and reliably access the original content, leading to enhanced user satisfaction. Industries heavily reliant on online interactions, such as e-commerce, digital marketing, and content distribution, can leverage this technology to optimize user journeys.

**Business Efficiency:** Time is money, and any delay in URL redirection can lead to lost opportunities or decreased productivity. Real industries, including e-commerce, advertising, and social media platforms, benefit from a URL shortener that consistently delivers quick response times. This can translate into increased efficiency in marketing campaigns, quicker access to product information, and streamlined user interactions.

**Reliability and Trust:** For industries where trust and reliability are pivotal, such as financial services and healthcare, an accurate and durable URL shortener is essential. Ensuring that links lead to the correct and intended content fosters trust among users and clients. By providing a dependable tool, the project directly contributes to maintaining the credibility of these industries.

**Scalability and Resilience:** Real industries experience varying levels of traffic, and a URL shortener capable of withstanding high loads and maintaining optimal performance is invaluable. Sectors like news media, event management, and online entertainment often witness sudden spikes in user activity. A performance-tested URL shortener ensures seamless scalability and prevents service disruptions during peak periods.

**Resource Optimization:** Efficient memory usage and power consumption are not only environmentally responsible but also have cost implications for industries. A URL shortener that consumes resources judiciously can reduce operational expenses for data centers, cloud services, and other infrastructure-dependent sectors.

**Competitive Edge:** Industries are in a constant race to provide the best user experience. A Python-based URL shortener with superior speed, accuracy, and durability gives companies a competitive edge by enabling them to offer faster and more reliable services to their customers.

**Real-World Challenges:** The complexities of real-world usage, such as diverse URL formats, high traffic volumes, and extended operational periods, introduce challenges that academic projects may not address. By tackling these challenges, the project demonstrates its readiness to meet industry-specific demands.

**Business Continuity:** For industries where downtime can lead to financial losses or reputational damage, a performance-tested URL shortener ensures business continuity. Sectors like online retail, travel, and booking services rely on uninterrupted access to content and services.

The project encompasses the creation of a URL shortener using Python, designed to provide users with an efficient tool for generating shortened URLs from lengthy web addresses. This note focuses on the comprehensive performance testing carried out to assess critical factors including speed, accuracy, durability, memory usage, and power consumption of the URL shortener.

Performance Testing Objectives: The performance testing of the URL shortener was undertaken with the following primary objectives:

1. Speed: Evaluate the system's responsiveness by measuring the time taken to generate shortened URLs under varying loads, ensuring a seamless and quick user experience.
2. Accuracy: Verify the accuracy of the URL shortener's redirection mechanism by ensuring that generated shortened URLs consistently lead to their corresponding original URLs.
3. Durability: Assess the system's robustness and stability over time, under sustained and heavy usage patterns.
4. Memory Usage: Quantify the memory footprint of the URL shortener application during various operations and traffic conditions.
5. Power Consumption: Measure the power consumption of the URL shortener application to gauge its efficiency and environmental impact.

Testing Methodology: To achieve these objectives, the performance testing was carried out through the following methodology:

1. **Speed Testing:**

* Measure the time taken to generate shortened URLs for a range of input URLs, considering different lengths and complexities.
* Perform load testing with incremental user loads to analyze response times and throughput under varying traffic scenarios.

1. **Accuracy Testing:**

* Create a comprehensive suite of test cases covering diverse URL types, including HTTP, HTTPS, URLs with query parameters, and more.
* Validate that the generated shortened URLs reliably redirect users to the correct original URLs.

1. **Durability Testing:**

* Conduct endurance testing by subjecting the system to sustained high loads for extended durations.
* Monitor system behaviour, response times, and error rates to identify potential performance degradation over time.

1. **Memory Usage Testing:**

* Utilize memory profiling tools to measure the application's memory consumption during different operations and load conditions.

1. **Power Consumption Testing:**

* Employ power monitoring tools to quantify the energy consumption of the URL shortener application during typical usage and peak loads.

Testing Environment: The performance testing was executed within a controlled environment leveraging the following tools and resources:

* Load testing tool: Apache JMeter was utilized to simulate varying levels of user traffic and record response times.
* Test environments: Both staging and production-like environments were established to replicate real-world usage scenarios.
* Profiling tools: Memory and power monitoring tools were integrated to capture memory usage and power consumption metrics.

Test Results and Recommendations: The comprehensive performance testing yielded the following results and recommendations:

* Speed: The URL shortener consistently generated shortened URLs within an acceptable time frame, with an average response time of [X] milliseconds across different load levels.
* Accuracy: All test cases successfully redirected users to the correct original URLs, establishing the URL shortener's accurate redirection mechanism.
* Durability: The system demonstrated resilience and stability during sustained high-load testing, exhibiting minimal performance degradation over extended periods.
* Memory Usage: Memory profiling revealed an efficient memory management approach, with memory consumption scaling proportionally to load.
* Power Consumption: Power consumption testing demonstrated that the URL shortener exhibited reasonable energy efficiency and environmental impact.

Recommendations: Based on the comprehensive performance testing outcomes, the following recommendations are proposed:

* Continuously monitor and optimize memory usage patterns to ensure efficient memory utilization as the application scales.
* Implement automated performance tests within the CI/CD pipeline to detect and address potential performance regressions promptly.
* Periodically review and optimize database queries and system architecture to sustain optimal performance during expansion.
* Consider energy-efficient programming practices to further enhance the URL shortener's power consumption profile.

Conclusion: The comprehensive performance testing of the Python-based URL shortener showcased its proficiency in meeting benchmarks for speed, accuracy, durability, memory usage, and power consumption. Adhering to the recommendations outlined above will facilitate the ongoing reliability and optimal performance of the URL shortener, ensuring a seamless user experience while minimizing resource usage and environmental impact.

## Test Plan/ Test Cases

Test Case 1: URL Shortening

1. **Description:** Verify that a valid long URL can be successfully shortened.
2. **Input:** Long URL: "<https://www.example.com/very/long/url>"
3. **Expected Output:** Shortened URL is generated and returned.

Test Case 2: URL Redirection

1. **Description:** Verify that a shortened URL redirects to the correct original URL.
2. **Input:** Shortened URL: "<https://short.url/abcd123>"
3. **Expected Output:** User is redirected to the original URL: "<https://www.example.com/very/long/url>".

Test Case 3: Invalid URL

1. **Description:** Verify that an error is displayed for an invalid URL.
2. **Input:** Invalid URL: "not\_a\_valid\_url"
3. **Expected Output:** Error message is displayed: "Invalid URL. Please enter a valid URL."

Test Case 4: Duplicate URL Shortening

1. **Description:** Verify that generating a short URL for the same long URL results in the same shortened URL.
2. **Input:** Long URL: "<https://www.example.com/long/url>"
3. **Expected Output:** Shortened URL is the same as previously generated.

Test Case 5: URL Validation

1. **Description:** Verify that the system validates URLs before shortening.
2. **Input:** Long URL: "invalid\_url"
3. **Expected Output:** Error message is displayed: "Invalid URL. Please enter a valid URL."

Test Case 6: Load Testing

1. **Description:** Verify that the URL shortener handles a high load of concurrent requests without significant performance degradation.
2. **Input:** Simulate [X] concurrent requests to shorten URLs.
3. **Expected Output:** Response times remain within acceptable limits, and the system maintains stability.

Test Case 7: Error Handling

1. **Description:** Verify that the system gracefully handles unexpected errors and displays appropriate error messages.
2. **Input:** Long URL: "[https://www.example.com](https://www.example.com/)"
3. **Expected Output:** Error message is displayed: "An unexpected error occurred. Please try again later."

Test Case 8: Cross-Browser Testing

1. **Description:** Verify that the URL shortener works correctly on different web browsers.
2. **Input:** Use various browsers (Chrome, Firefox, Safari, etc.) to shorten and redirect URLs.
3. **Expected Output:** Shortened URLs are generated, and redirection works consistently across different browsers.

Test Case 9: Performance Testing

1. **Description:** Measure the response time of the URL shortener under varying loads.
2. **Input:** Generate and redirect a large number of URLs concurrently.
3. **Expected Output:** Response times are within acceptable thresholds, and the system remains stable.

Test Case 10: Security Testing

1. **Description:** Verify that the URL shortener is protected against common security vulnerabilities like SQL injection and XSS attacks.
2. **Input:** Inject malicious scripts and SQL queries into the input fields.
3. **Expected Output:** The system rejects malicious inputs and displays appropriate error messages.

Test Case 11: Memory Usage Testing

1. **Description:** Monitor the memory usage of the URL shortener during URL shortening and redirection operations.
2. **Input:** Shorten and redirect a series of URLs while monitoring memory consumption.
3. **Expected Output:** Memory consumption remains within acceptable limits during different operations.

Test Case 12: Power Consumption Testing

1. **Description:** Measure the power consumption of the URL shortener application during different usage scenarios.
2. **Input:** Shorten and redirect URLs while monitoring power consumption.
3. **Expected Output:** Power consumption remains within acceptable limits during various operations.

## Test Procedure

**1. Preparations**

1.1 Set up the testing environment with the required software, including Python, frameworks, and testing tools.

1.2 Ensure the URL shortener application is deployed in the testing environment.

1.3 Prepare a set of test URLs for various test cases, including valid URLs, invalid URLs, and URLs with query parameters.

**2. Unit Testing**

2.1 Execute unit tests for individual functions and components using a testing framework .

2.2 Verify that each function produces the expected outputs for different input scenarios.

**3. Integration Testing**

3.1 Test the integration of different modules and components of the URL shortener.

3.2 Verify that data flows correctly between components, including the database.

3.3 Test the interaction between the front-end and back-end components.

**4. Functional Testing**

4.1 Test the primary functionalities of the URL shortener application.

4.2 Execute test cases for URL shortening and redirection.

4.3 Validate error handling for invalid URLs, server errors, and edge cases.

4.4 Verify the accuracy of URL redirection.

**5. Performance Testing**

5.1 Use a load testing tool (e.g., Apache JMeter) to simulate varying user loads.

5.2 Measure response times, throughput, and resource utilization under different load levels.

5.3 Analyze system behavior and ensure acceptable performance metrics.

6**. Security Testing**

6.1 Conduct security tests to identify vulnerabilities like SQL injection and XSS attacks.

6.2 Inject malicious inputs into the application and verify proper rejection or sanitization.

## Performance Outcome

Performance outcomes in the context of a URL shortener project using Python refer to the results and measurements obtained during performance testing. These outcomes help evaluate how well the system performs under different conditions, such as varying user loads, resource utilization, and response times. Here are some key performance outcomes to measure and analyze:

1. **Response Time:** Measure the time taken for the URL shortener to generate a shortened URL or redirect a user to the original URL. Lower response times indicate a more responsive and user-friendly system.
2. **Throughput:** Calculate the number of requests the URL shortener can handle per unit of time. This helps assess the system's capacity to handle concurrent users efficiently.
3. **Latency:** Analyze the delay between a user request and the system's response. Lower latency indicates quicker interactions and better user experience.
4. **Resource Utilization:** Monitor CPU usage, memory consumption, and other system resources during performance testing. Efficient resource utilization ensures optimal performance and scalability.
5. **Error Rates:** Measure the percentage of failed requests or incorrect responses under different load conditions. A low error rate indicates system reliability.
6. **Scalability:** Evaluate how well the URL shortener scales as user load increases. Measure whether response times and other performance metrics remain consistent as the system handles more requests.
7. **Stability:** Assess the system's ability to maintain acceptable performance levels over an extended duration of sustained load. Stability is crucial to ensure reliable service availability.
8. **Peak Load Performance:** Determine the system's behavior and performance under peak load conditions, which might occur during promotional campaigns or other events.
9. **Load Testing Results:** Analyze the performance metrics collected during load testing, such as response times, throughput, and resource utilization. Identify any bottlenecks or performance degradation points.
10. **Performance Degradation Thresholds:** Define thresholds for acceptable performance metrics (e.g., maximum response time, error rate) and measure whether the system meets these thresholds during testing.
11. **Comparison with Requirements:** Compare the actual performance outcomes with the performance requirements defined for the project. Ensure that the system meets or exceeds the specified benchmarks.
12. **Scalability Testing:** If applicable, test the system's ability to handle increased load by gradually increasing the number of concurrent users and observing how performance is affected.
13. **User Experience:** Consider user feedback and observations regarding the system's responsiveness and usability during performance testing.
14. **Power Consumption:** Measure the energy consumption of the URL shortener application during different usage scenarios. Optimize power consumption for energy efficiency.
15. **Memory Usage:** Analyze memory consumption patterns during URL shortening and redirection operations. Ensure that memory usage remains within acceptable limits.
16. **Redundancy and Failover Testing:** If the system has redundancy or failover mechanisms, test how well they perform under various load and failure scenarios.

# My learnings

During my time as a Python intern, I embarked on an enriching learning journey that broadened my horizons and honed my coding skills. Immersed in a dynamic and supportive environment, I had the privilege to work alongside seasoned professionals who generously shared their expertise and guided me through intricate aspects of Python programming. From crafting elegant algorithms to developing practical applications, each day presented a new challenge that I eagerly embraced. Collaborating on team projects not only deepened my understanding of version control and collaborative coding but also nurtured my ability to communicate complex technical concepts with clarity. As I delved into debugging sessions and optimized code for efficiency, I gained valuable insights into the art of problem-solving and gained a heightened appreciation for the power of Python's versatile libraries. This internship not only bolstered my proficiency in Python but also fostered a spirit of continuous learning.

As a Python intern, my learning experience was both transformative and enlightening. Throughout my internship, I delved into the intricate world of Python programming and gained insights that will undoubtedly shape my future endeavors.

Throughout the internship, I immersed myself in the world of Python, gaining an in-depth understanding of its syntax, data structures, and object-oriented programming concepts. Hands-on coding exercises and collaborative projects enabled me to grasp these fundamentals and apply them effectively to real-world scenarios. Perhaps the most enduring lesson from this experience was the value of continuous learning. Python's dynamic nature means that there is always something new to explore. Whether it's mastering the latest libraries, keeping up with best practices, or embracing emerging trends, the internship instilled in me a sense of curiosity and adaptability that will serve me well throughout my career

In conclusion, my time as a Python intern was marked by continuous growth and exploration. I emerged from this experience with not only enhanced technical proficiency but also a sense of confidence and enthusiasm to contribute meaningfully to the world of Python programming.

# Future work scope

As a Python intern, my future work scope is filled with exciting and diverse possibilities. The foundation I've established during my internship serves as a springboard for my career growth and specialization. Leveraging my proficiency in Python programming, I envision several potential pathways. There are many pathways through which I want to accelerate my learnings of python Due to shortage of time I wasn’t able to indulge myself in the possible fields where Python is required, but I would like to indulge myself in some of these fields:-

**Data Analysis** : The data manipulation and analysis skills I acquired through libraries like Pandas position me for a role as a data scientist. I intend to explore data sets, extract meaningful insights, and develop predictive models that inform critical business decisions.

**Machine Learning Engineer** : With a strong foundation in machine learning libraries such as TensorFlow and scikit-learn, I am eager to embark on a journey as a machine learning engineer. I will design, train, and deploy machine learning models that drive innovation across various industries.