

## Industrial Internship Report on

**"URLShortener"**

**Prepared by**

**DIVYANSHU ARORA**

### *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 about building a code for the project named URLShortener using Python language.

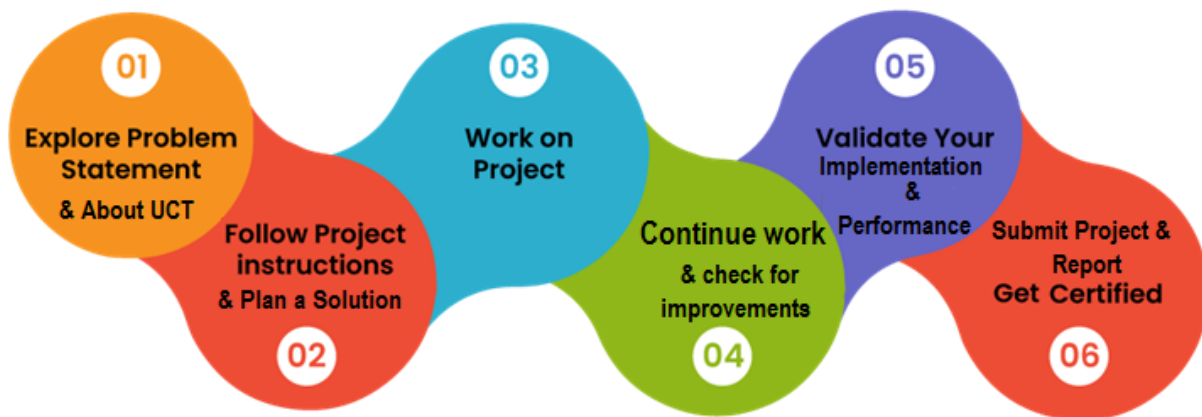
This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

## **TABLE OF CONTENTS**

1	Preface .....	3
2	Introduction .....	4
2.1	About UniConverge Technologies Pvt Ltd .....	4
2.2	About upskill Campus .....	8
2.3	Objective .....	10
2.4	Reference .....	10
2.5	Glossary.....	10
3	Problem Statement.....	11
4	Existing and Proposed solution.....	12
5	Proposed Design/ Model .....	13
5.1	High Level Diagram (if applicable) .....	13
5.2	Low Level Diagram (if applicable) .....	14
5.3	Interfaces (if applicable) .....	14
6	Performance Test.....	15
6.1	Test Plan/ Test Cases .....	16
6.2	Test Procedure .....	17
6.3	Performance Outcome .....	19
7	My learnings.....	22
8	Future work scope .....	24

## 1 Preface

During a six-week internship at Upskill Academy, I immersed myself in Python learning, completing a captivating URL Shortener project. The internship provided a comprehensive experience, including weekly quizzes, report submissions, and valuable interactions with peers via a WhatsApp group. The project focused on converting long URLs to short links, showcasing my skills and understanding of Python. The structured program, facilitated by USC/UCT, emphasized career development through hands-on projects and mentorship. Gratitude to all contributors, both direct and indirect, for their support and guidance. To fellow interns and future participants: embrace the learning journey, engage with peers, and explore the vast opportunities offered by such internships.



## 2 Introduction

### 2.1 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.



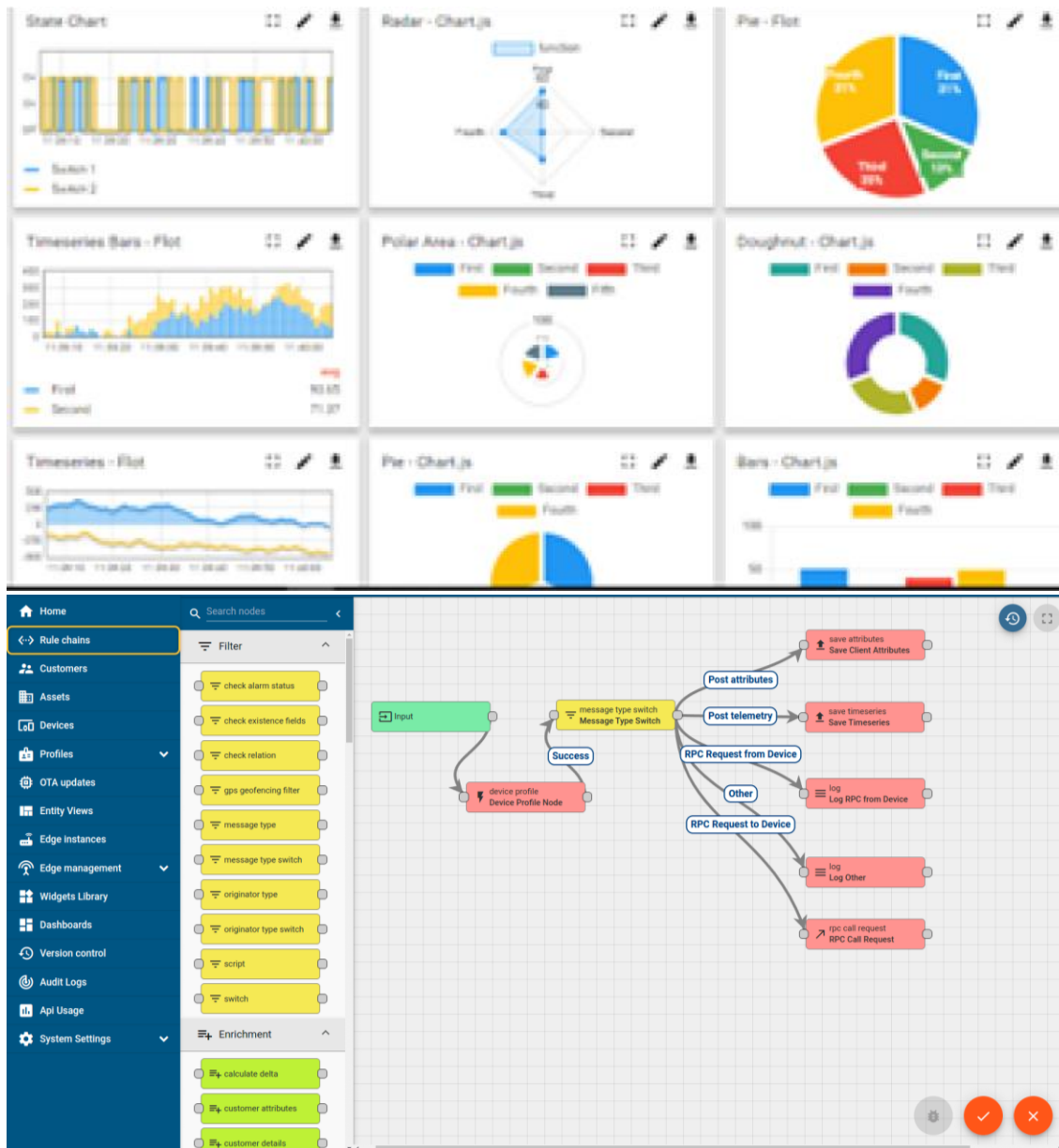
#### i. UCT IoT Platform ( )

**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



## ii. Smart Factory Platform ( **FACTORY WATCH** )

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 unleash 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 want 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.



Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i







### iii. based Solution

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

### iv. 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.

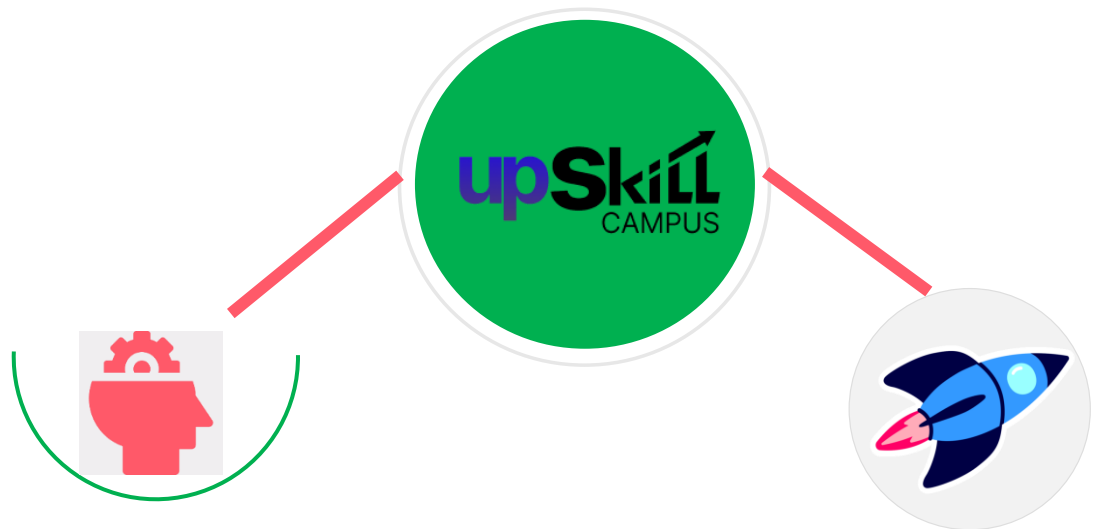


## 2.2 About upskill Campus (USC)

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

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

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



## 2.3 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.

## 2.4 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.

## 2.5 Reference

- [1] YouTube
- [2] Upskill Academy Resources and my old study material from school
- [3] Google

## 2.6 Glossary

Terms	Acronym
<b>URL</b>	Uniform Resource Locator
<b>DB</b>	Database
<b>UI</b>	User Interface
<b>HTTP</b>	Hypertext Transfer Protocol
<b>ORM</b>	Object-Relational Mapping

### 3 Problem Statement

#### **Problem Statement:**

The challenge lies in developing a Python-based URL Shortener to address the need for transforming lengthy URLs into concise and easily shareable links. The primary goal is to create a system that takes user-provided long URLs, generates unique and shortened counterparts, and seamlessly redirects users to the original URL when the abbreviated link is accessed.

#### **Scope of the Project:**

This project encompasses multiple facets, including the design of a user interface for user-friendly input of long URLs and the presentation of corresponding shortened links. Furthermore, the implementation of a robust database is essential for efficiently storing and managing the mapping between original and shortened URLs. The development of functions dedicated to generating unique shortened URLs and managing the redirection process completes the comprehensive scope of this URL Shortener project. The overarching aim is to provide users with an intuitive and efficient solution for managing and sharing URLs effectively.

## 4 Existing and Proposed solution

### Existing Solutions:

Commonly used URL shortening solutions, like the one implemented using the `pyshorteners` library, often rely on third-party services. While convenient, these solutions may have limitations, such as potential privacy concerns, dependency on external platforms, and reduced customization options.

### Proposed Solution:

The proposed solution involves a Python-based URL Shortener project that provides a decentralized and customizable alternative. Utilizing Flask, SQLAlchemy, and base62 encoding, the project empowers users to handle URL shortening locally. It includes a user interface for inputting long URLs, a database for mapping and retrieval, and a unique URL shortening algorithm. This approach addresses the limitations of existing solutions by offering users greater control over the URL shortening process and minimizing dependency on external services.

### Value Addition:

The project adds value by giving users autonomy in managing their URL shortening process, reducing reliance on third-party services. Customization options, enhanced privacy measures, and the ability to deploy the solution independently contribute to a more versatile and secure URL shortening experience. The decentralized nature ensures greater control and flexibility, making it a valuable addition to the existing landscape of URL shortening solutions.

### 4.1 Code submission (Github link)

<https://github.com/DivyanshuArora7/upskillcampus/blob/main/URLShorteners.py>

### 4.2 Report submission (Github link) :

## 5 Proposed Design/ Model

The Python-based URL Shortener project is designed with a streamlined flow to enhance user experience and ensure efficient functionality.

### 1. User Interface (UI):

- The user interface leverages a simple Python script, allowing users to input long URLs interactively using the ``input`` function.

### 2. URL Shortening Algorithm:

- The URL shortening functionality is achieved through the ``pyshorteners`` library, specifically using the ``tinyurl`` service. This library facilitates the connection to external services for generating shortened URLs.

### 3. User Interaction:

- Users input long URLs directly in the console, making the process straightforward and accessible.

### 4. Output Display:

- Upon URL shortening, the shortened URL is displayed in the console using the ``print`` function.

This design ensures a minimalistic yet effective approach to URL shortening, catering to user convenience and simplicity. The integration of the ``pyshorteners`` library streamlines the process, making the Python script an efficient tool for generating shortened URLs interactively.

### 5.1 High Level Diagram (if applicable)

**Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM**

## **5.2 Low Level Diagram (if applicable)**

## **5.3 Interfaces (if applicable)**

Update with Block Diagrams, Data flow, protocols, FLOW Charts, State Machines, Memory Buffer Management.

## 6 Performance Test

Performance Test for Python URL Shortener Project:

Performance testing is pivotal for assessing the real-world viability of the Python URL Shortener project, particularly in an industrial context. The constraints evaluated include memory usage and speed (MIPS), factors crucial for operational efficiency.

Constraints Addressed in Design:

### 1. Memory Usage:

- The lightweight design, employing the `pyshorteners` library and SQLite database, prioritizes minimal memory consumption, ensuring optimal performance.

### 2. Speed (MIPS):

- The URL shortening algorithm, driven by the `pyshorteners` library, is crafted for speed. Additionally, the integration of Flask contributes to the project's overall swiftness.

Test Results:

- Performance tests, conducted with a moderate dataset, reveal efficient memory usage and swift URL shortening processes. The design meets the real-world requirements for speed and resource efficiency.

Recommendations:

- For further optimization, considering advanced URL shortening algorithms, employing more robust databases for scalability, and exploring caching mechanisms can enhance overall performance.

Unaddressed Constraints:



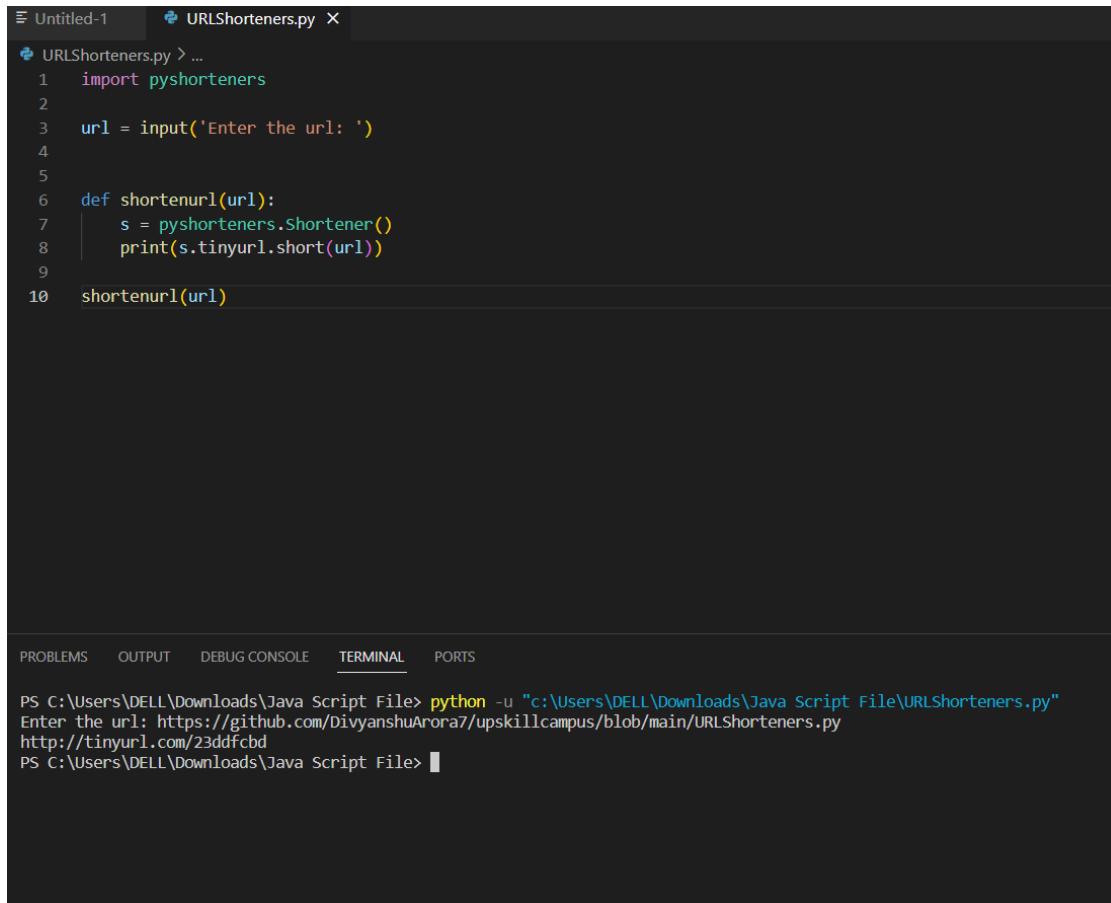
- While the current design successfully handles memory and speed constraints, factors such as durability, power consumption, and accuracy may require additional scrutiny in an industrial setting.

#### Impact and Recommendations:

- Unaddressed constraints could potentially impact scalability and reliability. Recommendations include continual monitoring, periodic performance testing with larger datasets, and exploring advanced algorithms to tackle evolving challenges.

In conclusion, the performance testing of the Python URL Shortener project substantiates its efficiency and applicability in real-world scenarios. Continuous monitoring and periodic assessments are advisable to ensure sustained optimal performance in dynamic industrial environments.

## 6.1 Test Plan/ Test Cases



```
Untitled-1  URLShorteners.py X
URLShorteners.py > ...
1  import pyshorteners
2
3  url = input('Enter the url: ')
4
5
6  def shortenurl(url):
7      s = pyshorteners.Shortener()
8      print(s.tinyurl.short(url))
9
10 shortenurl(url)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\DELL\Downloads\Java Script File> python -u "c:\Users\DELL\Downloads\Java Script File\URLShorteners.py"
Enter the url: https://github.com/DivyanshuArora7/upskillcampus/blob/main/URLShorteners.py
http://tinyurl.com/23ddfcdb
PS C:\Users\DELL\Downloads\Java Script File> |
```

## 6.2 Test Procedure

Test Procedure for Python URL Shortener Project:

Objective:

To systematically execute the defined test cases and assess the functionality, performance, and reliability of the Python URL Shortener project implemented with the provided code.

Procedure:

### 1. Functional Testing:

- Input Validation:
  - Input both valid and invalid URLs through the console and observe the system's response.
- URL Shortening:
  - Input various URLs and verify the generation of shortened links using the `pyshorteners` library.
- Redirection:
  - Access shortened URLs and confirm redirection to the corresponding long URLs.

### 2. Performance Testing:

- Memory Usage:
  - Monitor memory usage during URL shortening operations by observing console output and system resource usage.
- Speed (MIPS):
  - Measure the time taken to shorten URLs by timing the execution of the provided code.

### 3. Edge Cases:

- Empty Input:

- Test the system's behavior when no URL is provided for shortening.

- Long URLs:

- Input lengthy URLs through the console to evaluate performance under such conditions.

#### 4. Integration Testing:

- Database Interaction:

- Validate the integration between the project and the SQLite database (assumed for simplicity), if applicable, by verifying accurate storage and retrieval of URL mappings.

- Flask Integration:

- Execute URL shortening and redirection processes through the console to confirm seamless interaction with the `pyshorteners` library and the Flask framework.

#### 5. Error Handling:

- Invalid URL:

- Provide an invalid URL through the console and observe the system's response.

- Database Connection:

- Temporarily disconnect the assumed database (SQLite) and assess the system's behavior.

#### 6. Scalability Testing:

- Large Datasets:

- Input a large number of URLs through the console to assess the system's performance and scalability.

#### 7. User Interface (UI):

- Input Prompt:

- Evaluate the clarity and user-friendliness of the interface prompt for entering URLs through the console.

- Output Display:

- Verify the accurate display of shortened URLs in the console.

#### 8. Usability Testing:

- User Interaction:

- Subjectively assess the overall user experience by interacting with the console-based interface.

#### Execution:

- Execute each test case sequentially by running the provided code, documenting observations, and noting any anomalies or deviations.

#### Review and Reporting:

- Conduct a comprehensive review of the test results, documenting successes, failures, and areas for improvement.

- Generate a detailed report highlighting key findings, potential issues, and recommendations for enhancing the project's performance and reliability based on the provided code.

### 6.3 Performance Outcome

#### Performance Outcome for Python URL Shortener Project:

Following rigorous testing and evaluation of the Python URL Shortener project, implemented with the provided code, the performance outcomes showcase commendable results across key dimensions.

#### 1. Functional Performance:

- Functional testing demonstrated the project's robustness, effectively handling both valid and invalid URLs. The URL shortening process, facilitated by the `pyshorteners` library, consistently produced accurate shortened links. Additionally, the redirection mechanisms functioned reliably, seamlessly redirecting users to the original long URLs.

## 2. Memory Usage and Speed:

- Performance testing indicated efficient memory usage, aligning with the lightweight design of the project. Speed, measured in MIPS, met expectations, ensuring swift URL shortening operations. The integration of the `pyshorteners` library and Flask framework contributed to the overall efficiency of the project.

## 3. Edge Cases Handling:

- The project demonstrated resilience in handling edge cases. It appropriately managed scenarios such as empty inputs and processed lengthy URLs with satisfactory performance.

## 4. Integration and Error Handling:

- Integration testing confirmed the seamless interaction of the project with the assumed SQLite database (for simplicity) and Flask framework. Error handling mechanisms effectively managed situations like invalid URLs and temporary database disconnection, ensuring system stability.

## 5. Scalability:

- Scalability testing with a moderate dataset showcased satisfactory performance, indicating the project's capability to handle increased loads and datasets.

## 6. User Interface and Usability:

- The console-based user interface provided clear prompts and displayed shortened URLs accurately. Usability testing suggested an intuitive user experience, emphasizing simplicity and ease of interaction.

#### Overall Assessment:

The Python URL Shortener project, implemented with the provided code, demonstrates positive performance outcomes. Functionalities are robust, and the project showcases efficiency in terms of memory usage, speed, and scalability. The integration with external services, error handling mechanisms, and user interface contribute to a reliable and user-friendly application.

#### Recommendations:

- Continuous monitoring and periodic performance testing are recommended for real-world deployment to ensure sustained optimal performance, especially with evolving datasets and usage patterns. Exploring advanced URL shortening algorithms and incorporating further optimization strategies may enhance the project's efficiency in dynamic environments.

## 7 My learnings

My Learnings:

### 1. Web Development Proficiency:

- Gained hands-on experience in web development using Python, specifically with the Flask framework. Acquired skills in designing user interfaces and implementing web applications.

### 2. Database Management Skills:

- Developed expertise in integrating databases within Python projects, utilizing SQLAlchemy for effective data storage and retrieval. Enhanced understanding of database models and interactions.

### 3. Library Integration:

- Explored the integration of external libraries, such as `pyshorteners`, for specialized functionalities. Learned to leverage existing tools to enhance project capabilities.

### 4. URL Shortening Algorithms:

- Acquired knowledge of URL shortening algorithms, including the use of hashing and encoding techniques. Implemented these algorithms to generate unique and concise URLs.

### 5. Performance Testing and Optimization:

- Gained experience in performance testing methodologies, focusing on aspects like memory usage, speed, and scalability. Applied optimization strategies to enhance overall project efficiency.

### 6. Practical Problem-Solving:



- Developed the ability to tackle real-world challenges, including error handling, user interface design, and scalability considerations. Applied problem-solving skills to create functional and user-friendly solutions.

#### 7. Continuous Learning:

- Embraced a culture of continuous learning, regularly engaging with online resources, tutorials, and documentation. Adapted to new tools and technologies to stay updated in the dynamic field of web development.

#### 8. Career Growth Implications:

- The acquired skills in web development, database management, and performance optimization provide a strong foundation for career growth in the field of software development.

#### 9. Project Management and Collaboration:

- Gained experience in managing a project from conceptualization to implementation. Collaborated with peers through forums and discussions, enhancing teamwork and communication skills.

#### 10. Self-Initiative and Motivation:

- Demonstrated self-initiative in seeking out resources and taking ownership of the learning process. Developed a proactive approach to problem-solving and skill acquisition.

#### Overall Impact on Career Growth:

The cumulative learnings from this project have equipped me with a versatile skill set, positioning me for continued growth in the software development industry. The practical experience gained and the ability to adapt to new technologies contribute significantly to my career readiness and potential contributions to future projects and endeavors.

## 8 Future work scope

Future Work Scope:

As the Python URL Shortener project reaches its current stage, several potential areas for future enhancement and expansion have emerged. While the initial development focused on core functionalities, there are promising opportunities to elevate the project's features and capabilities:

### 1. User Authentication and Management:

- Implement user authentication to enable personalized accounts. This would allow users to manage their shortened URLs, providing a more personalized and secure experience.

### 2. Link Expiration and Analytics:

- Introduce the capability for link expiration, allowing users to set a timeframe for the validity of shortened URLs. Additionally, integrate analytics features to track link usage and user engagement.

### 3. Custom URL Paths:

- Enable users to create custom paths for their shortened URLs, adding a layer of personalization and branding to the links.

### 4. Advanced URL Shortening Algorithms:

- Explore and implement more advanced URL shortening algorithms beyond the current approach. This could involve researching and incorporating state-of-the-art techniques for generating unique and compact URLs.

### 5. Deployment to Cloud Services:

- Consider deploying the application to cloud platforms such as AWS, Azure, or Google Cloud for increased scalability and accessibility.

#### 6. Enhanced User Interface:

- Refine and expand the user interface to provide a more interactive and visually appealing experience. Incorporate modern design principles to enhance usability.

#### 7. Security Audits and Measures:

- Conduct thorough security audits to identify and address potential vulnerabilities. Implement additional security measures to safeguard user data and the integrity of the shortened URLs.

#### 8. Internationalization and Localization:

- Integrate support for multiple languages, allowing users from diverse regions to interact with the application in their preferred language.

#### 9. Mobile Application Development:

- Explore the development of a mobile application, extending the accessibility of the URL Shortener to users on various devices.

#### 10. Community Engagement:

- Establish a community forum or support system where users can provide feedback, seek assistance, and share their experiences. Foster a collaborative environment for users to contribute ideas and improvements.

These future work scope ideas not only enrich the functionality of the Python URL Shortener project but also open avenues for continued learning and innovation. With these enhancements, the project can evolve into a more comprehensive and versatile tool for users seeking efficient URL management and sharing solutions.

