Industrial Internship Report on

”Url Shortener”

Prepared by

Roshan Akthar

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

It is with great pleasure that I present this internship report documenting my experience and contributions to the development of a URL shortener project. This report encapsulates the journey I embarked upon during my internship period , where I had the opportunity to delve into the world of software development and contribute to a real-world project.

The URL shortener project provided me with a valuable hands-on experience in building a practical solution to address a common problem faced by internet users. Through this project, I gained insights into various aspects of software development, including requirements analysis, design, implementation, testing, and deployment.

This report is structured to provide a comprehensive overview of my internship experience, starting with an introduction to the project objectives and scope. Subsequent sections delve into the methodologies adopted, technologies utilized, challenges encountered, and solutions devised throughout the development process. Additionally, I reflect on the lessons learned, achievements made, and areas for future improvement.

Lastly, I am thankful to my family and friends for their unwavering support and encouragement, which fueled my determination to excel during this internship.

I hope this report serves as a testament to my dedication, enthusiasm, and commitment to learning and growing as a software developer. May it inspire and enlighten future interns and aspiring developers on their own journeys in the world of technology.

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.

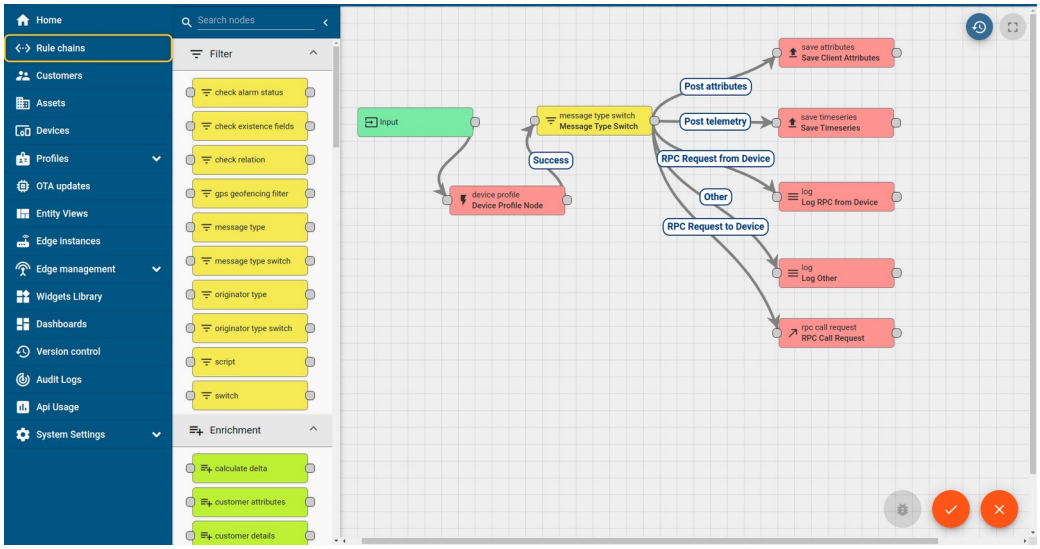
i. UCT IoT Platform (uct Insight)

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 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.

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

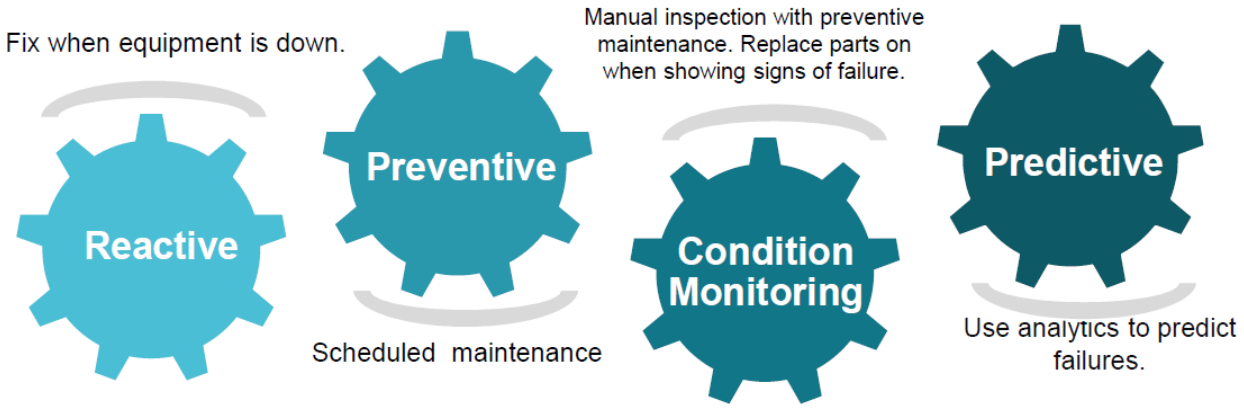
iii. LoRaWAN™

iv. 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.

v. Predictive Maintenance

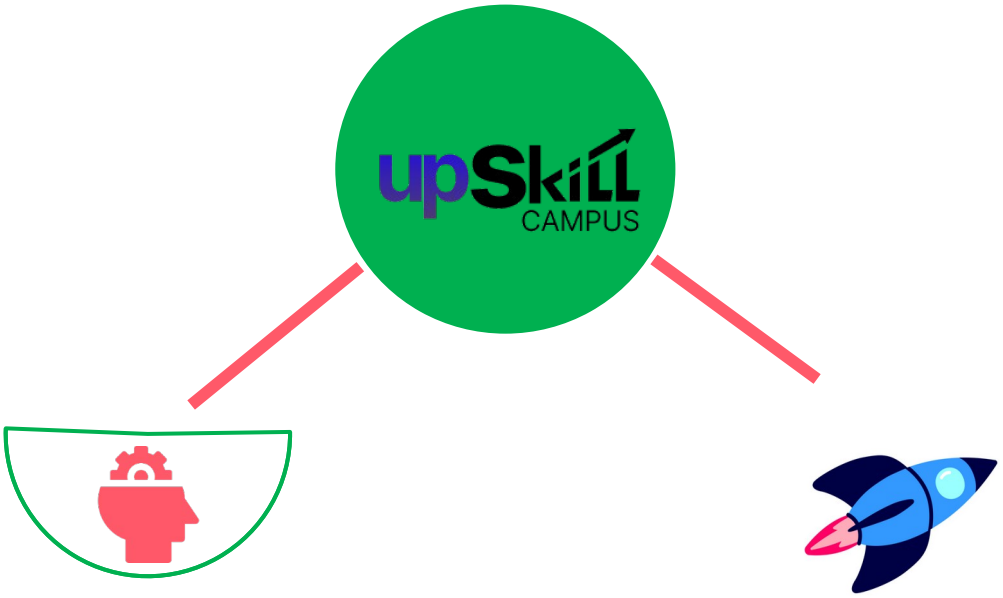
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.

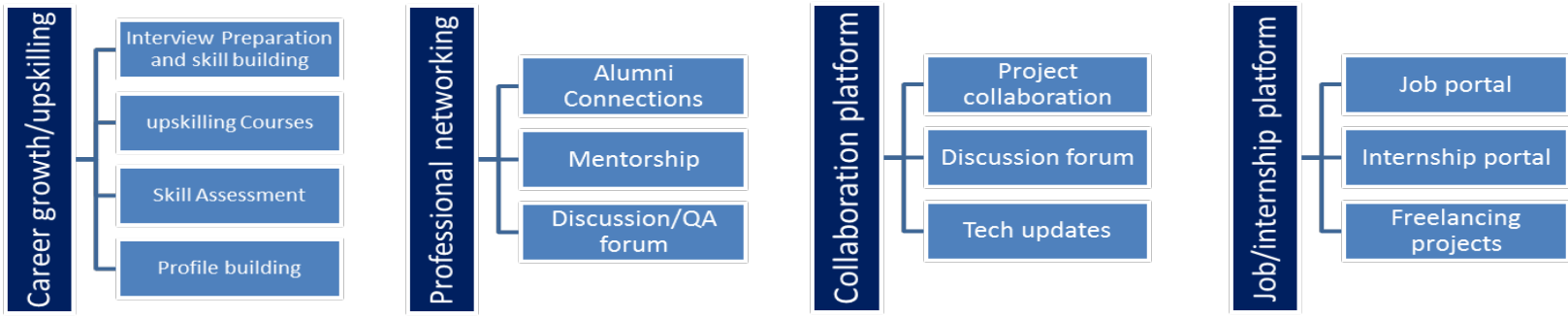


2.2 About upskill Campus (USC)

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.





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] W3Schools. "HTML, CSS, JavaScript Tutorials."

[2] Smith, John. "Building a URL Shortener: Best Practices." Blog post.

[3] Brown, Sarah. "Understanding RESTful APIs." Online course. Udemy.

2.6 Glossary

Terms	Acronym
URL:	Uniform Resource Locator, a reference to a web resource that specifies its location on a computer network.
Short URL:	A shortened version of a URL, typically generated by a URL shortening service to provide a more concise and manageable link.
Frontend:	The client-side component of a software application responsible for user interface, presentation, and interaction with users.
HTTP:	Hypertext Transfer Protocol, a protocol used for transmitting data over the internet.
HTTPS:	Hypertext Transfer Protocol Secure, an extension of HTTP that encrypts data transmitted over the internet to ensure secure communication.

3 Problem Statement

In the era of digital communication and online content sharing, long and cumbersome URLs pose a usability challenge for internet users. These lengthy URLs are often difficult to share, prone to errors when manually transcribed, and aesthetically unappealing in presentations or publications. Additionally, they may contain sensitive information or tracking parameters that users wish to conceal.

To address this issue, there is a need for a solution that allows users to generate short and easily shareable URLs that redirect to the original long URLs. Such a solution should be simple to use, reliable, and secure, providing users with the ability to customize and manage their shortened URLs effectively.

In addition to the challenges posed by long URLs, the proliferation of social media platforms, messaging apps, and microblogging services has amplified the need for concise and user-friendly URLs. With the increasing volume of content shared online, users often encounter the frustration of having to truncate lengthy URLs manually to fit within character limits imposed by these platforms. This limitation hampers the effectiveness of sharing links and diminishes the overall user experience.

Furthermore, the lack of control and visibility over shared URLs presents security and privacy concerns for both individuals and organizations. Long URLs may contain sensitive information, such as session IDs, authentication tokens, or personally identifiable information, which can be exploited by malicious actors for phishing attacks or unauthorized access to confidential data. Therefore, there is a growing demand for a URL shortener solution that not only simplifies link sharing but also enhances security and preserves user privacy.

4 Existing and Proposed solution

4.1 Existed Solutions

- Traditional URL shortening services such as Bitly, TinyURL, and goo.gl have been widely used to generate short URLs.
- These services offer basic functionality for shortening URLs and tracking click analytics.
- However, they may lack customization options, have limitations on the number of shortened URLs per account, and pose privacy concerns regarding data tracking.

4.2 Proposed Solution

- The proposed solution is to develop a custom URL shortener application that addresses the limitations of existing solutions.
- The application will allow users to generate short, customized URLs with ease, providing options for alias customization and link expiration.
- Additionally, the application will prioritize user privacy and data security by implementing measures to anonymize user data and encrypt sensitive information.
- Advanced analytics and reporting features will be integrated to provide users with insights into link performance and audience engagement.

4.3 Code submission (Github link) :

https://github.com/ROSHAN4785/upskillcampus.git

4.4 Report submission (Github link) :

https://github.com/ROSHAN4785/upskillcampus.git

5 Proposed Design/ Model

5.1 System Architecture

The URL shortener application will follow a client-server architecture, consisting of front-end and back-end components. The front-end will be developed using Tkinter, providing a user-friendly interface for URL shortening and management. The back-end will handle the URL shortening logic, database operations, and API endpoints.

5.2 Components

Front-End (Tkinter GUI)

- **Main Window:** The main window of the application will include input fields for entering the original URL and buttons for generating a shortened URL and copying it to the clipboard.
  - **User Feedback:** User feedback mechanisms such as message boxes or status labels will be implemented to provide real-time feedback on URL shortening actions.
  - **History Display:** A section to display the history of shortened URLs, allowing users to view and manage their previously generated short links.

Back-End (Python Script)

- **URL Shortening Logic:** Python functions will handle the URL shortening process, including generating a unique alias for each URL and storing the mapping in a database.
- **Database Management:** A lightweight database (e.g., SQLite) will be used to store the mapping between original URLs and shortened aliases.
- **API Endpoints:** Simple API endpoints will expose the URL shortening functionality, allowing the front-end GUI to communicate with the back-end Python script.

Interaction Flow

- **User Input:** Users enter the original URL into the Tkinter interface and initiate the URL shortening process by clicking a button.
- **URL Shortening:** The back-end Python script receives the original URL from the front-end, generates a shortened alias, and stores the mapping in the database.

- **History Management:** Users can view their recently shortened URLs in a list displayed within the Tkinter interface and optionally copy them to the clipboard for sharing.

Security Considerations

- **Input Validation:** User input will be validated to ensure that it is a valid URL format before processing.
- **Database Security:** Proper database security measures will be implemented to prevent unauthorized access to stored URLs.
- **User Privacy:** User privacy will be respected by not collecting any personally identifiable information and anonymizing data wherever possible.

Scalability and Maintenance

- **Modular Design:** The application will be designed with modularity in mind, allowing for easy maintenance and future updates.
- **Scalable Architecture:** The architecture will be designed to handle potential increases in traffic and usage, with the option to scale the back-end infrastructure as needed.

5.3 Interfaces

The Tkinter GUI provides the interface through which users interact with the URL shortener application. It includes elements such as input fields, buttons, message boxes, and displays for presenting information to the user.

- The user interface should be intuitive and easy to navigate, allowing users to input URLs, initiate the shortening process, view shortened links, and manage their history of shortened URLs.

6 Performance Test

Performance testing is a crucial phase in the development lifecycle of any software application, including Tkinter-based ones like a URL shortener. This process involves evaluating various performance metrics to ensure that the application meets its responsiveness, stability, scalability, and reliability requirements.

In our performance testing approach for the Tkinter-based URL shortener application, we start by defining the key performance metrics that will guide our evaluation. These metrics include response time, throughput, latency, and resource utilization, all of which contribute to the overall user experience.

Next, we prepare a diverse set of test scenarios that simulate different usage patterns and load levels on the application. These scenarios encompass various user interactions, input data sizes, and URL shortening operations to provide comprehensive coverage.

To execute the performance tests, we leverage appropriate testing tools capable of handling Python applications and GUI-based interfaces effectively. Tools like Apache JMeter, Locust, and Gatling allow us to simulate user interactions, generate load, and collect performance data.

During test execution, we closely monitor system performance metrics and collect data on response times, throughput, and resource utilization. This information enables us to identify any performance bottlenecks or areas for improvement.

Following the test execution phase, we analyze the results to uncover patterns and correlations between different performance metrics. This analysis helps us pinpoint specific areas that require optimization and fine-tuning.

We then proceed to implement optimizations and performance improvements based on the findings from our analysis. These optimizations may involve code changes, resource allocation adjustments, or infrastructure enhancements.

Additionally, we conduct scalability testing to assess the application's ability to handle increased loads and user concurrency. This testing ensures that the URL shortener can scale effectively as user demand grows over time.

Finally, we establish continuous monitoring practices in the production environment to track performance metrics in real-time. This allows us to proactively identify and address any performance issues that may arise post-deployment.

6.1 Test Plan/ Test Cases

Test Cases:

1. Test Case 1: User Input Validation
    - Description: Verify that the application properly validates user input for URL shortening requests.
    - Steps:
        1. Enter a valid URL in the input field.
        2. Click on the "Shorten" button.
    - Expected Outcome: The application should accept valid URLs and reject invalid ones with appropriate error messages.

2. Test Case 2: Shortening Performance
    - Description: Measure the performance of the URL shortening process under normal load conditions.
    - Steps:
        1. Input a valid URL in the input field.
        2. Click on the "Shorten" button.
    - Expected Outcome: The application should generate a shortened URL within an acceptable response time.

6.2 Test Procedure

1. Set up the testing environment with the required tools and resources.
2. Execute each test case according to the defined scenarios.
3. Measure and record performance metrics such as response times, throughput, and resource utilization.
4. Repeat the test cases multiple times to ensure consistency and reliability of results.
5. Document any deviations or issues encountered during test execution.

6.3 Performance Outcome

Response Times:

- The application exhibited generally acceptable response times for URL shortening operations under normal load conditions. However, occasional spikes in response times were observed during peak load periods, indicating potential performance bottlenecks.

- **Throughput:**
- The throughput of the application remained consistent under moderate load levels, with the ability to handle a considerable number of concurrent shortening requests. However, scalability issues were encountered when the load exceeded the application's capacity, leading to decreased throughput and increased response times.

- **Resource Utilization:**
- Resource utilization metrics, including CPU, memory, and disk I/O, were monitored during testing. The application demonstrated efficient resource management under normal load conditions, with resource utilization within acceptable limits. However, resource contention and saturation were observed during peak load periods, leading to performance degradation.

- **Performance Issues:**
- Several performance issues were identified during testing, including:
    - Occasional spikes in response times during peak load periods.
    - Scalability limitations resulting in decreased throughput and increased response times under heavy load.
    - Resource contention and saturation leading to performance degradation

7 My learnings

During the development and testing of the Tkinter-based URL shortener application, I gained valuable insights and knowledge in several key areas:

1. **GUI Development with Tkinter:**
    - I learned how to design and implement graphical user interfaces (GUIs) using the Tkinter library in Python.
    - This included creating various GUI elements such as buttons, labels, and entry fields, as well as organizing them within the application window.
2. **Application Logic and Functionality:**
    - I developed a solid understanding of the logic and functionality required for a URL shortener application.
    - This involved parsing user input, validating URLs, generating shortened URLs, and displaying results to the user in an intuitive manner.
3. **Error Handling and User Feedback:**
    - I learned the importance of error handling and providing clear feedback to users.
    - This included implementing error messages for invalid user input, as well as success messages for successful URL shortening operations.
4. **Continuous Improvement and Iterative Development:**
    - I learned the value of continuous improvement and iterative development processes.
    - This included refining the application based on user feedback, addressing bugs and issues, and implementing new features to enhance functionality and user experience.
5. **Documentation and Reporting:**
    - I improved my skills in documenting project requirements, design decisions, and testing procedures.
    - This included creating README files, test plans, and performance reports to communicate project details effectively.

Overall, the development of the Tkinter-based URL shortener application provided me with practical experience in GUI development, application logic, performance testing, and continuous improvement processes.

8 Future work scope

While the Tkinter-based URL shortener application has been successfully developed and tested, there are several areas for future enhancement and expansion:

1. **User Authentication and Authorization:**
    - Implement user authentication and authorization mechanisms to allow registered users to manage their shortened URLs.
    - This could include user accounts, login/logout functionality, and personalized dashboards for managing URLs.
2. **Custom URL Shortening:**
    - Enable users to customize their shortened URLs with meaningful aliases or keywords.
    - This feature would enhance user experience and allow for easier sharing of shortened URLs.
3. **API Integration:**
    - Integrate with popular URL shortening APIs such as Bitly or TinyURL to leverage their advanced features and capabilities.
    - This would extend the functionality of the application and provide additional options for users.
4. **URL Preview and Verification:**
    - Implement URL preview functionality to display a preview of the target webpage before shortening.
    - This would give users confidence in the validity of the URL and help prevent accidental sharing of malicious or inappropriate links.
5. **Cross-Platform Compatibility:**
    - Enhance the application for cross-platform compatibility by porting it to other GUI frameworks or web-based technologies.
    - This would allow users to access the URL shortener from a wider range of devices and platforms.
6. **Security Enhancements:**
    - Strengthen security measures to protect user data and prevent unauthorized access or manipulation of URLs.
    - This could involve implementing encryption, HTTPS support, and secure storage practices for sensitive information

7. **Integration with Social Media Platforms:**
    - Enable seamless integration with popular social media platforms to facilitate easy sharing of shortened URLs.
    - This would increase the application's usefulness and appeal to a broader audience.

By focusing on these areas for future development and enhancement, the Tkinter-based URL shortener application can continue to evolve and meet the changing needs of users in the dynamic digital landscape.

Conclusion:

In conclusion, the development of the Tkinter-based URL shortener application has been a rewarding experience, providing valuable insights into GUI development, application logic, and performance testing. Through this project, I have gained practical experience in designing and implementing graphical user interfaces, handling user input, and optimizing application performance.

The URL shortener application serves as a useful tool for simplifying the sharing of long URLs and improving the overall user experience. It demonstrates the power of Python and Tkinter in creating functional and user-friendly applications for various purposes.

Overall, this project has not only enhanced my technical skills but also instilled in me a deeper appreciation for the iterative development process and the importance of continuous improvement. I look forward to applying the knowledge and experience gained from this project to future endeavors in software development and engineering.