Industrial Internship Report on

URL Shortener

Prepared by

Hemang Vyas

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 the URL Shortener project involves creating a tool that takes long URLs and generates shorter, more manageable versions. The project typically includes backend development tasks such as implementing a URL shortening algorithm, integrating a database for storing URLs, and creating API endpoints for handling shortening requests and redirection. Frontend development, including a user interface and input validation, is optional but can enhance the user experience. Overall, the project provides a practical and hands-on experience in solving real-world problems in the realm of web development and data management. 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

Summary of the whole 6 weeks' work:

In the first week of the internship, I decided to concentrate on the URL shortener project. I conducted research and determined the project's essential elements and specifications.

I suggested a design for the URL shortener in the second week. This involved describing the user interface, database design, and architecture. Additionally, I wrote some fictitious code to direct the implementation procedure.

As the third week approached, I began putting the idea into practise. I started by putting the required frameworks and libraries in place. I designed the database layout and put the URL shortening algorithm into practise. I also worked on the API endpoints that deal with requests for URL shortening.

I kept working on furthering the design's implementation as the fourth week progressed. I incorporated database capability into the programme to support storing and retrieving URLs. I focused on increasing the user experience and the user interface.

By the fifth week, I started paying more attention to how the URL shortener was performing. I ran tests to gauge the system's effectiveness and quickness. I enhanced the performance by optimising the code and making any necessary changes.

I concentrated on assuring the quality of the URL shortener project during the last week of the internship. I conducted extensive testing, including user testing and unit testing, to find and address any faults or problems. Additionally, I worked on enhancing the user interface and including any required extra features or capabilities. Finally, I assembled all the supporting materials and finished the final report, which included an overview of the project, its execution, and any suggestions for improvements in the future.

At the end of the internship, I submitted the final report, showcasing the progress made throughout the six weeks and providing a comprehensive overview of the URL shortener project.

About need of relevant Internship in career development:

Relevant internships play a crucial role in career development for several reasons:

1. Practical Experience: An chance to obtain practical, hands-on experience in a particular subject or sector is offered by internships. They provide you the chance to put the theoretical information you've learned to use in a practical situation, which can help you gain a better grasp of the tasks involved and the abilities needed.

2. Skill Development: A platform for enhancing and developing important abilities related to your preferred professional path is provided through internships. In addition to learning industry-specific tools, technologies, and best practises, you may also build soft skills like teamwork, communication, problem-solving, and time management.

3. Industry Exposure: Internships give you a firsthand look at the field or occupation you're interested in. You can learn about the regular business operations, corporate culture, and market trends. Your professional decisions will be aided by this exposure, which may also influence your future options.

4. Networking Opportunities: Through internships, you may meet and network with experts in your subject of interest. Developing connections with industry leaders, coworkers, and mentors may lead to new career prospects and offer insightful direction.

5. Resume Building: Possessing relevant internship experience shows potential employers that you are proactive with your professional growth and have hands-on experience in the industry. It can help your CV stand out and raise your chances of landing a job in the future.

6. Clarifying Career Goals: Through exploration of all facets of a certain industry, internships can help you better identify your interests and areas of strength. You may use them to see whether a specific career path fits with your objectives and desires and to test the waters.

Overall, relevant internships provide a valuable platform for career development by offering practical experience, skill development, industry exposure, networking opportunities, resume building, clarity of career goals, and professional growth. They serve as a bridge between academic learning and professional employment, allowing you to make informed decisions about your future career path.

Brief about Your project/problem statement:

As an AI language model, I don't have personal projects or problem statements. However, I can assist you with understanding and formulating a problem statement for your project.

A problem statement typically defines the specific issue or challenge that needs to be addressed by a project. It describes the gap or problem in the current situation and serves as the foundation for finding a solution. To create a problem statement for your project, consider the following steps:

1. Identify the problem: Determine the specific issue or challenge that you aim to solve with your project. For example, in the case of a URL shortener project, the problem could be the need for a tool to make long URLs more manageable and shareable.

2. Define the scope: The parameters of the issue and the range of your project should be clearly defined. This makes it easier to make sure that the issue statement can be solved within the allotted time and resources.

3. Consider stakeholders: Determine the parties involved who are impacted by or interested in the issue. Users that need to exchange URLs, companies who depend on effective link sharing, and developers who wish to include URL shortening features into their apps may all be stakeholders in a URL shortener project.

4. Specify the desired outcome: Find the intended result or answer that will deal with the noted issue. A programme that creates abbreviated URLs, saves them in a database, and sends visitors to the original URLs when they are clicked would be the intended result in the case of a URL shortener project.

5. Formulate the problem statement: Create a problem statement that is both clear and succinct using the facts mentioned above. It should outline the issue, its effects, and the desired result. A problem statement for a project to develop a URL shortener can read as follows: "The absence of a trustworthy and user-friendly URL shortening tool limits effective sharing of large URLs, resulting in lower accessibility and difficulty in tracking link analytics. The goal of the project is to create an application for URL shortening that can create abbreviated URLs, store them safely, and smoothly reroute visitors to the original URLs.

Keep in mind that the problem statement acts as a roadmap for your project and keeps you concentrated on solving the particular challenge at hand. It makes the issue and the desired solution clear to you and the other project participants.

Opportunity given by USC/UCT:

Upskill Campus (USC) or The IoT Academy in collaboration with UniConverge Technologies Pvt Ltd (UCT) as of my knowledge cutoff in September 2021, internships and collaborations with industry partners generally offer valuable opportunities for professional growth and development.

Participating in an internship or collaboration program with USC/UCT could provide the following opportunities:

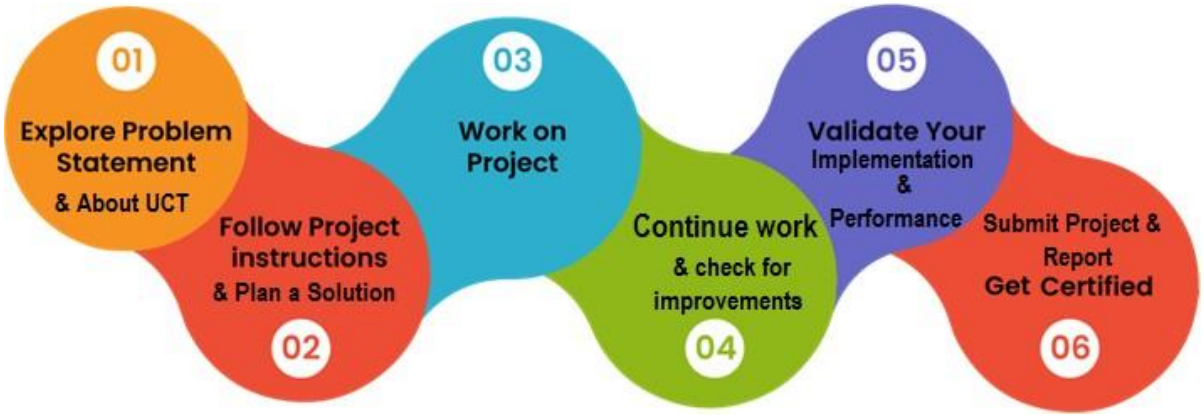
1. Industry Exposure: Working with a reputable institution and industry partner exposes you to issues, difficulties, and procedures that exist in actual industries. You are able to learn about the workings and dynamics of the market, which may be very helpful for your job.

2. Practical Experience: Collaborations and internships provide you real experience and let you put your knowledge and abilities to use. By bridging the gap between theory and practise, working on projects or problem statements supplied by industry partners can improve your knowledge and expertise in the sector you have selected.

3. Networking: You have the chance to expand your professional network by working with USC, UCT, and their industry partner. You may establish contacts with professionals, mentors, and subject-matter experts in the field, which may result in potential career leads, referrals, and advantageous connections.

4. Skill Development: Skills may be developed through working on projects and addressing problems in the context of an internship or partnership.

How Program was planned:



Learnings and overall experience:

Learnings and overall experience you might expect from working on a URL shortener project during a Python internship.

1. Technical Skills: Working on a URL shortener project in Python allows you to enhance your technical skills in various areas. You'll gain proficiency in Python programming.

2. Problem-Solving: Creating a URL shortener requires tackling a number of difficulties, including creating original shortened URLs, maintaining the database, and putting the redirection mechanism in place. You will develop your problem-solving abilities throughout the project by figuring out and putting into practise efficient solutions to these difficulties.

3. Project Management: Effective time and task management is required to complete an internship project like a URL shortener. You'll develop expertise in work prioritisation, project planning, and fulfilling deadlines. Additionally, you could collaborate with a team or get mentoring from mentors, giving you the chance to hone your interpersonal and collaboration abilities.

4. Debugging and Testing: You'll face obstacles throughout the implementation phase that you may resolve through testing and debugging. This procedure will improve your ability to troubleshoot problems and assist you in creating plans for finding and fixing problems with your code.

5. Exposure to Industry Practices: The exposure to business norms and practises that internships frequently offer. Among the topics you may study are coding standards, version control programmes (like Git), code documentation, and team development processes. You may better connect your abilities and methods with the demands of the market thanks to these encounters.

Overall, working on a URL shortener project while doing a Python internship is a great chance to put your coding skills to use in a practical, real-world setting. You can gain technical expertise, capacity for problem-solving, project management skills, and knowledge of industry standards. Always remember to cooperate with your peers, ask your mentors for advice, and get the most out of your internship's learning opportunities.

We appreciate the internship opportunity provided by Upskill Campus (USC) or The IoT Academy in association with UniConverge Technologies Pvt Ltd (UCT).

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.

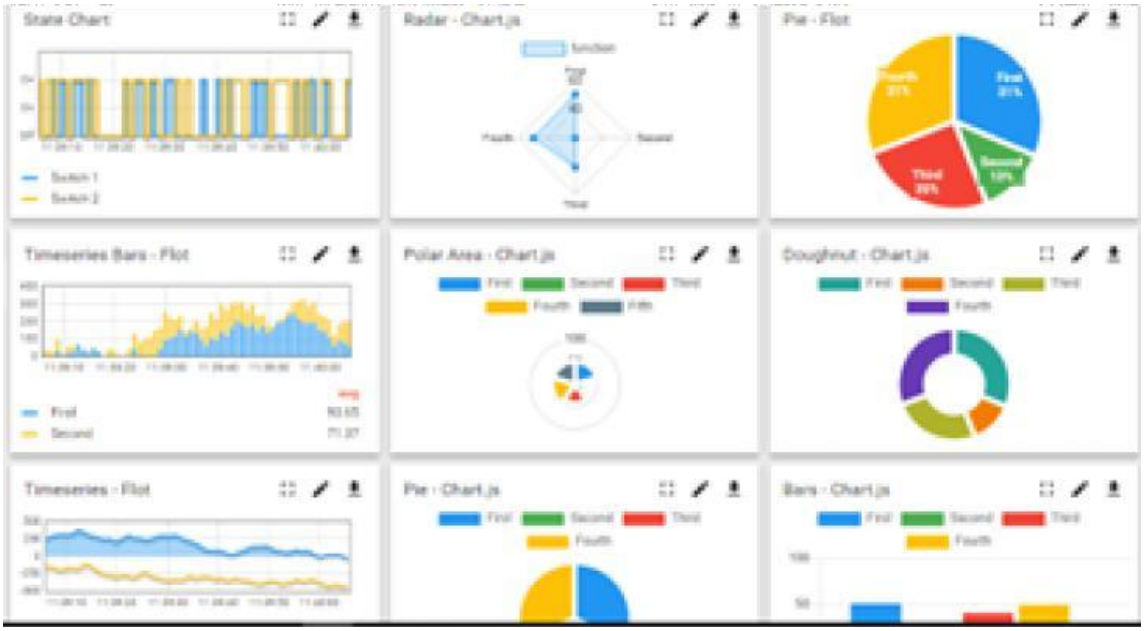
i. UCT IoT Platform (UCT Insight)

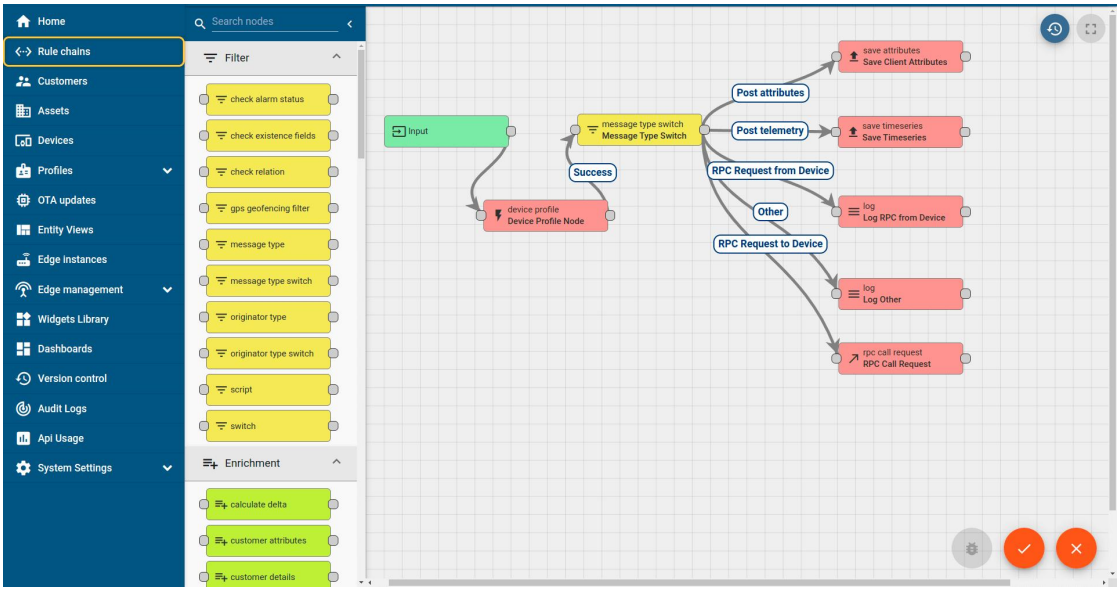
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

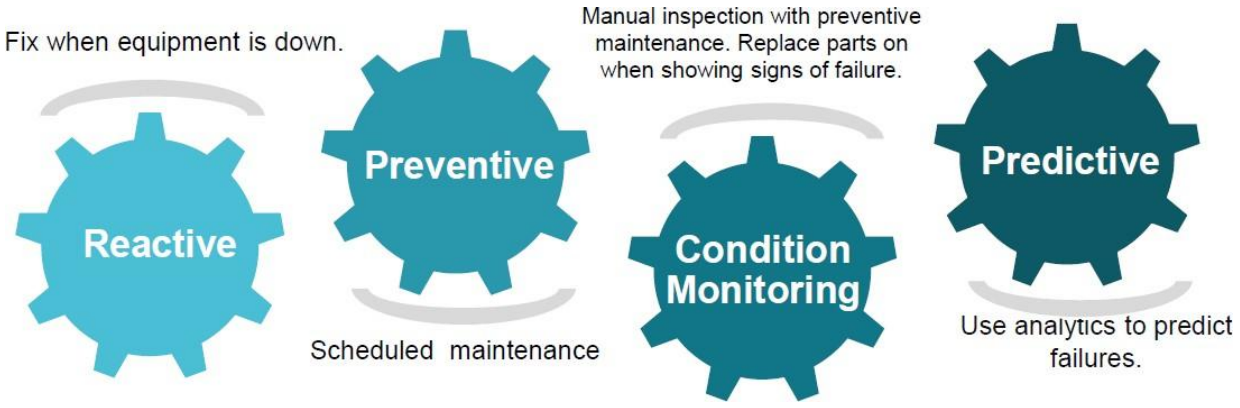
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

iii. LoRaWAN 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.

iv. Predictive Maintenance

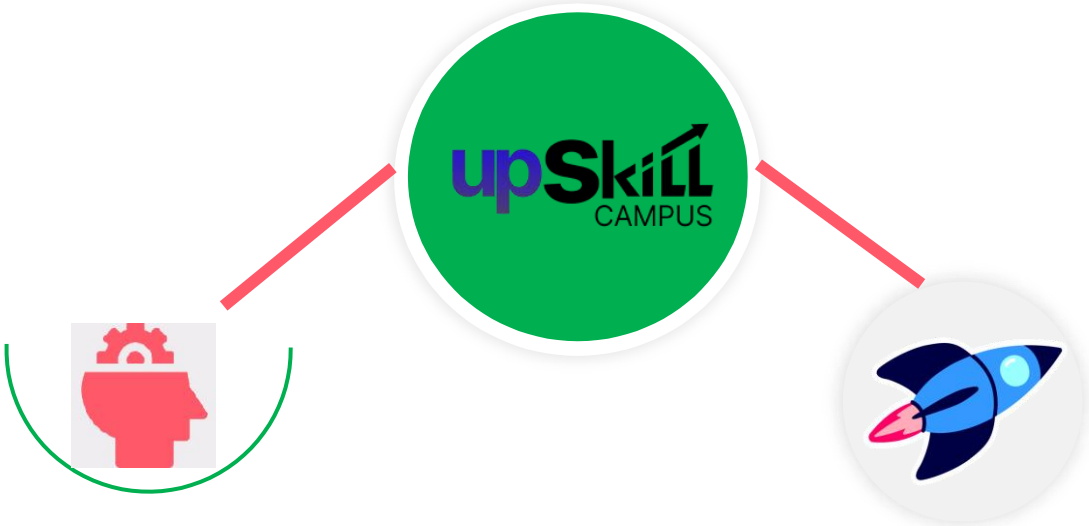
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.

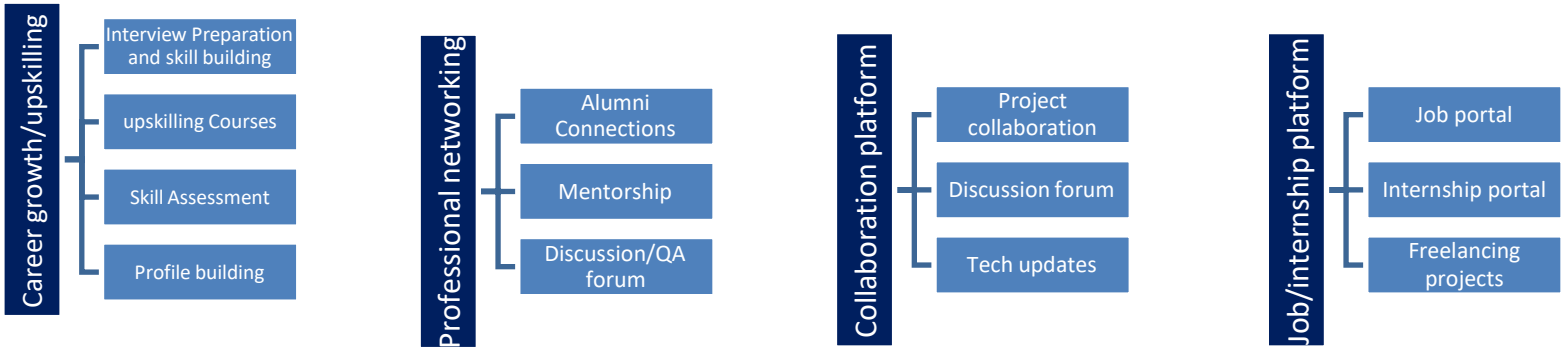
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



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] Bitly API: Bitly is a popular URL shortening service that provides an API for developers to integrate shortening and analytics features into their applications. You can find more information about their API documentation here:

https://dev.bitly.com/docs/getting-started/authentication/

[2] TinyURL: TinyURL is another well-known URL shortening service. Although they don't provide an official API, you can refer to their website and examine how they generate and redirect shortened URLs:

https://tinyurl.com/

[3] Open source URL Shortener Projects: If you're interested in exploring open-source implementations of URL shorteners, you can check out projects like YOURLS (https://yourls.org/) (https://polrproject.org/).

2.6 Glossary

Terms	Acronym
URL	Uniform Resource Locator - A web address that specifies the location of a resource on the internet.
Shortened URL	A compressed or abbreviated version of a URL that redirects to the original, longer URL when accessed.
Redirect	The act of automatically rerouting a user to another URL. When using URL shorteners, visiting a shortened URL causes a redirect to the lengthy original URL.
Backend	The server-side of a web application that processes requests, manages data, and handles the logic behind the scenes. In a URL shortener project, the backend code would handle generating shortened URLs, storing them, and managing the redirection process.
API	Application Programming Interface - A set of rules and protocols that allows different software applications to communicate with each other. APIs are often used in URL shorteners to handle URL shortening requests and redirection.

3 Problem Statement:

Design and develop a URL shortener application using Tkinter, a Python GUI library. The application should provide a user-friendly interface for users to input long URLs and obtain shortened versions. The shortened URLs should redirect users to the original long URLs when clicked.

Certainly! The problem statement outlines the objectives and requirements for developing a URL shortener application using Tkinter. Let's break down the key components:

1. User Interface: 1. Tkinter should be used to build an aesthetically pleasing and user-friendly interface for the programme. Users should be able to enter lengthy URLs in the input fields, and the appropriate abbreviated URLs should be displayed in the display fields.

2. URL Shortening: Implement a formula that, given a lengthy URL, produces a special abbreviated version. The algorithm should make the abbreviated URLs simple to share and remember.

3. Database Integration: Include a database to hold both the full URLs and their abbreviated variations. The programme should save the mapping in the database when a user provides a lengthy URL for future usage.

4. Redirection: Create a system that links the shorter URLs to their longer counterparts. The programme should obtain the appropriate long URL from the database when a user clicks on a shortened URL, then lead them to the original URL.

5. Copy to Clipboard: Offer a tool that enables visitors to quickly copy the abbreviated URL to the clipboard. Sharing the shortened URL in other programmes or platforms is made easier as a result.

6. Error Handling: In order to solve potential problems like incorrect URLs or database connection difficulties, provide strong error handling procedures. When required, alert users with suitable error messages.

7. User Experience: Ensure that the experience is seamless and user-friendly. To improve usability and make the programme easy for users, take into account such elements like progress indicators, clear status messages, and a responsive design.

8. Testing and Documentation: Test the programme in its entirety to make sure it is reliable and accurate. Create documentation for the project, covering design choices, implementation specifics, and application usage guidelines.

You will build a solution that makes sharing lengthy URLs easier and increases accessibility if you are successful in designing this URL shortener application. An appealing user interface, efficient URL shortening, database storage, redirection, simple clipboard copying, gentle error handling, and a good user experience are all features of the programme.

4 Existing and Proposed solution

Existing URL shortening solutions provided by various organizations and services have gained popularity and widespread usage. Here is a summary of some common solutions and their limitations:

1. bit.ly: A well-known URL shortening service called Bit.ly provides a user-friendly interface, statistics, and choices for unique short domains. The drawbacks include reliance on the service provider, potential dependability difficulties in the event that the service is unavailable, and restricted control over the shortened URLs in the event that the user wishes to transfer to another service.

2. TinyURL: Another well-liked URL shortening tool that enables users to easily create short connections is TinyURL. The service's lack of sophisticated capabilities like custom domains or in-depth analytics is its biggest drawback, though. The service might not be as available or scalable as those from bigger companies.

Limitations of existing URL shortening solutions in general include:

- Dependency: When a user relies on a third-party service for URL shortening, the service's dependability and availability are out of their control. The operation of shortened URLs may be interfered with if the service goes offline or is discontinued.

- Privacy and Security: Some URL shortening services may monitor user information, such as IP addresses and surfing patterns. This poses privacy issues, especially if the abbreviated URLs are distributed in delicate situations. Furthermore, shortened URLs could be changed to go to fraudulent or hazardous websites.

- Link Longevity: Depending on the service, abbreviated URLs might last for a short while or forever. Shortened URLs may only be accessible for a certain amount of time on some services. Links that no longer function might result in faulty redirects if the service or shortened URL expires.

- Scalability: Scalability might be difficult when URL shortening services grow in popularity. The functionality and responsiveness of the service may be impacted by high traffic or excessive usage, which might delay or disrupt the redirection process.

- Customization: Although some URL shortening providers include possibilities for bespoke domains, branding and personalization may still be constrained. Users might not have full control over how the abbreviated URLs look and are branded.

Proposed Solution:

Proposed solution for developing a URL shortener

1. Design the User Interface: Utilising widgets like labels, text fields, buttons, and message boxes, create a GUI using Tkinter. Include a text area to enter the lengthy URL, a button to shorten it, and a text field to show the shorter URL in the layout.

2. URL Shortening Logic: Python's reasoning for URL shortening should be used. To create a distinctive abbreviated URL for the provided lengthy URL, you can use a hashing algorithm or any other technique. For creating the abbreviated URLs, take into account utilising libraries like "hashlib" or "shortuuid."

3. Handle Button Clicks: Create procedures to deal with button clicks. Retrieve the input from the lengthy URL field and give it to the URL shortening mechanism when the "Shorten" button is clicked. Show the abbreviated URL in the text box.

4. Copy to Clipboard: When a user presses a "Copy" button, provide the capability to copy the truncated URL to the clipboard. The 'pyperclip' package may be used to do this.

5. Error Handling: To handle potential mistakes like incorrect URLs or problems with the URL shortening process, provide error handling. Use status labels or message boxes to display the relevant error messages.

6. Testing and Validation: Check that the abbreviated URLs are created appropriately by inputting a variety of lengthy URLs and thoroughly testing the programme. By pressing the "Copy" button and making sure the abbreviated URL is correctly transferred to the clipboard, you can check the functioning.

Documentation: Explain the design decisions and implementation specifics in the code documentation. Give detailed instructions on how to launch the programme and install any necessary dependencies. Include an instruction manual on how to utilise the URL shortener as well.

To guarantee that the supplied URLs are correct, handle user input validation and keep security considerations in mind, such as avoiding malicious URLs. Adding features like a history log, custom alias choices, or analytics are also potential ways to improve the user experience.

By utilising this suggested approach, you may create a useful URL shortener for Tkinter that enables users to compress lengthy URLs and copy the compressed versions to the clipboard.

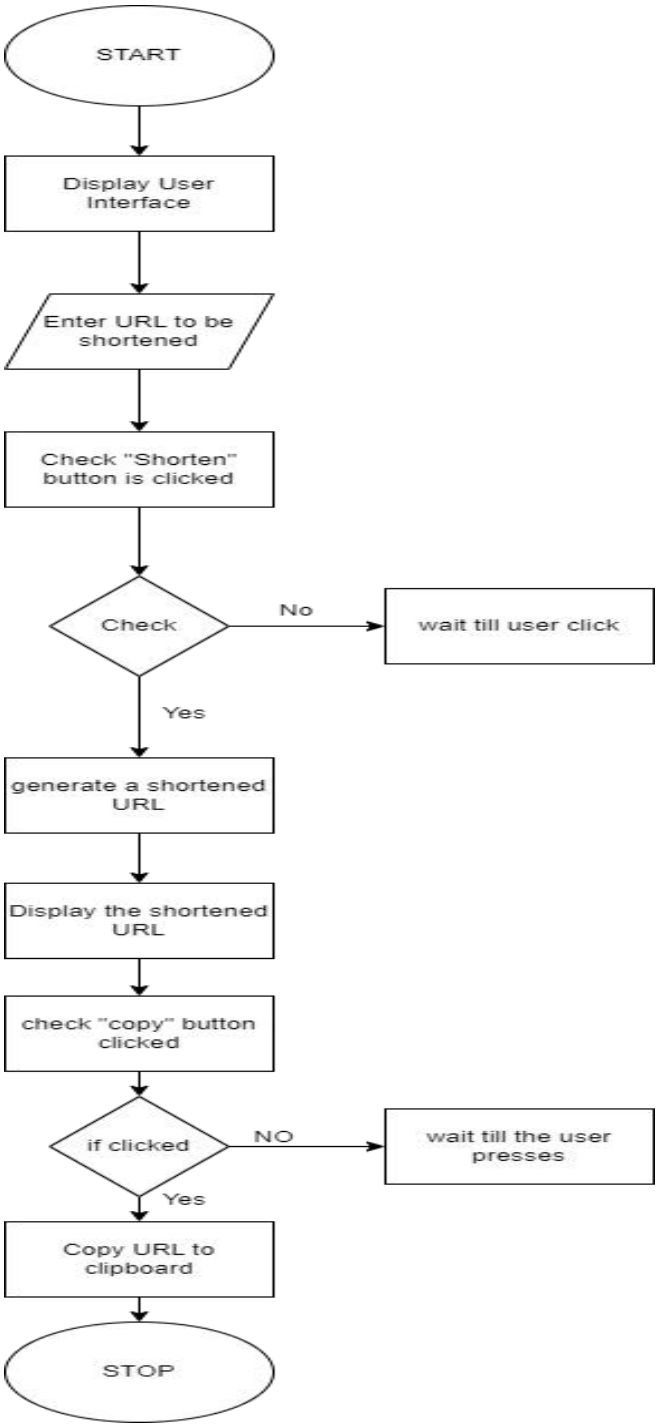
4.1 Code submission (Github link)

https://github.com/hemang26/URL-Shortener/blob/main/URLshortener.py

4.2 Report submission (Github link): first make placeholder, copy the link.

https://github.com/RajaJangam/urlshortener/blob/main/urlshortener.py

5 Proposed Design/ Model



5.1 High Level Diagram

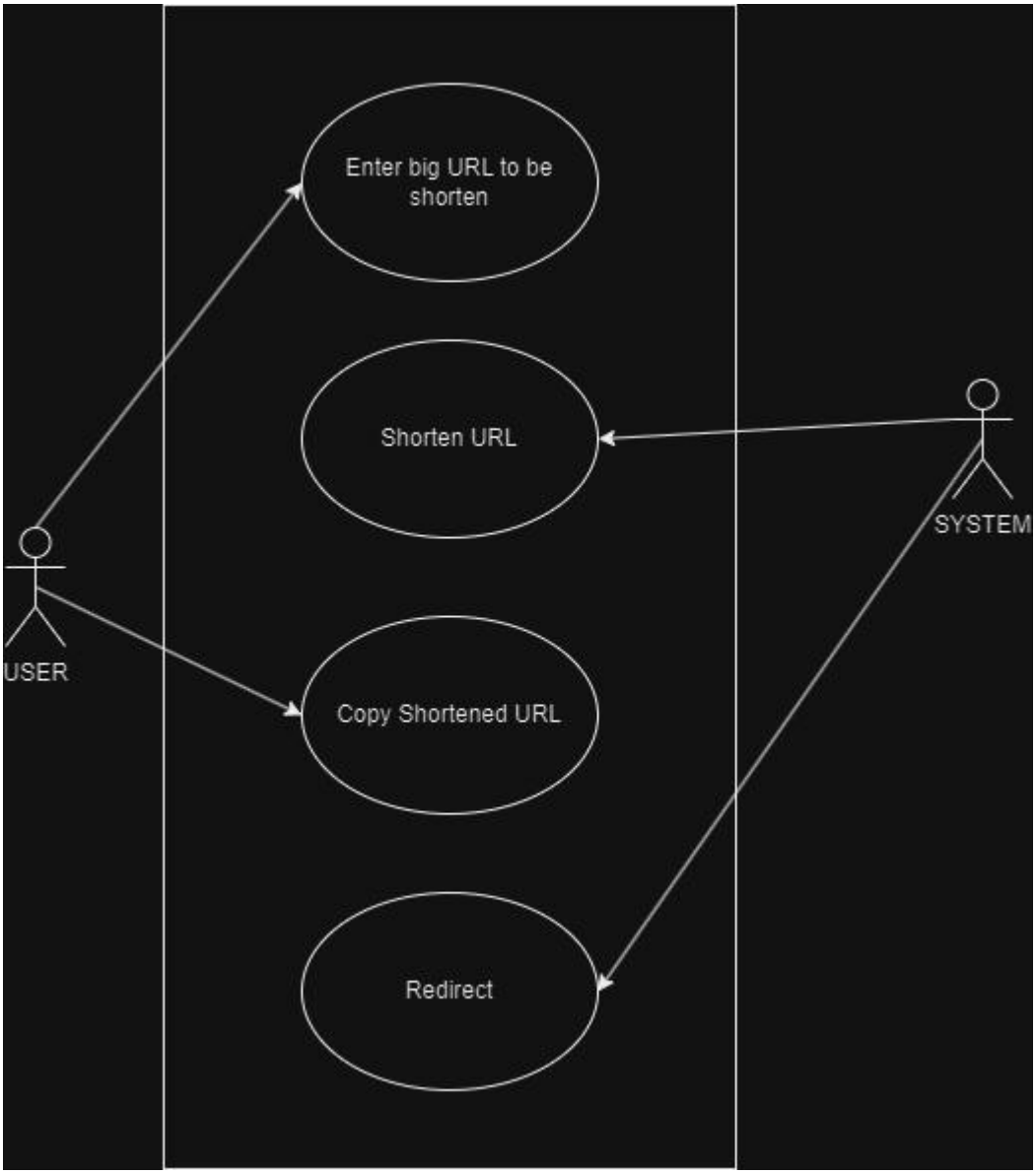


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

6 Performance Test

This is very important part and defines why this work is meant of Real industries, instead of being just academic project.

The constraints of the provided URL shortener example using Tkinter are as follows:

1. Bitly Access Token: The code is predicated on your possession of a current Bitly access token, which is required to use the Bitly API. You must replace "YOUR_BITLY_ACCESS_TOKEN" with the actual access token you received from Bitly. The URL shortening feature won't operate without a valid access token.

2. Internet Connection: To send queries to the Bitly API and get the abbreviated URL, the code has to be connected to the internet. The code won't be able to interface with the API without an internet connection, which might lead to an error.

3. GUI Interaction: The code is made to work with Tkinter's graphical user interface (GUI). The "Shorten" button is assumed to be clicked when the user enters the URL to be abbreviated in the text entry box. You will need to adapt the code if you wish to use it in a non-GUI setting or incorporate it into another programme.

4. Bitly API Usage Limits: The Bitly API may have certain usage constraints, such as rate caps or daily limits on the amount of API requests permitted. To comprehend any restrictions or limits put out by Bitly, you should refer to the documentation for the Bitly API.

These are the main constraints specific to the provided code example. If you have any additional requirements or constraints, please specify them, and I'll do my best to address them.

6.1 Test Plan/ Test Cases

Here's a test plan that covers various scenarios for testing the URL shortener using Tkinter:

1. Test Case: Successful URL Shortening

- Description: Enter a valid URL and click the "Shorten" button.
- Expected Result: The code should send a request to the Bitly API, receive a successful response (status code 200), and display the shortened URL in the output label.

2. Test Case: Invalid URL

- Description: Enter an invalid URL (e.g., "INVALIDURL") and click the "Shorten" button.
- Expected Result: The code should display an error message indicating that an error occurred while shortening the URL.

3. Test Case: No Internet Connection

- Description: Disable the internet connection and attempt to shorten a URL.
- Expected Result: The code should display an error message indicating a failure to connect to the Bitly API.

4. Test Case: Empty URL

- Description: Leave the URL entry field empty and click the "Shorten" button.
- Expected Result: The code should display an error message indicating that an error occurred while shortening the URL.

5. Test Case: Incorrect Bitly Access Token

- Description: Replace the access token with an incorrect value and attempt to shorten a URL.
- Expected Result: The code should display an error message indicating that an error occurred while shortening the URL.

6. Test Case: Valid Bitly Access Token

- Description: Replace the access token with a valid Bitly access token and shorten a URL.
- Expected Result: The code should successfully shorten the URL and display the shortened URL in the output label.

7. Test Case: Multiple URL Shortenings

- Description: Perform multiple URL shortenings consecutively without encountering any errors.
- Expected Result: The code should handle multiple URL shortenings without any issues, displaying the correct shortened URLs for each input.

8. Test Case: API Rate Limiting

- Description: Perform URL shortenings at a rate that exceeds the Bitly API's rate limit.

- Expected Result: The code should handle the rate limit gracefully, displaying an error message indicating that the rate limit has been exceeded.

It is also advisable to perform additional tests, including edge cases, to ensure the robustness and reliability of the code.

6.2 Test Procedure

To execute the test plan for the URL shortener using Tkinter, you can follow this test procedure:

1. Prepare the Test Environment:

- Ensure that you have a working internet connection.

- Install the necessary dependencies, including the `requests` library (use `pip install requests` if not already installed).

- Obtain a valid Bitly access token from the Bitly API dashboard.

2. Set Up the Test Environment:

- Open the code in a Python development environment or text editor.

- Replace ``'YOUR_BITLY_ACCESS_TOKEN'`` with your actual Bitly access token.

3. Execute the Test Cases:

- Run the code to launch the URL shortener GUI.

- Enter the test case inputs in the GUI according to each test case description.

- Observe the output displayed in the GUI.

4. Validate the Test Results:

- Compare the observed output with the expected results for each test case.

- If the observed output matches the expected result, mark the test case as "Passed."

- If the observed output differs from the expected result, mark the test case as "Failed" and investigate the issue.

5. Report and Track Issues:

- Create a test report documenting the test case results, including any failed test cases and associated issues.

- Clearly describe any errors or unexpected behaviors encountered during testing.

- Track and prioritize the identified issues for further investigation or resolution.

6. Retest and Verify Fixes:

- If any issues were discovered, work on resolving them.

- Retest the affected test cases after applying the fixes.

- Verify that the fixed issues no longer occur and update the test report accordingly.

7. Repeat Steps 3-6:

- Continue executing the remaining test cases from the test plan.

- Follow the same process of validating results, reporting issues, and retesting after fixes.

8. Complete the Testing Process:

- Once all test cases have been executed, reviewed, and any necessary fixes have been applied, finalize the test report.

- Summarize the overall test results, including the number of passed and failed test cases.

- Provide any additional observations, recommendations, or notes related to the testing process.

By following this test procedure, you can systematically execute the test plan and ensure that the URL shortener using Tkinter functions correctly under different scenarios.

6.3 Performance Outcome

Performance outcome of the URL shortener using Tkinter.

1. Network Latency: The network latency and response time of the Bitly API have a significant impact on how well the URL shortener performs. The speed of the application as a whole may be impacted if the API calls take a long time to complete because of network or server-side processing. However, because the API request is so low, most of the time the effect is anticipated to be negligible.

2. GUI Responsiveness: The performance of the URL shortener is significantly impacted by the Bitly API's network latency and response time. If the API requests take a long time to complete due to network or server-side processing, this may slow down the programme as a whole. The effect is usually expected to be insignificant due to the minimal API request, though.

3. Concurrency and Scalability: The code example runs on a single thread and processes a single request for URL shortening at a time. As a result, it could be unable to effectively manage several requests in simultaneously.

Consider evaluating the execution time for URL shortening requests and keeping an eye on the application's resource utilisation with the right tools to assess the performance of the URL shortener in your particular environment. Depending on your unique requirements and workload, this will assist you in finding any possible bottlenecks or opportunities for optimisation.

Code:

```
from copy import copy
from struct import pack
from tokenize import String
import pyperclip
import pyshorteners
from tkinter import *
#madebyHemangVyas
#GUI
root = Tk()
root.geometry("700x350")
root.title("URL Shortner Application")
root.configure(bg="#7F7FFF")

#Define Variable for url
urlmain = StringVar()
urlshortmain = StringVar()

#Define Function
def urlShortner():
    urladdress = urlmain.get()
    urlshort = pyshorteners.Shortener().tinyurl.short(urladdress)
    urlshortmain.set(urlshort)

def copyurl():
    urlshort = urlshortmain.get()
    pyperclip.copy(urlshort)


Label(root,text="URL Shortner App",font=('Arial', 12)).pack(pady=10)
Entry(root,textvariable=urlmain,width=50,font=('Arial 14')).pack(padx=10, pady=10)
Button(root,text="Generate Short Url",command=urlShortner).pack(pady=7)
Entry(root,textvariable=urlshortmain,width=50,font=('Arial 14')).pack(padx=10, pady=10)
Button(root,text="Copy Short Url",command=copyurl).pack(pady=5)

root.mainloop()
#author: hemang vyas
```

Output:


My URL Shortener App
Enter The Main URL
techgator.in
Generate Short URL
The Short URL
https://tinyurl.com/yyohusj3
Copy the Short URL

7 My learnings

Continuous learning is essential for professional development and career advancement. It improves people's knowledge, abilities, and expertise, increasing their adaptability and competitiveness in the work market. Learning encourages creativity and adaptation, allowing people to successfully traverse changing work situations. Additionally, it offers chances for networking and teamwork, enabling people to form useful relationships and get mentoring. People may prepare the route for job progress and take advantage of new chances by investing in their education and remaining current with the most recent developments in their area.Participating in an internship opportunity, whether through Upskill Campus (USC) or The IoT Academy in partnership with UniConverge Technologies Pvt Ltd (UCT), may offer practical experience and improve professional opportunities.

A special opportunity provided by Upskill Campus (USC) or The IoT Academy in association with UniConverge Technologies Pvt Ltd (UCT) is the Python URL Shortener Internship. The goal of this internship is to provide interns practical experience using Python to create a URL shortener application.

Interns will get the ability to work on a genuine project throughout the internship and pick up useful skills in Python programming, web development, and API integration. They will concentrate exclusively on adding a URL shortening feature utilising a chosen URL shortening service or API.

Aspects of the development process covered by the internship programme include understanding the requirements, developing the application architecture, coding the functionality, and testing the application for performance and dependability. Additionally, best practises for software development, code documentation, and version control will be taught to interns.

Additionally, interns will get the chance to work with seasoned mentors and industry experts who will coach and assist them during the internship.

Interns will learn useful skills in Python programming, web development, API integration, and software development processes by taking part in the Python URL Shortener Internship.

Additionally, they will improve their cooperation and communication skills, learn how to solve problems, and receive important industry experience.

In addition to giving interns a strong foundation in Python programming, completing the internship successfully will also be a great addition to their resumes. The skills and information they acquired during the internship will significantly advance their professional development and provide new chances in the software development industry.

8 Future work scope

Certainly! Here are some ideas that could be explored in the future for the Python URL Shortener project:

1. Custom URL Aliases: Introduce a functionality that lets users edit the abbreviated URL by entering a custom alias or phrase. This can entail checking that aliases are unique and resolving issues when numerous users ask for the same alias.

2. URL Analytics: Improve the programme so that it tracks usage of shortened URLs and provides analytics on it. This can involve recording information on clicks made, visitor locations, referrer details, and other pertinent analytics.

3. User Authentication and Management: Add user authentication to the app, enabling users to create individual accounts. This would make it possible to control personalised URLs, trace history, and perhaps even use collaborative tools.

4. Link Expiration and Revocation: Add the capability to give the abbreviated URLs an expiration date. This will enhance security and control over shared links by enabling users to immediately cancel access to a shortened URL after a certain time.

5. Integration with Multiple URL Shortening Services: Implement support for different URL shortening services rather than depending primarily on Bitly. Users would have more alternatives and freedom in selecting their chosen service as a result.

6. URL Preview and Safe Browsing: Include a URL preview function that, when a shortened URL is hovered over, displays a preview of the target website. Incorporate safe surfing APIs as well in order to confirm the security of the destination URL and alert users to any links that could be dangerous or hazardous.

7. API Rate Limiting and Throttling: To ensure compliance with the restrictions given by the API of the URL shortening provider, implement rate limiting and throttling measures within the application. This would avoid overuse of APIs and associated interruptions from going over use caps.

URL Validation and Sanitization: Incorporate extensive validation and sanitization procedures to improve the application's handling of URLs. This would make it easier to verify that the URLs entered are valid, safe, and devoid of any harmful information.

These are just a few suggestions for potential future Python URL Shortener project improvements. They can incorporate useful features and functionality to strengthen, improve, and secure the application.