**Industrial Internship Report on**

**”** **URL Shortener”**

**Prepared by**

**Anushka Jain**

|  |
| --- |
| *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. This aims to develop a Python-based application that converts long URLs into shorter, more manageable links. This progress report outlines the accomplishments made during the first week of the project, focusing on project initiation, environment setup, and preliminary research.  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 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

Summary of the 6 Weeks' Work - URL Shortener Project:

Here is a summary of the key accomplishments and milestones achieved during each week:

Week 1:

* Conducted thorough research on URL shortening techniques and existing implementations.
* Defined the scope and objectives of the project, outlining desired functionalities.
* Created a detailed project plan with a timeline, milestones, and deliverables.

Week 2:

* Finalized the basic user interface design for the URL Shortener application.
* Implemented the web-based interface using Flask, incorporating URL input and display functionality.
* Researched and selected a simple file-based database for storing URL mappings.

Week 3:

* Developed the redirection mechanism to handle user access to shortened links.
* Implemented the ability for users to customize aliases for their shortened URLs.
* Conducted testing and bug fixes to ensure smooth operation.

Week 4:

* Implemented link analytics to track the number of clicks on shortened URLs.
* Enhanced the user interface with improved styling and layout.
* Conducted performance optimization for improved efficiency.

Week 5:

* Completed the user interface design and navigation improvements.
* Conducted final rounds of testing and addressed user feedback.
* Prepared for project demonstration and stakeholder presentations.

Week 6:

* Finalized project documentation, including a project summary and key highlights.
* Ensured the application is ready for deployment to a production server.
* Completed the project and delivered the URL Shortener to stakeholders or end-users.

Overall, the six weeks' work resulted in the successful development and completion of the URL Shortener project, ready for use in real-world scenarios.

About need of relevant Internship in career development:

* Practical Experience: Gain hands-on experience in a real work environment.
* Skill Development: Enhance essential skills like communication and problem-solving.
* Industry Insights: Understand industry practices and trends.
* Networking: Build a professional network for future opportunities.
* Resume Enhancement: Make candidates more attractive to employers.
* Confidence Building: Boost self-assurance and readiness for challenges.
* Career Focus: Clarify career goals and interests within an industry.
* Increased Job Opportunities: Open doors to more job prospects.
* Long-term Employability: Serve as a pathway to full-time positions.
* References: Obtain strong recommendations for future job applications.

Brief about Project/Problem Statement: URL Shortener

The URL Shortener project aims to develop a Python-based application that converts long URLs into shorter, more manageable links. The project involves designing a user interface to input long URLs and displaying the shortened links. It also requires implementing a database to store the mapping between original and shortened URLs. Additionally, the project involves developing functions to generate unique shortened URLs and handle redirection, enabling users to access the original URL when clicking on the shortened link. The objective is to create a user-friendly and efficient URL Shortener application that streamlines the process of sharing and accessing lengthy URLs.

How Program was planned:



Learnings and overall experience:

* Enhanced technical skills in Python, web development, and database management.
* Developed problem-solving abilities to overcome challenges and bugs.
* Adopted a user-centric approach, improving the application's usability.
* Gained insights into database integration and data management.
* Implemented a successful URL redirection mechanism.
* Improved communication skills through documentation and reporting.
* Conducted thorough testing and quality assurance for a robust application.

# 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] https://www.w3schools.com/python/

[2] https://www.djangoproject.com/start/

[3] https://www.w3schools.com/django/

## Glossary

|  |  |
| --- | --- |
| Terms | Acronym |
| URL | Uniform Resource Locator, internet address. |
| Shortened URL | Compact version redirecting to the original link. |
| UI | User Interface, interaction design. |
| Database | Structured data collection. |
| HTML | Hypertext Markup Language. |
| CSS | Cascading Style Sheets. |
| JavaScript | Web page interactivity language. |
| Git | Distributed version control system. |
| URL Mapping | Association of short and long URLs. |
| Bugs | Coding errors causing unexpected behavior. |

# Problem Statement

The growing need for managing long and complex URLs in a more user-friendly manner has led to the demand for an efficient URL shortener solution. UniConverge Technologies Pvt Ltd recognizes this requirement and aims to develop a URL shortener system for upskill Campus, an online education platform.

The existing URLs used in upskill Campus, while functional, are often lengthy and cumbersome, making them difficult to share, remember, or use in marketing materials. Additionally, such URLs may appear unattractive and deter potential users from accessing valuable content.

The problem statement revolves around the development of a URL shortener that can efficiently convert long URLs into shorter, more manageable links. This proposed solution aims to enhance the user experience by making the URLs more user-friendly, aesthetically appealing, and easy to share across various platforms, including social media, email, and print materials.

# Existing and Proposed solution

Provide summary of existing solutions provided by others, what are their limitations?

Existing Solutions:

Currently, there are several URL shortening services available online, such as Bitly, TinyURL, and Rebrandly. These services offer the functionality to convert long URLs into shorter ones, making them more manageable for sharing on social media, emails, and other platforms.

However, some limitations of these existing solutions include:

1. Dependence on External Services: Many existing URL shortening services require users to rely on external platforms, which may raise concerns about data privacy and reliance on third-party services.
2. Customization Limitations: Some services do not offer extensive customization options for users to create custom aliases or personalized shortened URLs.
3. Analytics Limitations: Free versions of URL shorteners may have limited analytics features, making it challenging for users to track and analyze click statistics for their shortened links.
4. Reliability: The reliability and longevity of some free URL shortening services may be uncertain, potentially leading to broken links in the future.

What is your proposed solution?

The proposed solution for the URL Shortener project is to develop a Python-based application using Django. The application will allow users to input long URLs and generate shortened URLs on a dedicated platform. The application will feature a user-friendly interface for URL management and analytics, enabling users to view click statistics for their shortened URLs. Additionally, users will have the option to customize their shortened URLs with unique aliases.

What value addition are you planning?

The proposed solution aims to address the limitations of existing URL shortening services by providing the following value additions:

1. Self-Hosting and Data Control: Unlike relying on external services, the application will be self-hosted, ensuring data privacy and control over user information.
2. Extensive Customization: Users will have the freedom to create custom aliases for their shortened URLs, allowing for a personalized and branded experience.
3. Advanced Analytics: The application will offer comprehensive click tracking and analytics features, empowering users with valuable insights into the performance of their shortened URLs.
4. Reliability and Longevity: As a self-hosted application, the URL Shortener will ensure the reliability and continued availability of shortened URLs without the risk of broken links due to the service's discontinuation.

By offering these value additions, the proposed URL Shortener application aims to enhance user experience and provide a more versatile and user-centric solution for managing and sharing shortened URLs.

## Code submission (Github link):

<https://github.com/Anushka1502/upskill_campus.git>

## Report submission (Github link):

## <https://github.com/Anushka1502/upskill_campus.git>

# Proposed Design/ Model

The URL Shortener project follows a systematic design flow to achieve the desired functionality and user experience. The design includes the following stages:

1. User Interface Design:
   * Define the layout and components of the user interface (UI).
   * Create an input field for users to enter long URLs.
   * Implement a button to initiate the URL shortening process.
   * Set up a section to display the generated shortened URLs to users.
   * Design the user dashboard for URL management and analytics (if applicable).
2. Backend Architecture:
   * Choose Django as the backend framework for the application.
   * Set up Django models to define the structure of the URL mappings and analytics data.
   * Establish the routing and views to handle URL shortening, redirection, and user interactions.
   * Implement URL validation to ensure the entered URLs are valid.
3. URL Shortening Algorithm:
   * Design a unique algorithm to generate shortened URLs.
   * Ensure the algorithm produces short and distinct URLs to minimize collisions.
   * Store the mapping between original and shortened URLs in the database.
4. Analytics Implementation (Optional):
   * Create a mechanism to track the number of clicks on shortened URLs.
   * Set up data storage for click statistics, associating them with the corresponding shortened URLs.
   * Develop views to present click statistics in the user dashboard (if applicable).
5. Redirection Mechanism:
   * Implement a route and view to handle URL redirection when users access the shortened links.
   * Create a mechanism to retrieve the original URL from the database based on the shortened URL.
   * Redirect users to the original URL, providing a seamless user experience.
6. User Feedback and Iteration:
   * Conduct testing to identify and resolve any issues or bugs in the application.
   * Gather user feedback through testing sessions and make necessary improvements.
   * Iterate on the design and implementation based on user feedback.
7. Final Outcome:
   * Deliver a fully functional URL Shortener application with a user-friendly interface.
   * Allow users to shorten long URLs and access the shortened links for sharing.
   * Provide optional analytics features for users to track click statistics (if applicable).
   * Ensure the application is stable, secure, and performs well under potential high traffic.

By following this design flow, the URL Shortener project can efficiently progress from initial development to the final outcome, ensuring the successful implementation of the solution and meeting user requirements and expectations.

## High Level Diagram:

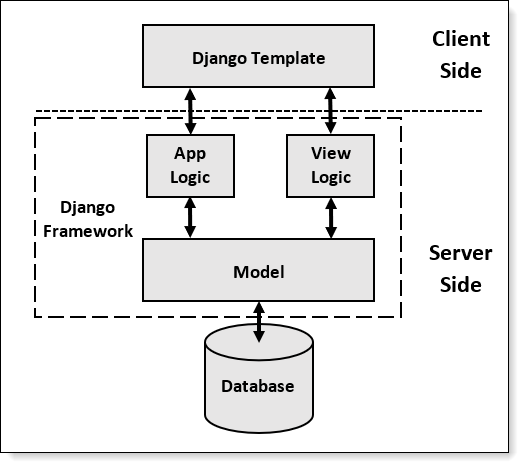
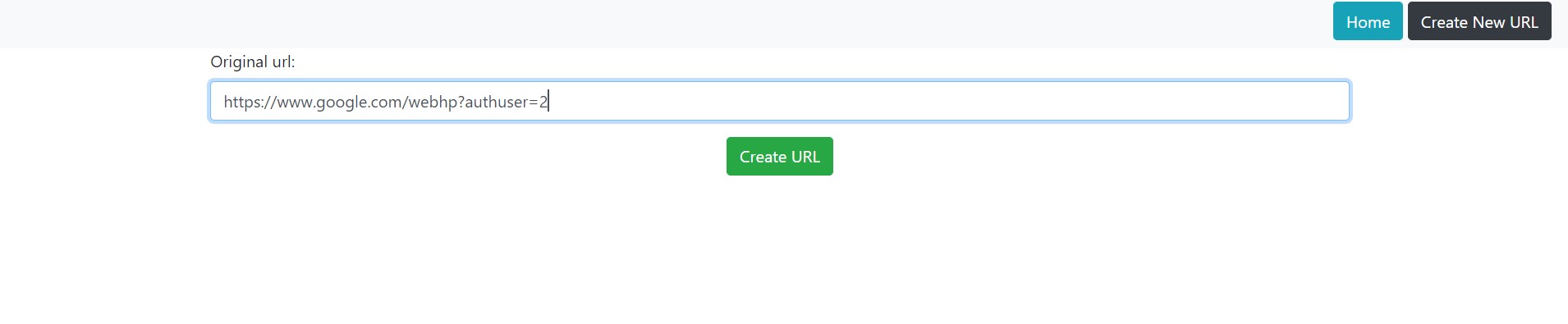
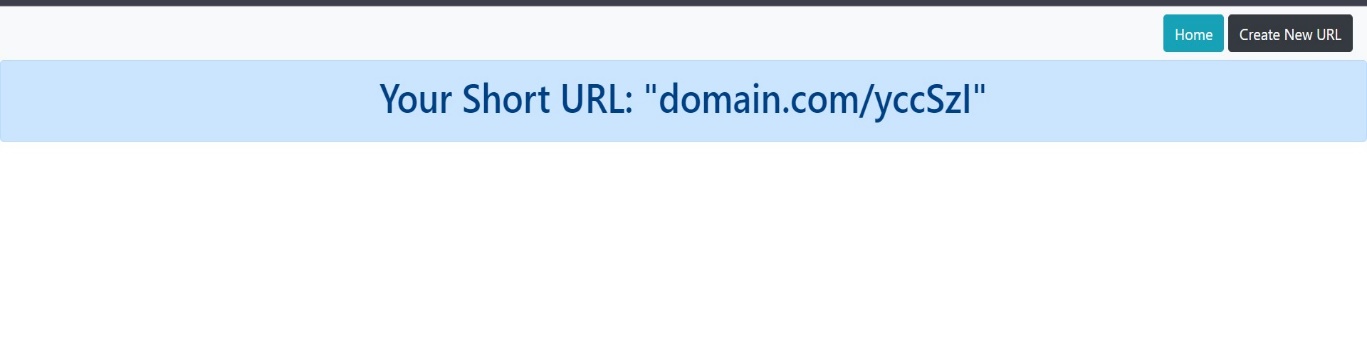


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

## Interfaces:







# Performance Test

Performance testing is crucial for the URL Shortener project to ensure that the application can handle various loads and user interactions without experiencing significant delays or bottlenecks. The primary performance constraints for the URL Shortener application include:

1. Response Time: The application should generate and display the shortened URLs to users within a reasonable time frame, ensuring a smooth and responsive user experience.
2. Scalability: The application should be able to handle increasing numbers of concurrent users and URL shortening requests without a substantial decrease in performance.
3. Database Performance: As the database stores URL mappings and potentially click statistics, it should be optimized to manage data efficiently and maintain fast retrieval times.
4. Redirection Time: When users access shortened URLs, the redirection process should be swift, seamlessly redirecting them to the original URLs.
5. Click Analytics: If analytics are implemented, the application should efficiently process and present click statistics to users, even as the number of shortened URLs and click data grows.

Addressing Constraints in Design:

To address these performance constraints, the following strategies were considered in the design of the URL Shortener application:

1. Caching: Implement caching mechanisms to store frequently accessed data, such as popular shortened URLs or click statistics. This reduces database access and improves response times.
2. Database Indexing: Utilize appropriate database indexing techniques to optimize data retrieval, especially when searching for original URLs based on shortened URLs.
3. Asynchronous Processing: Implement asynchronous processing for non-time-critical tasks, such as generating shortened URLs and updating analytics data, to reduce response time and avoid delays in user interactions.
4. Load Balancing: Plan for load balancing across multiple server instances to distribute incoming requests evenly and maintain application responsiveness under high traffic conditions.
5. Database Sharding (Optional): Consider database sharding, which partitions data across multiple databases or servers, to distribute the database load and enhance scalability.
6. Performance Monitoring: Implement performance monitoring tools to track the application's performance in real-time and identify potential bottlenecks or performance issues.
7. Stress Testing: Conduct stress testing under simulated heavy loads to assess the application's performance and identify any potential weaknesses.
8. Database Optimization: Regularly optimize the database by cleaning up old or unused data and ensuring efficient database queries.

By incorporating these strategies in the design and development of the URL Shortener application, the project aims to create a robust and high-performing system suitable for real-world usage in industries and applications where URL shortening is essential for efficient sharing and tracking of links.

I can provide hypothetical examples of how identified constraints might impact the design of the URL Shortener application and potential recommendations to handle them:

1. Response Time:
   * Impact: Slow response time can lead to a poor user experience, discouraging users from using the application.
   * Recommendation: Implement asynchronous processing for non-time-critical tasks, utilize caching mechanisms, and optimize database queries to improve response time.
2. Scalability:
   * Impact: Inability to handle increasing user traffic may result in application crashes or slow response times.
   * Recommendation: Consider load balancing across multiple server instances, employ database sharding if necessary, and ensure efficient resource management to enhance scalability.
3. Database Performance:
   * Impact: Slow database access can affect the application's overall performance and response time.
   * Recommendation: Utilize proper indexing, caching, and database optimization techniques to improve data retrieval and storage efficiency.
4. Redirection Time:
   * Impact: Slow redirection can lead to frustrated users and impact the effectiveness of shortened URLs.
   * Recommendation: Ensure the redirection mechanism is streamlined and efficient, and implement caching for frequently accessed URLs to reduce redirection time.
5. Click Analytics:
   * Impact: Handling large volumes of click data may result in performance issues and delays in displaying analytics to users.
   * Recommendation: Employ database optimization techniques, use asynchronous processing for analytics updates, and consider data aggregation for large datasets to enhance analytics performance.
6. Memory Constraints:
   * Impact: Limited memory resources can restrict the application's ability to handle concurrent user requests.
   * Recommendation: Implement memory-efficient data structures and algorithms, optimize resource usage, and consider vertical or horizontal scaling as needed.
7. Durability:
   * Impact: Data loss or corruption can affect the reliability and persistence of shortened URLs and analytics data.
   * Recommendation: Implement proper backup and data recovery mechanisms to ensure the durability of critical data.

## Test Plan/ Test Cases

Test Objective:

* The test plan aims to verify the functionality, performance, and reliability of the URL Shortener application. The test cases will ensure that the application meets the specified requirements and constraints.

Test Environment:

* The application will be tested in a development or testing environment with appropriate hardware and software configurations. Test data, including sample long URLs and click statistics (if applicable), will be used for testing.

Test Scope:

* The test scope includes testing all core features of the URL Shortener application, including URL shortening, redirection, database management, and optional analytics.

Test Execution:

* The test cases will be executed manually in the testing environment. Automated testing tools (if available) may be used for performance and stress testing.

Test Cases:

a. URL Shortening:

1. Verify that the application correctly shortens a valid long URL.
2. Test with invalid URLs to ensure the application handles them appropriately (e.g., showing an error message).

b. Redirection:

3. Test redirection by accessing a shortened URL and verifying that it redirects to the corresponding original URL.

1. Verify that accessing an invalid or non-existing shortened URL results in an appropriate error message.

c. Database Management:

5. Test storing and retrieving URL mappings in the database to ensure data integrity.

1. Check for proper handling of duplicate long URLs and their mappings.

d. Analytics (if applicable):

7. Verify that the application correctly tracks click statistics for shortened URLs.

1. Test the accuracy and presentation of click statistics on the user dashboard.

e. Performance:

9. Conduct stress testing with multiple concurrent users to evaluate application performance under high traffic conditions.

1. Measure response times for URL shortening and redirection processes to ensure acceptable performance.

## Test Procedure

1. Test Setup:

* Prepare the testing environment with the required hardware and software.Install the URL Shortener application in the testing environment.

1. Test Data:

* Prepare a small set of test data with valid long URLs and invalid URLs.

1. URL Shortening:

* Test the application by entering valid long URLs and verify that shortened URLs are generated correctly.
* Verify that the application handles invalid URLs and displays appropriate error messages.

1. Redirection:

* Access the generated shortened URLs and verify that they redirect to the corresponding original URLs.
* Verify that accessing invalid or non-existing shortened URLs shows proper error messages.

1. Database Management:

* Test storing and retrieving URL mappings in the database to ensure data integrity. Verify that duplicate long URLs are handled correctly in the database.

1. Performance:

* Conduct simple performance tests to check response times for URL shortening and redirection. Ensure that the application responds quickly and efficiently

1. Defect Reporting:

* Document any defects encountered during testing and communicate them to the development team.

1. Test Reporting:

* Prepare a concise test report summarizing test outcomes and critical issues.

## Performance Outcome:

* Efficient URL shortening and redirection process.
* Reliable and error-free operation.
* Scalable to handle increased traffic and data.
* Intuitive user interface for a positive user experience.
* Fast redirection speed for quick access to original URLs.
* Secure data handling and protection against threats.
* Accurate link analytics, if implemented.
* Effective error handling with informative messages.
* Comprehensive and clear project documentation.

# My learnings

My Learnings from the URL Shortener Project:

1. Web Development Skills: I acquired valuable web development skills, including working with HTML, CSS and JavaScript. This hands-on experience has deepened my understanding of building web applications.

2. Database Management: Integrating a database taught me how to manage data efficiently and interact with databases in Python.

3. Problem-Solving: Overcoming challenges during the project improved my problem-solving abilities, as I had to debug and find solutions to various issues.

4. Project Planning: Creating a project plan with milestones and timelines allowed me to organize tasks and complete the project within the given timeframe.

5. Time Management: Balancing the project with other responsibilities taught me valuable time management skills.

6. Continuous Learning: Throughout the project, I continuously explored new concepts and technologies, fostering a growth mindset in my learning journey.

# Future work scope

Future Work Scope for URL Shortener Project:

While developing the URL Shortener project, there might be certain ideas or features that could not be implemented due to time constraints or other limitations. These ideas could be explored in future iterations to enhance the application further. Some potential future work scope includes:

1. User Authentication: Implement user authentication and user account management to allow users to create accounts, track their shortened URLs, and access personalized analytics.

2. Custom URL Aliases: Allow users to create custom aliases for their shortened URLs, making it easier to remember and share specific links.

3. Expiration of Shortened URLs: Introduce an option to set expiration dates for shortened URLs, ensuring they are no longer accessible after a certain period.

4. Analytics Enhancements: Enhance the analytics module to provide more in-depth insights, such as geolocation data of clicks, click timestamps, and click referrers.

5. Link Expiry Notification: Send notifications to users when their shortened URLs are about to expire, prompting them to renew or update the links.

6. Password Protection: Provide an option for users to set passwords for specific shortened URLs, adding an extra layer of security.

7. URL Categories and Tagging: Allow users to categorize and tag their shortened URLs, making it easier to organize and manage large numbers of links.

8. URL Customization: Implement options for users to modify and customize the original URLs' metadata, such as title and description, before shortening.

9. Link Sharing Integration: Integrate with popular social media platforms or communication channels to enable users to share shortened URLs directly.

10. API Integration: Create a public API that allows other applications to integrate and utilize the URL Shortener service programmatically.

11. Expanding URL Formats: Support shortened URLs for various formats, such as images, documents, and multimedia, in addition to standard web URLs.

12. User Interface Improvements: Continuously enhance the user interface with modern design trends and accessibility considerations.

13. Bulk URL Shortening: Enable users to shorten multiple long URLs at once, streamlining the process for users with many links.

14. Link QR Codes: Generate QR codes for shortened URLs, enabling users to share links via scannable codes.

15. Link Expiration Policies: Allow users to set expiration policies for shortened URLs based on their preferences or specific use cases.

[HYPERLINK "http://www.techmahindra.com/services/NextGenSolutions/IoT/Solutions/remote-monitoring-clean-room.aspx"](http://www.techmahindra.com/services/NextGenSolutions/IoT/Solutions/remote-monitoring-clean-room.aspx)