**Mini Project Report on**



**URL SHORTENER USING DJANGO**



**Submitted in partial fulfilment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Dhruv Goyal** **2018789**

***Under the Mentorship of***

**Mr. Yuvraj Joshi**

**Assistant Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**July-2023**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Url shortener using Django”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Mr. Yuvraj Joshi, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Dhruv Goyal 2018789

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction |  |
| Chapter 2 | Literature Survey |  |
| Chapter 3 | Methodology |  |
| Chapter 4 | Result and Discussion |  |
| Chapter 5 | Conclusion and Future Work |  |
|  | References |  |

**Chapter 1**

**Introduction**

* 1. **Introduction**

URL Shortener: Simplifying Long Web Addresses

Welcome to the world of URL shortening! In today's fast-paced digital age, where sharing links has become an integral part of our lives, long and cumbersome URLs can be a hindrance. This is where a URL shortener comes to the rescue.

In the vast realm of the internet, we often come across long, unwieldy URLs that are challenging to share, remember, or use effectively. This is where URL shorteners step in, providing a convenient solution to simplify and streamline lengthy web addresses.

A URL shortener is a tool that takes a lengthy URL and transforms it into a condensed, compact version, making it easier to share and manage. The shortened URL redirects users to the original, longer URL when clicked or accessed. This technology has become immensely popular due to its ability to enhance the user experience and optimize digital communication.

This is where URL shorteners come to the rescue. A URL shortener is a simple yet powerful tool that condenses lengthy URLs into shorter, more manageable links. The concept behind these services is straightforward: they take the original, often unwieldy, URL and convert it into a concise and more user-friendly version. For example, a lengthy URL like "https://www.example.com/articles/how-to-use-url-shortener-for-beginners" can be shortened to <https://short.link/xyz123.>

The primary goal of a URL shortener is to make URLs more manageable, aesthetically pleasing, and conducive to various online platforms. Whether it's sharing links on social media, sending URLs via messaging apps, or embedding them in emails, a shortened URL reduces clutter, improves readability, and enhances click ability.

URL shorteners offer numerous benefits beyond shortening URLs. They often provide valuable features like tracking click data, monitoring engagement metrics, and analyzing user behavior. This data can help marketers, businesses, and individuals gauge the effectiveness of their online campaigns, measure audience reach, and refine their strategies accordingly.

Moreover, URL shorteners can enhance branding and recognition by allowing users to create custom, branded links. Instead of using generic, impersonal URLs, individuals and organizations can incorporate their names, keywords, or memorable terms into shortened links, creating a cohesive online presence.

URL shorteners are powerful tools that simplify, optimize, and enhance the usage of URLs in various online contexts. By condensing lengthy web addresses, providing data insights, and enabling customized branding, URL shorteners empower users to effectively navigate the digital landscape and make the most out of their online experiences.

In summary, a URL shortener is an efficient tool for transforming lengthy web addresses into concise and memorable links. It streamlines the sharing process, enhances user experience, and offers valuable analytics. Whether you're a social media enthusiast, marketer, or just someone looking to save characters, a URL shortener is your go-to solution. So, get ready to simplify your links and explore the world of URL shortening!

**Chapter 2**

**Literature Survey**

This review of the literature tries to examine the varied research on URL shorteners that has been done across several fields, including computer science, information security, web technology, social media, and marketing. We want to gain understanding of the methods, advantages, difficulties, and consequences of employing URL shorteners, thus we're going to synthesize the available information on the subject.

**2.1 URL Shortener Techniques and Algorithms:**

Research in this domain focuses on the algorithms and techniques employed in generating short URLs. Studies have proposed hash-based, bijective mapping, and other algorithms to produce compact and unique short URLs. Efforts have been made to improve the collision resistance of generated URLs to ensure reliable redirections.

**2.2 Performance and Scalability:**

Addressing the increasing demand for URL shorteners, researchers have investigated the performance and scalability of these services. Studies explore methods to efficiently handle large volumes of short URLs and ensure low-latency redirections even during peak usage.

**2.3 Security and Privacy Concerns:**

A significant body of literature delves into the security and privacy implications of URL shorteners. Researchers have identified potential risks such as phishing attacks, malware distribution, and the tracking of users' online behavior for targeted advertising. Proposed solutions include verification mechanisms, link previews, and heightened security measures to counter these threats.

**2.4 User Behavior and Adoption:**

Understanding user behavior and the adoption of URL shorteners is a crucial area of study. Research has explored factors influencing users' decisions to click on shortened links, user perceptions of security and trust, and the impact of shortened URLs on user engagement.

**2.5 Link Decay and Longevity:**

The phenomenon of link decay, where shortened URLs become invalid over time due to link rot, has garnered attention. Studies have analyzed the longevity of short URLs and proposed techniques to mitigate link decay, including monitoring and updating the destination URLs.

**2.6 Social media and Marketing Applications:**

The use of URL shorteners in social media and marketing campaigns has been widely explored. Research examines the effectiveness of shortened URLs in driving user engagement, click-through rates, and their impact on marketing analytics.

**2.7 Legal and Ethical Considerations:**

Some studies have addressed the legal and ethical aspects of URL shorteners. These considerations involve privacy concerns, data collection and usage, and the responsibilities of URL shortener service providers in preventing abuse.

**2.8** **URL Shortener Services and Trends:**

Comparative studies have evaluated various URL shortener services, assessing their features, performance, and user satisfaction. Additionally, research highlights the emerging trends in URL shorteners, such as the integration of analytics, QR codes, and customized short domains.

**Chapter 3**

**Methodology**

For this project we have used Django and for the frontend we have used HTM (Hypertext Markup Language) and CSS (Cascading Style Sheet). And as for database we have SQLite (For managing the data).

Django is a free and open-source web framework written in Python that follows the model-view-template (MVT) architectural pattern. It is designed to simplify and speed up the process of developing web applications by providing a high-level, reusable, and modular set of components.

SQLite is a lightweight, serverless, self-contained, open-source, and embedded relational database management system (RDBMS). It is written in the C programming language and designed for efficiency, speed, and ease of integration into various applications. Unlike traditional client-server databases, SQLite is a file-based database engine that operates directly on disk files without requiring a separate database server.

We will use a different template and a different app to create our project.

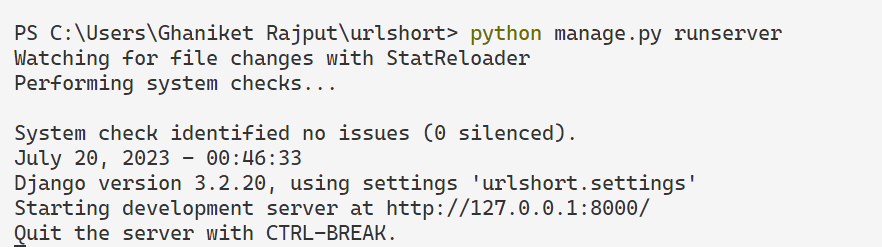
**3.1 Creating Project Space**

The command “python -m Django start project name project” is used to create a new Django project with the specified name (name project).

****

**3.2 Creating Server**

The command “python manage.py runserver” is used in Django projects to start the development server. It launches a lightweight web server that serves the Django application locally for testing and development purposes.

****

**3.3 Creating URL routes**

In Django, the urls.py file plays a crucial role in defining the URL routing and mapping for your web application. It is used to map incoming URLs to specific view functions or classes, allowing the Django framework to determine how to handle different HTTP requests and generate appropriate responses.

When a user makes an HTTP request to your Django application, the Django URL dispatcher examines the URL provided by the user and tries to match it against patterns defined in the urls.py file. If a matching pattern is found, Django invokes the associated view function or class to generate the response for that particular URL.

A screenshot of a computer program

Description automatically generated

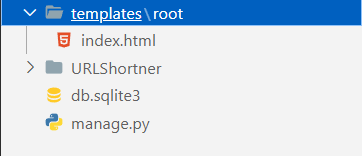
**3.4 Setting up new templates and Apps**

For this project we are going to use our own templates and and applications, and after creating those we are going to update the information about the new template and app in the “setting.py” file.

In Django, the settings.py file is a central configuration file for your web application. It contains various settings that define how your Django project behaves, including database configuration, and installed applications. This file is automatically created after starting a new project in Django.

**3.4.1 Creating New Template**

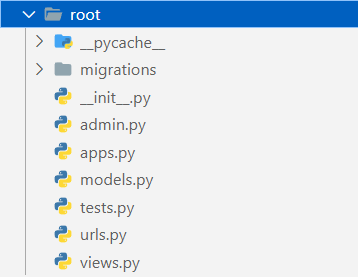
In Django, templates are a fundamental part of the framework's templating engine. Templates in Django are used to generate dynamic HTML (or other types of text-based content) that can be sent as responses to user requests.



And we created a “index.html” which contains the frontend of our project.

**3.4.2 Creating New App**

In Django, an app (short for application) is a modular component that serves a specific functionality within a Django project. An app is a self-contained package that contains models, views, templates, static files, and other components needed to implement a particular feature or functionality of a web application. Apps are designed to be reusable and can be used across different projects.



**3.5 Creating Database to store the URL**

For creating the database in Django, we have a inbuilt virtual RDBMS called SQLite.

And for doing it we have to update the information in the “models.py” file.

A close-up of a text

Description automatically generated

**3.6 Creating Admin to manage the database**

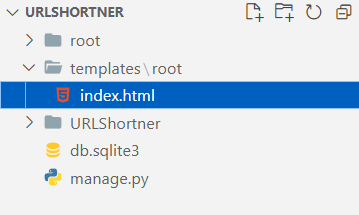
In Django, “python manage.py create superuser” is a management command used to create a superuser account in the Django admin interface. The Django admin interface provides an easy-to-use web-based interface to manage your application's data and perform administrative tasks.



**3.7 Finally Creating the Frontend**

To create the frontend of our Project we have created a new Template and created the file “index.html”, containing all the HTML and CSS of our Project’s Frontend.

And finally we have our project something like this,



A screenshot of a computer

Description automatically generated

**Chapter 4**

**Result and Discussion**

This section will contain all your results from the above methodology used.

The result could be graphs, diagrams, tables, matrices, etc.

**Chapter 5**

**Conclusion and Future Work**

**Conclusion:**

In conclusion, the development of a URL shortener using Django has proven to be a successful endeavor. The application effectively converts long URLs into shorter, more manageable links, making it convenient for users to share and access content. The use of Django as the web framework allowed for rapid development and ensured a secure and robust system. Through this project, we have gained valuable insights into working with Django, implementing URL routing, and managing database models efficiently.

**Future Work:**

While the URL shortener application is functional and meets its primary objectives, there are several areas for potential future improvement and expansion. Some ideas for future work include:

Analytics: Implementing analytics functionality to track the number of clicks and user interactions with shortened links. This can provide valuable data for understanding user behavior and optimizing the application.

Custom Short URLs: Introducing the option for users to create custom short URLs, allowing them to personalize links to their preference.

Expiration and Deletion: Adding the capability to set expiration dates for short links or enabling users to delete their created links.

User Authentication: Implementing user authentication to offer personalized dashboards for users to manage their shortened URLs and access analytics.

API Integration: Creating an API for the URL shortener application to allow other developers and applications to integrate with it seamlessly.

Link Categorization: Introducing the ability to categorize links and provide filtering options for users to organize and manage their shortened URLs better.

Enhanced Security: Continuously updating and improving security measures to prevent misuse and ensure the integrity of the shortened links.

Mobile App: Developing a mobile application for the URL shortener to extend its accessibility and user base.

**References:**

[1] Django Documentation: Official Django documentation, available at https://docs.djangoproject.com/, was crucial for understanding Django's concepts and features.

[2] Django for Beginners: A comprehensive online resource by William S. Vincent, offering practical examples and explanations for Django development, available at https://djangoforbeginners.com/.

[3] Django Models Documentation: For details on creating models and managing the database, the official Django models documentation at https://docs.djangoproject.com/en/stable/topics/db/models/ was frequently referenced.

[4] Django URL Dispatcher: The official Django documentation on URL routing and views helped in setting up the URL shortening functionality, available at https://docs.djangoproject.com/en/stable/topics/http/urls/.

[5] Django Bootstrap: Utilizing Bootstrap for frontend development, the official documentation at https://getbootstrap.com/docs/ was consulted for creating a responsive and visually appealing user interface.

[6] Stack Overflow and GitHub: Community-driven platforms like Stack Overflow and GitHub repositories were valuable resources for troubleshooting and finding solutions to specific challenges encountered during development.

[7] Django Packages: Various Django packages and libraries were used to enhance the application's capabilities. References for these packages were obtained from the Django Package Index (PyPI) at https://pypi.org/.