Task Documentation

Index

S.N.	Topic Name	Page No
1.	Objective	1
2.	Folder Structure	2
3.	API Documentation	2
4.	API Payloads & Responses	3
5.	Database Inforamtion	6
6.	Table Information	6
7.	Prerequisites	7
8.	Steps to Setup & Run the Project	8

Objective:

Task Title	Implement a URL Shortener with Expiry and Analytics
Objective	Build a Python-based URL shortener system that shortens URLs, tracks usage analytics, and allows for link expiration.
Requirements	• Core Functionality: • Expiry • Analytics • Storage • CLI or API: • Constraints
Bonus	• Use hashlib to create a hash-based short URL identifier. • Add optional password protection for accessing certain shortened URLs.
Task PDF	The task details and further instructions can be found in the attached PDF inside the **README** folder.

Folder Structure:

S.N.	File Name	Descriptions
1	main.py	Contains the Flask application and all the API route definitions.
2	api_methods.py	Handles the implementation of all API methods and business logic.
3	dbApi.py	Manages all database queries and interactions.
4	config.py	Stores configuration settings and constants (unchangeable data).
5	requirements.txt	Lists all required libraries and dependencies for the project to run smoothly.
6	py310env	The virtual environment where all packages and libraries are installed.
7	README	It contains documentation, execution flow, and instructions for running the project.

API Documentation:

S.N.	API Name	Method	Descriptions
	https://short.ly		Base URL
1	/shorten	POST	Create a shortened URL.
2	/<short_url>	GET	Redirect to the original URL with validations.
3	/analytics/<short_url>	GET	Retrieve analytics data for a specific shortened URL.

API Payloads & Responses:

API Name	/shorten
URL	http://127.0.0.1:5000/shorten
Methods	POST
Case 1	New URL

Payload	{
	  "URL": "https://github.com/SaurabhSingh86/Case-Study/blob/main/Python%20Project/Food_hub_orders_Project.html",
	  "expiry_hour": "",
	  "password": "123456"
	}

Response	{
	  "message": "URL shortened successfully",
	  "short_url": "https://short.ly/0a4c98",
	  "status": "success"
	}

Case 2	Duplicate URL

Response	{
	  "error": "Duplicate entry",
	  "message": "URL already exist",
	  "short_url": "https://short.ly/0a4c98"
	}

API Name	/<short_url>
URL	http://127.0.0.1:5000/e4b33c
Methods	GET
Payload	`{ "password": "123456" }`
Case 1	Expired URL
Response	`{ "error": "Expired URL", "field": "expiry_hour", "message": "Short URL has expired" }`
Case 2	Incorrect or no password provided (if a password was defined during the shorten API request).
Response	`{ "error": "Invalid Password", "field": "password", "message": "Incorrect Password" }`
Case 3	Redirect to the original URL (after successful validation) and return the response in HTML format
Response	`<!DOCTYPE html> <html lang="en" data-color-mode="auto" data-light-theme="light" data-dark-theme="dark" data-a11y-animated-images="system" data-a11y-link-underlines="true" >`

API Name	/analytics/<short_url>
URL	http://127.0.0.1:5000/analytics/e4b33c
Methods	GET
Case 1	Corrected Short URL
Payload	Not Required
Response	`{ "accessed_count": 2, "log": [ { "access_time": "Sun, 19 Jan 2025 12:30:48 GMT", "ip_address": "127.0.0.1" }, { "access_time": "Sun, 19 Jan 2025 13:05:04 GMT", "ip_address": "127.0.0.1" } ] }`
Case 2	Incorrect Short URL
Payload	Not Required
Response	`{ "accessed_count": 0, "log": [] }`

Database Information:

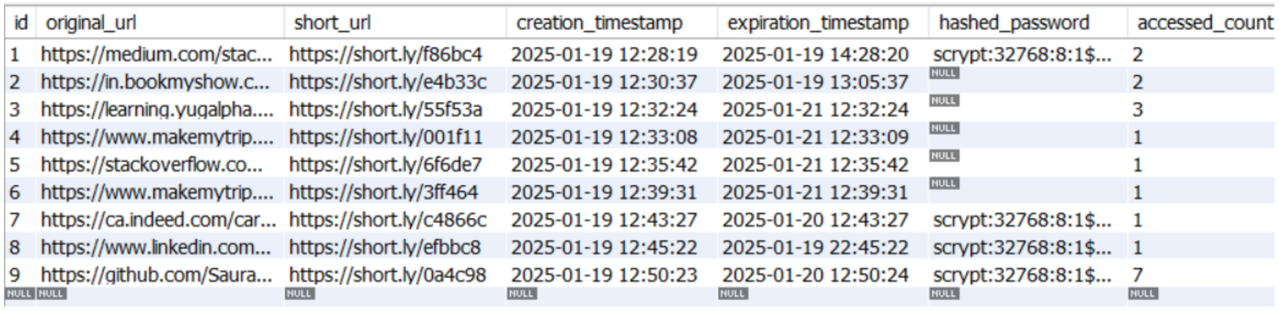
DB Info	Values
Host Name	localhost
User Name	root
Password	123456
Schema Name	url_shortener_db
Table Name	url_info_table access_logs_table
Remarks	I am currently using **MySQL(**Version 8.0.31**)** for the project.

Table Information:

Table Name	**url_info_table**
Purpose	This table stores the original URL, its shortened version, and other related metadata for the URL shortening system.

SQL Code	CREATE TABLE IF NOT EXISTS \`url_info_table\` (
	  \`id\` INT AUTO_INCREMENT PRIMARY KEY,
	  \`original_url\` TEXT NOT NULL,
	  \`short_url\` VARCHAR(30) NOT NULL UNIQUE,
	  \`creation_timestamp\` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
	  \`expiration_timestamp\` TIMESTAMP,
	  \`hashed_password\` VARCHAR(255) DEFAULT NULL,
	  \`accessed_count\` INT DEFAULT 0
	);

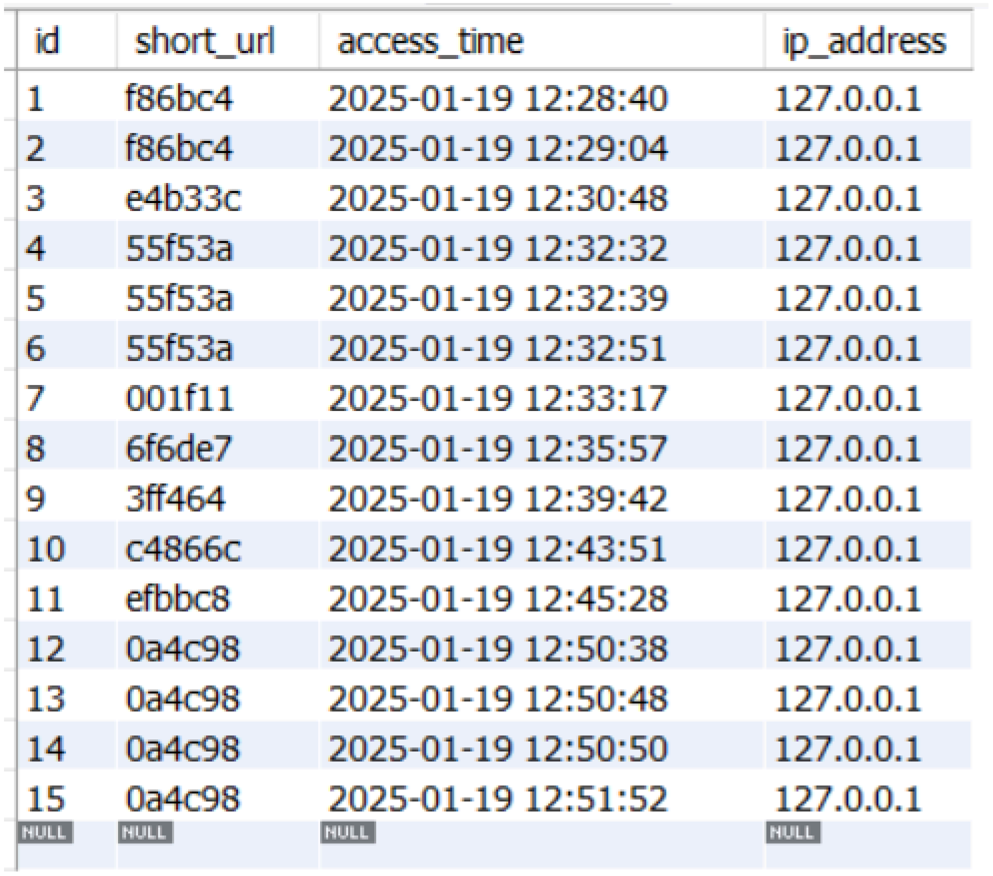
Attached is a screenshot of the table with test data, captured while performing and testing the APIs, for your reference.



id	original_url	short_url	creation_timestamp	expiration_timestamp	hashed_password	accessed_count
1	https://medium.com/stac...	https://short.ly/f86bc4	2025-01-19 12:28:19	2025-01-19 14:28:20	scrypt:32768:8:1$...	2
2	https://in.bookmyshow.c...	https://short.ly/e4b33c	2025-01-19 12:30:37	2025-01-19 13:05:37	NULL	2
3	https://learning.yugalpha....	https://short.ly/55f53a	2025-01-19 12:32:24	2025-01-21 12:32:24	NULL	3
4	https://www.makemytrip....	https://short.ly/001f11	2025-01-19 12:33:08	2025-01-21 12:33:09	NULL	1
5	https://stackoverflow.co...	https://short.ly/6f6de7	2025-01-19 12:35:42	2025-01-21 12:35:42	NULL	1
6	https://www.makemytrip....	https://short.ly/3ff464	2025-01-19 12:39:31	2025-01-21 12:39:31	NULL	1
7	https://ca.indeed.com/car...	https://short.ly/c4866c	2025-01-19 12:43:27	2025-01-20 12:43:27	scrypt:32768:8:1$...	1
8	https://www.linkedin.com...	https://short.ly/efbbc8	2025-01-19 12:45:22	2025-01-19 22:45:22	scrypt:32768:8:1$...	1
9	https://github.com/Saura...	https://short.ly/0a4c98	2025-01-19 12:50:23	2025-01-20 12:50:24	scrypt:32768:8:1$...	7
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table Name	**access_logs_table**
Purpose	This table is used to track access statistics and logs for shortened URLs.
SQL Code	```CREATE TABLE IF NOT EXISTS `access_logs_table` (`id` INT AUTO_INCREMENT PRIMARY KEY, `short_url` VARCHAR(30) NOT NULL, `access_time` TIMESTAMP DEFAULT CURRENT_TIMESTAMP, `ip_address` VARCHAR(45));```

Attached is a screenshot of the table with test data, captured while performing and testing the APIs, for your reference.



id	short_url	access_time	ip_address
1	f86bc4	2025-01-19 12:28:40	127.0.0.1
2	f86bc4	2025-01-19 12:29:04	127.0.0.1
3	e4b33c	2025-01-19 12:30:48	127.0.0.1
4	55f53a	2025-01-19 12:32:32	127.0.0.1
5	55f53a	2025-01-19 12:32:39	127.0.0.1
6	55f53a	2025-01-19 12:32:51	127.0.0.1
7	001f11	2025-01-19 12:33:17	127.0.0.1
8	6f6de7	2025-01-19 12:35:57	127.0.0.1
9	3ff464	2025-01-19 12:39:42	127.0.0.1
10	c4866c	2025-01-19 12:43:51	127.0.0.1
11	efbbc8	2025-01-19 12:45:28	127.0.0.1
12	0a4c98	2025-01-19 12:50:38	127.0.0.1
13	0a4c98	2025-01-19 12:50:48	127.0.0.1
14	0a4c98	2025-01-19 12:50:50	127.0.0.1
15	0a4c98	2025-01-19 12:51:52	127.0.0.1
NULL	NULL	NULL	NULL

Prerequisites:

Python	python 3.10.5
PIP	pip 24.3.1
MySQL	mysql 8.0.31

Steps to Set Up and Run the Project:

Steps	**Step Names**	**Descriptions**
Step 1	Create Virtual Environment (In this project envirome_name: **py310env**)	Create a virt_env to isolate project dependencies. **Commands:** python -m venv enviroment_name or pip install virtualenv virtualenv environment_name
Step 2	Activate environment	Activate the virtual environment created in Step 1. **Commands:** # Windows enviroment_name\Scripts\activate enviroment_name/Scripts/activate # Linux or Mac source enviroment_name\bin\activate
Step 3	Install all packages (Requirements.txt file)	Install the required packages listed in requirements.txt to the environment. **Command:** pip install -r requirements.txt
Step 4	Run main.py file	Run the main.py file to start the project **Command:** python main.py
Step 5	Hit shorten API	Refer to the API Payloads & Responses documentation to test the shortening API.
Step 6	Hit /<short_url> API	Refer to the API Payloads & Responses documentation to test the redirect functionality for a shortened URL.
Step 7	Hit /analytics/<short_url> API	Refer to the API Payloads & Responses documentation to test the analytics for the shortened URL.