**Abbottabad University of Science & Technology**

# REPORT

# For

# < URL Shortener>

**Developed By: WAQAS AHMED**

**Supervisor:(Sir Jamal Abdul Ahad)**

Table of Contents

## 1. Introduction

- **Purpose**: This section would explain why the URL shortener was developed, its goal to simplify long URLs for easy sharing, and potential use cases.
- **Document Conventions**: Defines any specific formatting or naming conventions used within the document.
- **Project Scope**: Describes the boundaries of the project—what the project includes and what is excluded.
    - **Scope Definition**: Details about the functional scope, e.g., shortening URLs.
    - **Core Features**: Key features of the URL shortener, such as URL input, shortening, copying to clipboard, sharing, and opening the shortened URL.
    - **Subsequent Releases**: Potential future releases with additional features.
    - **Alignment with User and Business Goals**: Ensures the system meets user needs, such as ease of URL sharing and managing URLs.
- **References**: Citations for any external resources or frameworks used.

## 2. Overall Description

- **Product Perspective**: This section provides a broader context of the application within the product ecosystem.
    - **Product Context**: How the shortener fits within a larger system (e.g., a website or service).
    - **Product Origin**: Information on how the product came into existence.
    - **Product Relationship to Existing Systems**: Explains how the URL shortener interacts with other systems or services.
    - **Product Ecosystem**: Describes how the system interacts with other products, tools, or services.
- **User Classes and Characteristics**: Defines the types of users who will interact with the application.
    - **Tech Enthusiasts**: Users who are familiar with the technology behind URL shortening.
    - **Casual Shoppers**: Users who use the shortener for personal use, like sharing URLs for purchases.
    - **Favored User Class**: The target user base that the project focuses on.
    - **Alignment with User Needs**: Describes how the project fulfills the needs of these users.
- **Operating Environment**: Describes the hardware, operating system, and network environment needed for the application.
    - **Hardware Platform**: The kind of servers or devices the application is expected to run on.
    - **Operating Systems and Versions**: Specific platforms the application supports.
- **Design and Implementation Constraints**: Constraints that may affect development.
    - **Database Technology**: The type of database or data storage mechanism.
    - **Third-Party Integrations**: Other services integrated into the project (e.g., sharing services).
    - **User Interface Design**: Guidelines for the design of the user interface.
- **Assumptions and Dependencies**: Any assumptions made during development (e.g., network availability, user device type) and dependencies on external libraries or services.
    - **Assumptions**: Assumptions like the user having an internet connection or using modern browsers.

    - **Dependencies**: Any external packages or APIs used, like `Flask`, `hashlib`, or the clipboard API.

## 3. System Features

- **Feature 1**: The ability to input a long URL.
- **Feature 2**: URL shortening using a hash generation function.
- **Feature 3**: Displaying the shortened URL for the user.
- **Feature 4**: Providing options to copy the shortened URL, share it, or open it in a new tab.

## 4. External Interface Requirements

- **User Interfaces**: Describes the design and layout of the user interface for entering URLs and interacting with the shortened URL.
    - **Design Standards and Guidelines**: Visual design conventions.
    - **Screen Layout and Resolution**: Responsiveness of the layout for different devices.
    - **Standard Interface Elements**: Buttons, inputs, and other interface components.
- **Software Interfaces**: Describes interactions between software components, such as API calls from the frontend to the backend.
    - **Non-Functional Requirements**: Includes performance, security, scalability.
- **Hardware Interfaces**: Describes the physical device and its interaction with the software (e.g., supporting mobile or desktop devices).
    - **Supported Device Types**: Whether the application is optimized for mobile, desktop, or both.

## 5. Quality Attributes

- **Performance**: Ensures the system responds quickly, even when handling multiple URL shortening requests.
- **Reliability**: Ensures the system works without failure (no downtime, etc.).
- **Usability**: Makes sure the system is user-friendly and easy to navigate.
- **Security**: Ensures shortened URLs can't be misused or compromised.
- **Maintainability**: The system can be easily updated or expanded in the future.

## Technical Explanation of the Code:

The project is built with **Flask**, which is used to create a web application.

1. **Frontend (HTML/CSS)**:
    - The page consists of an input field where users can paste a long URL, and a button to trigger the shortening process.
    - The result is displayed in a separate input field showing the shortened URL.
    - Users have additional options to **copy** the shortened URL, **share** it via compatible platforms, or **open** the shortened URL in a new browser tab.

- **Responsive Design**: The design adjusts based on screen size for mobile and desktop users.

2. **Backend (Flask/Python)**:
    - **URL Mapping**: The backend uses a dictionary (`url_mapping`) to store original URLs and their corresponding shortened URLs.
    - **Hash Generation**: The `generate_hash()` function creates a 6-character hash of the original URL using **MD5**.
    - **Shortening Process**: The URL is checked if it's already shortened. If not, a new hash is generated.
    - **Redirection**: The application listens for routes with shortened URLs and redirects them to the original URL using the `redirect_to_url()` function.
3. **Frontend and Backend Interaction**:
    - The frontend sends a **POST** request to `/shorten` with the URL to be shortened.
    - The backend returns the shortened URL, which is then displayed on the page.
    - The frontend allows users to interact with the shortened URL (copy, share, open) through JavaScript event listeners.