

Abstract

The **URL Shortener Web Application** is a lightweight and user-friendly solution developed using **ASP.NET Core MVC** that enables users to convert long, complex URLs into short and manageable links. This project demonstrates the integration of core web development concepts such as **Model-View-Controller architecture**, **session management**, **form validation**, and **routing**.

Users can **register** and **log in** to the system, after which they can input any valid long URL to receive a **randomly generated short URL**. The application maintains a list of all URLs shortened by each user during their session, allowing them to **view previously created short links**. Data is stored in-memory using a **fake user store**, simulating database operations for demonstration purposes.

The project also outlines potential **RESTful API endpoints** for real-world integration, supporting operations like:

- **Shortening a long URL** (/api/Url/generateShortUrl)
- **Retrieving the original URL** using a shortened link (/api/Url/getOriginalUrl)

With a clean, responsive UI built using **Bootstrap 5**, this application serves as a practical example for learning web application development, particularly in areas such as **authentication**, **state handling**, **form processing**, and **microservice-style thinking**.

CHAPTER 2: INTRODUCTION

1.1 Problem Statement

In today's digital landscape, Uniform Resource Locators (URLs) are fundamental to accessing online resources, navigating websites, and sharing information across digital platforms. Despite their importance, URLs are often long, complex, and difficult to manage, especially when shared via social media, emails, or printed materials. These lengthy links not only affect readability but also reduce the efficiency of communication by increasing the risk of link breakage or user error during manual copying.

Public URL shortening services such as Bitly and TinyURL provide a temporary solution by converting long URLs into compact versions. However, these services come with several limitations. Most do not offer session-based tracking, user-specific link management, or customizable options, which are essential for users who require personalized control over their URLs. Additionally, such third-party services may not guarantee data privacy, nor are they designed for educational or developmental environments where hands-on experience with the underlying architecture is critical.

Another significant gap exists in the availability of URL shorteners built using modern frameworks like ASP.NET Core MVC. For students, developers, and educators seeking to learn full-stack web development within the .NET ecosystem, there is a lack of practical, open-source examples that incorporate essential web concepts such as MVC architecture, session handling, form validation, authentication, and RESTful API design.

This project aims to address these issues by developing a lightweight, session-based URL shortener web application using ASP.NET Core MVC. The system allows users to register, log in, shorten URLs, and manage their personalized URL history during an active session, providing both functional value and educational insight into modern web development practices.

1.2 Purpose/Objective and Goals

The main **purpose** of this project is to develop a custom **URL Shortener Web Application** using **ASP.NET Core MVC** that enables authenticated users to:

- Register and log in securely.
- Shorten long URLs into unique, compact short links.
- Maintain a session-based history of previously shortened URLs.
- Provide RESTful API endpoints to facilitate scalability and integration.
- Learn and demonstrate key web development concepts such as:
 - MVC architecture
 - Session management
 - Form validation
 - Routing and controller logic
 - API design

The **goals** include:

- Improving the usability and shareability of URLs.
- Providing a lightweight and responsive user interface.
- Building an educational tool for students and developers exploring ASP.NET Core MVC.
- Laying a scalable foundation for future enhancements like analytics, database integration, and mobile app support.

1.3 Project Scope and Limitations

Scope:

- A web-based URL shortener for registered users.
- Session-based URL tracking that is private and user-specific.
- In-memory storage (FakeUserStore) used to simulate database operations.
- Responsive front-end UI using Bootstrap 5.
- RESTful API endpoints for shortening and retrieving original URLs.
- User-friendly interface with basic form validation and authentication.

Limitations:

- Data is not persisted beyond the session (no real database used).
- No advanced analytics or click tracking in the current version.
- Security features like encryption, CAPTCHA, and rate limiting are not fully implemented.
- Limited to single-user session storage (not scalable for production use).
- Custom short code functionality is not yet available.

CHAPTER 2: SYSTEM ANALYSIS

2.1 Scope and Limitations of the Existing System

In the current environment, users share long and complex URLs directly, which often creates several usability challenges:

Scope of the Existing System:

- Users can manually copy and share long URLs via social platforms or email.
- Public tools like Bitly or TinyURL offer basic shortening services.

Limitations:

- No user authentication or session-based link tracking.
- No personalization or control over generated short links.
- No user-specific URL history or management tools.
- No integration with .NET MVC frameworks for learning or enterprise use.
- Limited customization, data privacy, and extensibility.

2.2 Project Perspective and Features

Project Perspective:

This project is a web-based URL shortener developed using ASP.NET Core MVC with session-based user authentication and RESTful API integration. It is designed as both a practical tool and an educational platform for developers.

Key Features:

- **User Registration & Login** – Secure authentication and session-based access control.
- **URL Shortening** – Converts long URLs into 6-character short codes using GUID logic.
- **User-Specific URL Management** – Logged-in users can view their URL history during their session.
- **RESTful API Support** – Endpoints to generate and retrieve short URLs.
- **Responsive UI** – Built using Bootstrap 5 for clean and mobile-friendly interaction.

2.3 Stakeholders

The following are the main stakeholders involved in this project:

Stakeholder	Role/Interest
Project Developer	Designs and implements the system features using ASP.NET Core MVC.
End Users (Students, Professionals)	Use the application to shorten and manage URLs securely.
Educators/Trainers	Use the system as a learning tool to demonstrate MVC, API, and session management.
Future Admins	May manage user accounts, analytics, and system configurations.

2.4 Requirement Analysis

1. Functional Requirements

- User registration and secure login.
- Ability to input a long URL and generate a unique short URL.
- Maintain session-based history of shortened URLs for logged-in users.
- Logout functionality.
- REST API endpoints for:
 - Generating a short URL.
 - Retrieving the original URL from a short code.

2. Performance Requirements

- Application must respond to user actions (login, shorten URL) within 2 seconds.
- Should support at least 10 concurrent users without performance degradation (for demo purposes).
- URLs should be shortened instantly after submission.

3. Security Requirements

- Only authenticated users can access the URL shortening feature.
- User sessions must be securely maintained using ASP.NET Core session management.
- Passwords should be stored securely (hashed) when using a real database.
- Input validation to prevent malicious links or scripts.

CHAPTER 3: SYSTEM DESIGN

3.1 Design Constraints

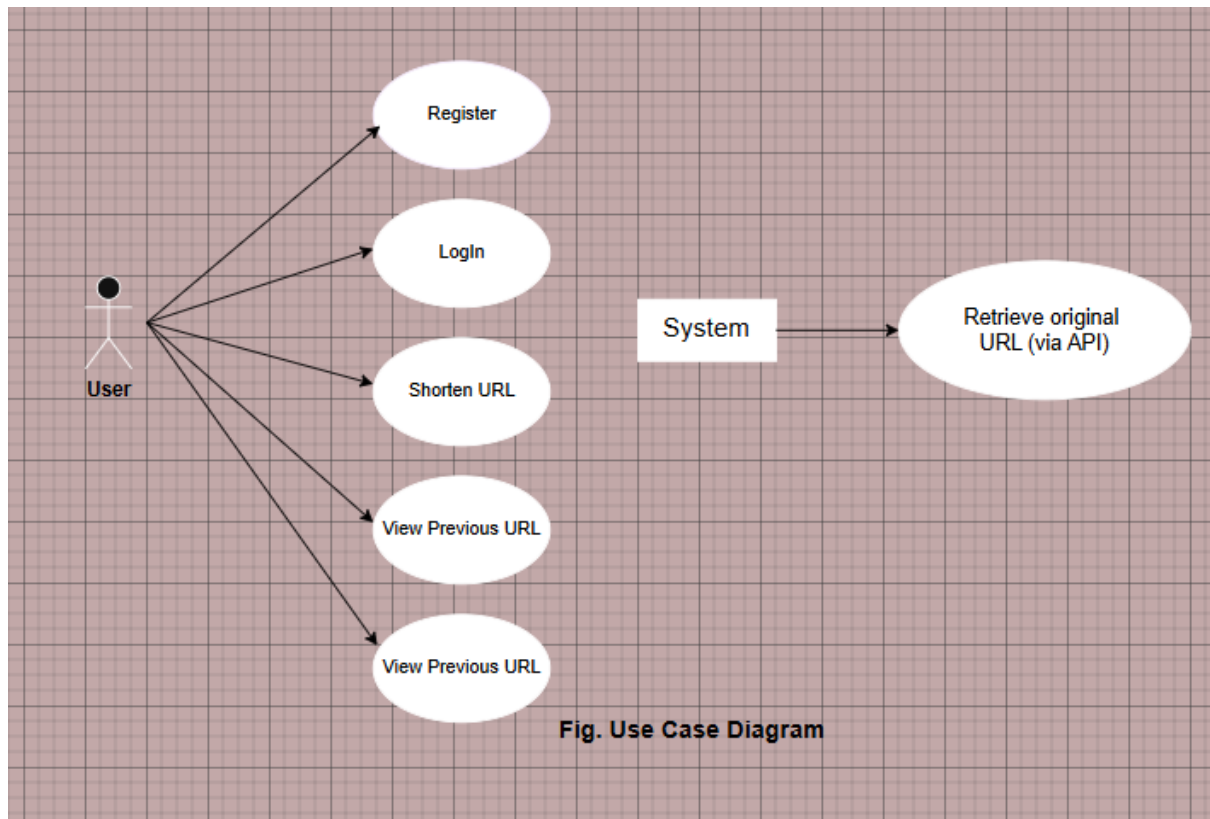
Design constraints are limitations that influence the system's architecture, user interface, and operational performance. These constraints help shape the overall system and ensure it aligns with the intended platform and user requirements.

Identified Design Constraints:

- **Framework Limitation:** Must be developed using ASP.NET Core MVC architecture.
- **In-Memory Data Store:** Uses FakeUserStore for data simulation; no real database support in the current version.
- **Session-Based Access:** URL history and user activity are limited to active sessions; data is lost after logout or session expiration.
- **Security Measures:** Basic validation and authentication only; advanced security (encryption, CAPTCHA, etc.) is not implemented.
- **Single User Type:** No role-based access control in the current build (admin/user distinction not available).
- **Short URL Generation:** Uses GUID-based short codes, not user-defined/custom codes.

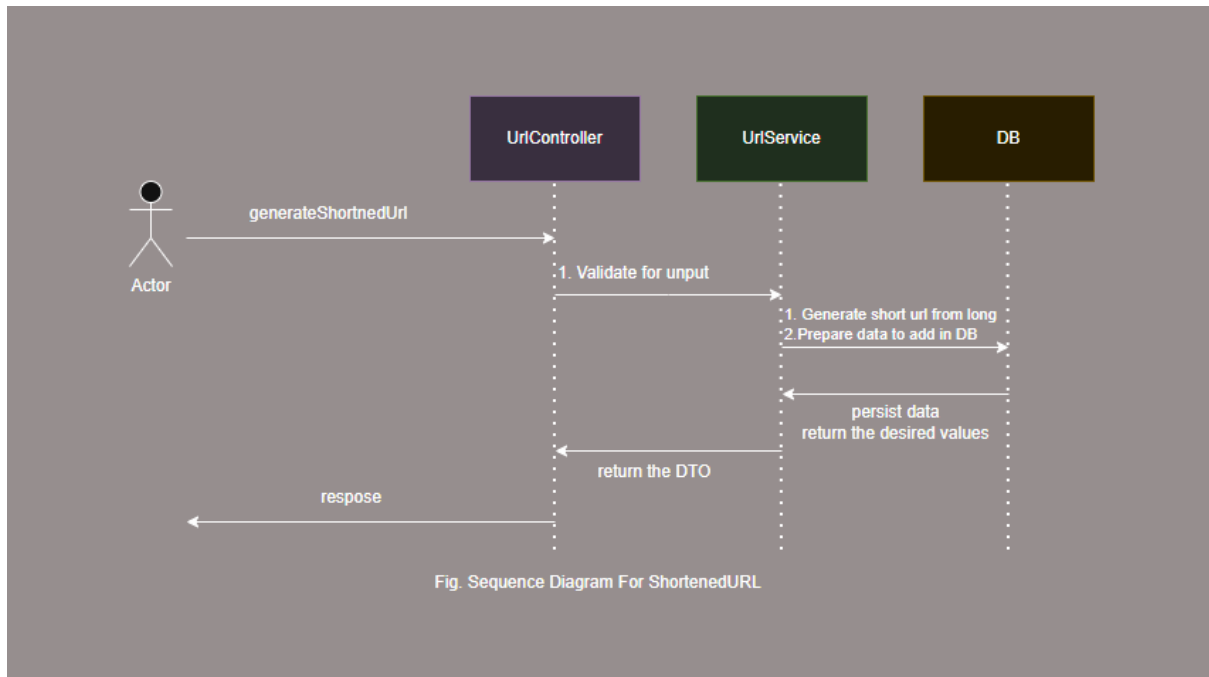
3.2 System Model: UML Diagrams

1. Use Case Diagram

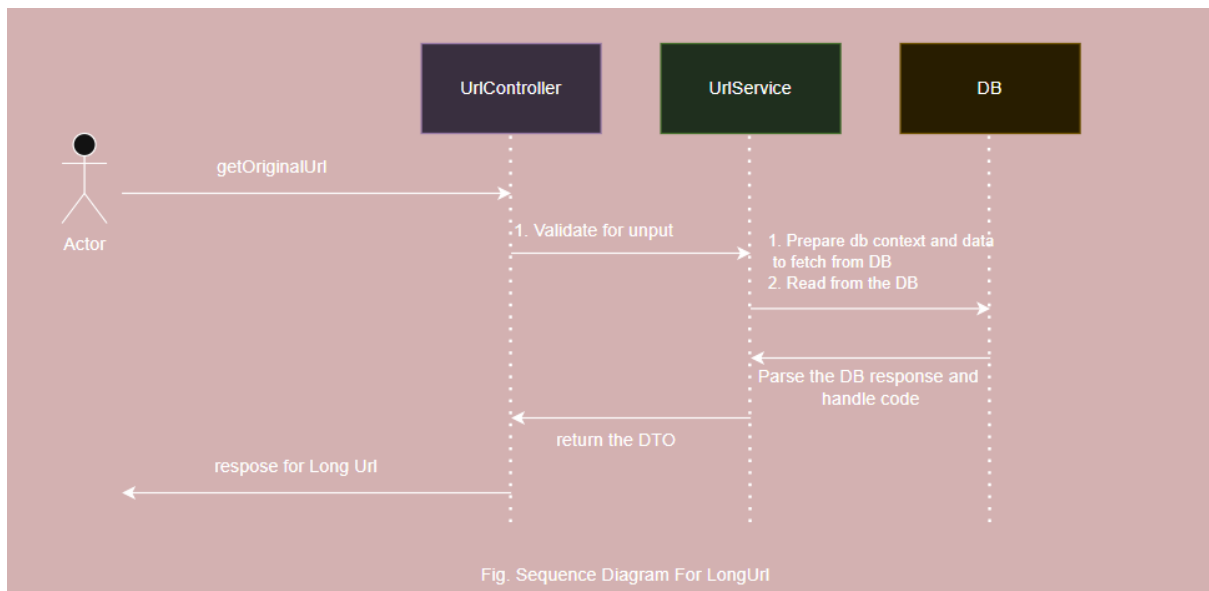


2. Sequence Diagram

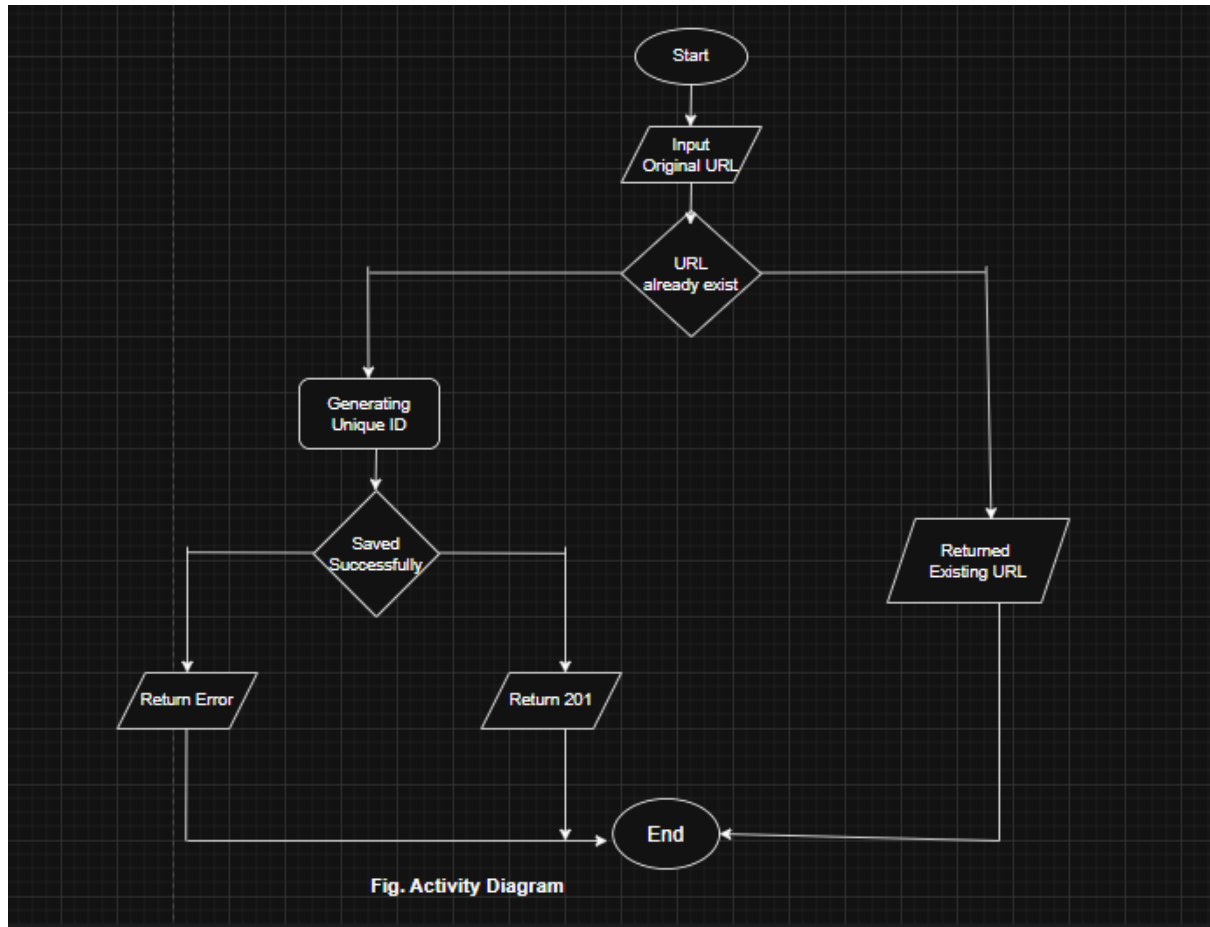
Shorten URL



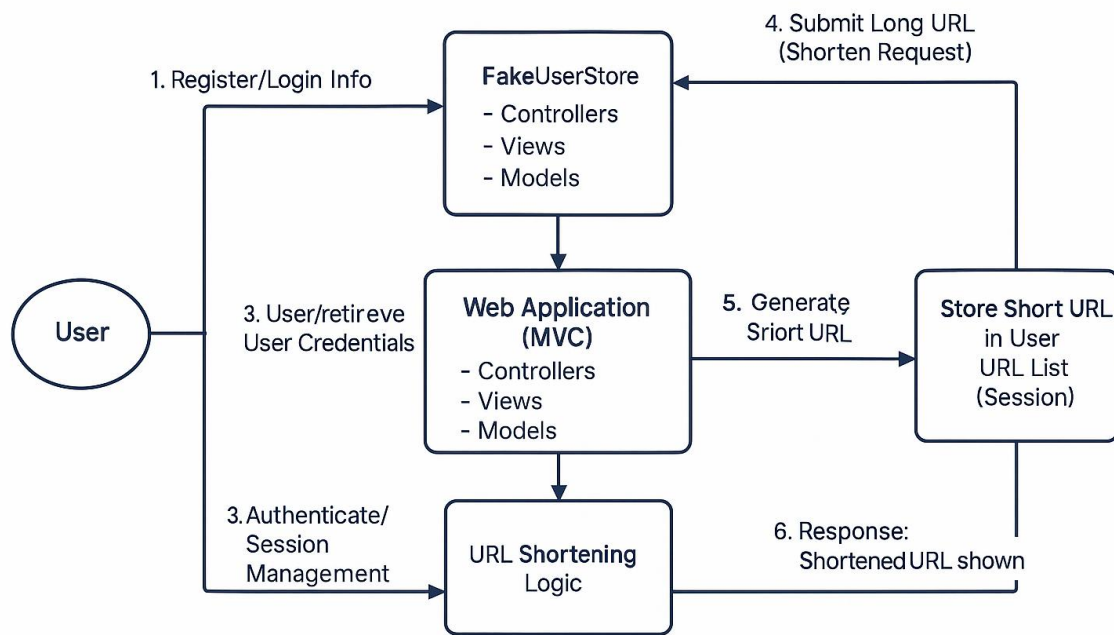
Long URL



3. Activity Diagram (Login + URL Shortening)



4. Data Flow Diagram



Data Flow Diagram (DFD-1)

3.3 Data Model

Entities:

1. UserModel

Field	Type	Description
Username	string	Unique identifier
Password	string	User's password
Urls	List<ShortUrl>	URLs shortened by this user

2. ShortUrl

Field	Type	Description
OriginalUrl	string	Long URL submitted by user
ShortenedUrl	string	6-character shortened URL

3. FakeUserStore (In-memory)

Field	Type	Description
Users	List<UserModel>	Simulates persistent storage

3.4 User Interfaces

The system features a clean, responsive UI designed with Bootstrap 5 & HTML.

Main Screens:

1. Login Page

- Fields: Username/Email, Password
- Actions: Login Button, Register Link

2. Registration Page

- Fields: Username, Email, Password, Confirm Password
- Actions: Register Button

3. Dashboard/Home Page

- Input: Long URL textbox
- Action: “Shorten” Button
- Display: Table of previously shortened URLs (ID, Original, Short, Clicks)

4. URL History Page

- Table format with:
 - Original URL
 - Shortened URL
 - Click Count (optional)
 - Actions (Delete/Edit if implemented)

5. Redirect Page

- Auto-redirects users from the short URL to the original URL
- No visible UI, only redirection logic

6. Error Page (Optional)

- Displays validation errors or access issues
- Includes navigation back to home or login

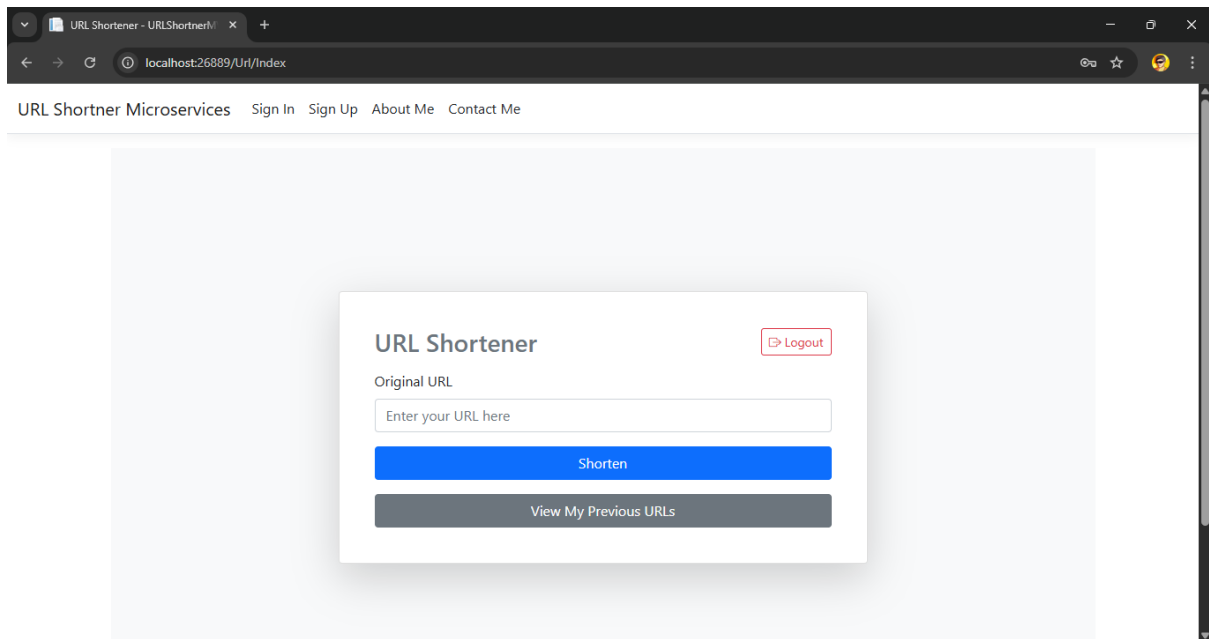
Screenshot Login Page

The screenshot shows a web browser window with the title "Login - URLShortnerMVC_Proj". The address bar displays "localhost:26889". The page header includes the text "URL Shortner Microservices" followed by navigation links: "Sign In", "Sign Up", "About Me", and "Contact Me". The main content area features a white login form centered on a light gray background. The form has the title "URL Shortner Microservices" in blue. It contains two input fields: "Username" with a placeholder "Enter your username" and "Password" with a placeholder "Enter your password". Below these fields is a blue "Login" button. At the bottom of the form, there is a link: "Don't have an account? [Register here](#)".

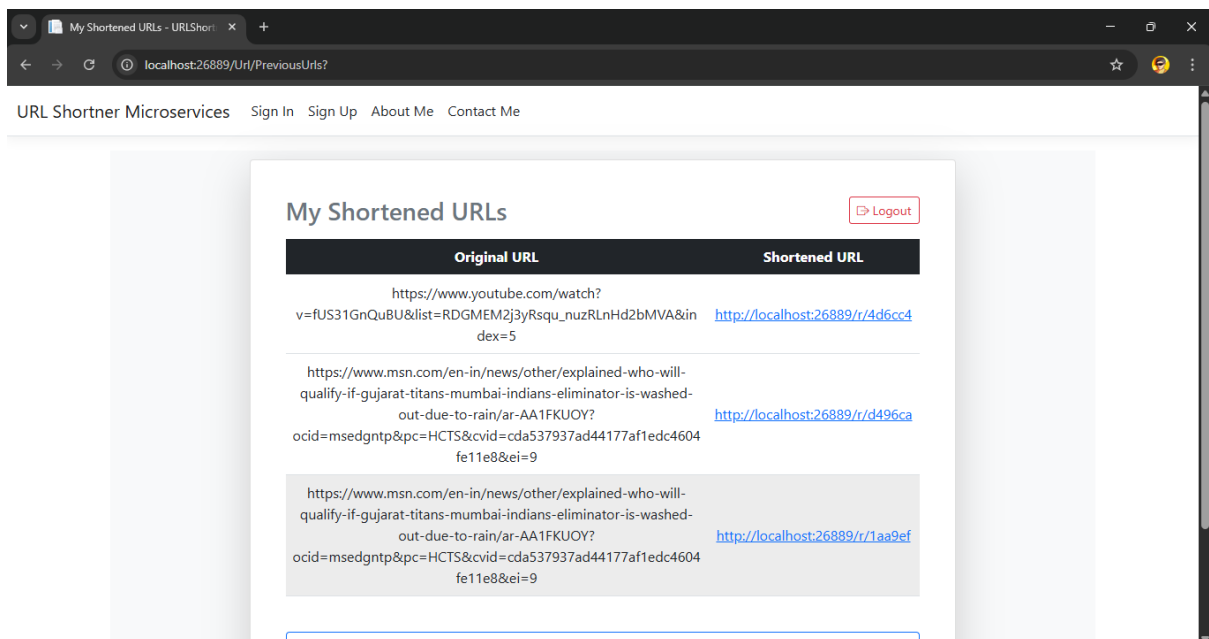
Register Page

The screenshot shows a web browser window with the title "Register - URLShortnerMVC_Proj". The address bar displays "localhost:26889/Account/Register". The page header includes the text "URL Shortner Microservices" followed by navigation links: "Sign In", "Sign Up", "About Me", and "Contact Me". The main content area features a white registration form centered on a light gray background. The form has the title "Create an Account" in green. It contains two input fields: "Username" with a placeholder "Choose a username" and "Password" with a placeholder "Create a password". Below these fields is a green "Register" button. At the bottom of the form, there is a link: "Already have an account? [Login here](#)".

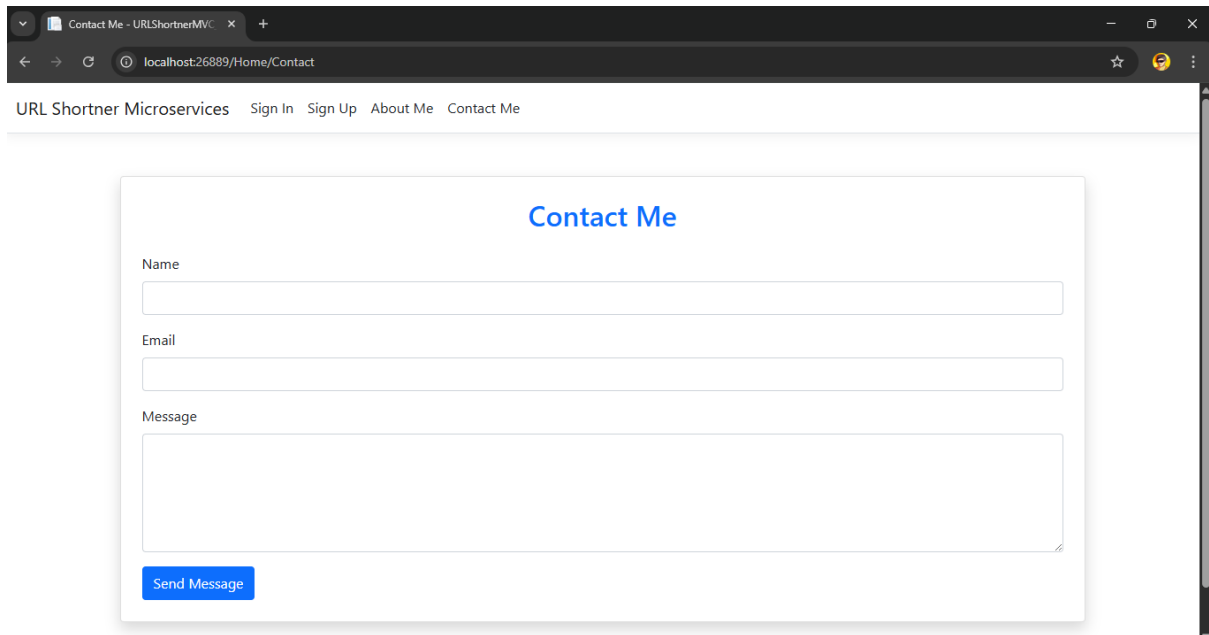
Home Page / Index



Original URL View Page



Contact Page



The screenshot shows a web browser window with the title "Contact Me - URLShortnerMVC". The address bar displays "localhost:26889/Home/Contact". The navigation bar includes links for "URL Shortner Microservices", "Sign In", "Sign Up", "About Me", and "Contact Me". The main content area features a "Contact Me" form with the following fields:

- Name**: A text input field.
- Email**: A text input field.
- Message**: A large text area for the message.
- Send Message**: A blue button to submit the form.

CHAPTER 4: IMPLEMENTATION DETAILS

4.1 Hardware Specification

The system is designed to be lightweight and does not require high-end hardware. Below are the recommended hardware specifications for development and testing:

Component	Specification
Processor (CPU)	Intel Core i3 or higher
RAM	Minimum 4 GB (Recommended: 8 GB or more)
Storage	At least 500 MB of free disk space
Display	Minimum resolution: 1024 × 768
Input Devices	Standard keyboard and mouse
Internet	Required for package downloads and API testing

These specifications are suitable for running the application locally using Visual Studio or Visual Studio Code and testing its functionality through a browser.

4.2 Software Specification

The software environment used to develop and run the URL Shortener Web Application includes both development tools and runtime dependencies:

Category	Requirement
Operating System	Windows 10/11 or equivalent
Framework	.NET 6 SDK or later
IDE	Visual Studio 2022 or Visual Studio Code
Language	C#
Web Browser	Google Chrome / Microsoft Edge / Firefox
UI Framework	Bootstrap 5 (CDN-based, no local install needed)
Operating System	Windows 10/11 or equivalent

Reports

This project is a web-based URL Shortener Application developed using ASP.NET Core MVC. The primary aim of this project is to convert long and complex URLs into short, manageable links that are easier to share, remember, and use across various digital platforms. The system supports user registration and login, allowing authenticated users to generate short URLs and view their session-based history. The project is structured around the Model-View-Controller architecture, ensuring clean code organization and separation of concerns.

The URL shortening functionality uses a 6-character random code generation method based on GUID logic. For the purpose of demonstration and lightweight operation, all user and URL data are stored in-memory using a fake user store. This simulates real database operations without the need for persistent storage, making it ideal for learning and prototyping. A responsive user interface is built using Bootstrap 5 to ensure mobile compatibility and an intuitive experience.

In addition to its core functionality, the application provides RESTful API endpoints that allow external systems to interact with it, such as generating a short URL or retrieving the original link using the short code. This adds scalability and integration potential for future enhancements. Testing was conducted at various levels including unit, integration, system, and user acceptance, and the application performed successfully under all test cases.

Overall, this project serves both as a functional URL shortening tool and an educational platform for understanding ASP.NET Core MVC, session handling, authentication, routing, and web API development. It lays a strong foundation for future upgrades like database integration, analytics, custom short codes, and enhanced security features.

CHAPTER 5: TESTING

5.1 Introduction

Testing is a critical phase in the Software Development Life Cycle (SDLC) to ensure that the application functions as intended, meets the requirements, and is free from defects. For the URL Shortener Web Application, a structured testing strategy was followed, covering all major components including user authentication, URL shortening logic, session management, and UI behavior.

5.2 Testing Strategy

A multi-level testing approach was adopted:

1. Unit Testing

- Focuses on individual components or methods.
- Example: Testing short URL generation logic.
- Tool Used: xUnit (for .NET Core)

2. Integration Testing

- Verifies interaction between modules (e.g., Controllers and FakeUserStore).
- Ensures end-to-end functionality works as expected.

3. System Testing

- Validates the complete system.
- Includes functional, usability, compatibility, and performance testing.

4. User Acceptance Testing (UAT)

- Conducted with end-users (students or developers).
- Ensures the system meets real-world expectations.

5.3 Test Case Design – User Module

Test Case ID	Description	Input	Expected Output	Status
UM_TC_01	Register with valid data	Valid username, email, password	Registration successful	Pass
UM_TC_02	Register with existing username	Duplicate username	Error: "Username already exists"	Pass
UM_TC_03	Register with invalid email	Wrong email format	Error: "Invalid email address"	Pass
UM_TC_04	Password and confirm password mismatch	Mismatched fields	Error: "Passwords do not match"	Pass
UM_TC_05	Missing mandatory fields	Empty form	Validation errors shown	Pass
UM_TC_06	Login with valid credentials	Correct username/password	Login successful, redirect to dashboard	Pass
UM_TC_07	Login with incorrect credentials	Wrong password	Error: "Invalid username or password"	Pass
UM_TC_08	Access dashboard without login	Direct URL	Redirect to login page	Pass
UM_TC_09	Logout and try accessing protected page	Logout, then visit dashboard	Redirect to login page	Pass

5.4 Test Case Design – URL Module

Test Case ID	Description	Input	Expected Output	Status
URL_TC_01	Shorten a valid long URL	https://example.com	Returns short URL (e.g., abc123)	Pass
URL_TC_02	Submit an invalid URL	http://wrong-url	Error: "Invalid URL format"	Pass
URL_TC_03	View previously shortened URLs	Active session	Display list of shortened URLs	Pass
URL_TC_04	Retrieve original URL via short code	Short code in URL path	Redirects to original URL	Pass
URL_TC_05	Access short URL after logout	Expired session	May return error or require login again	Pass

5.5 Performance and Compatibility Testing

- Application tested across modern browsers: Chrome, Edge, Firefox.
- Response time for URL shortening is under 1 second for local deployments.
- UI confirmed to be responsive on desktop, tablet, and mobile devices using Bootstrap 5.

5.6 Security Testing (Basic)

- Only authenticated users can access dashboard features.
- Session expires upon logout.
- Input validation prevents basic injection and malformed input.

Note: Advanced security testing such as SQL injection, token-based API access, rate limiting, and CAPTCHA is recommended for future implementation.

5.7 Conclusion of Testing

The system has passed all planned unit, integration, and system test cases successfully. It meets all functional requirements and provides a smooth, responsive user experience. Based on User Acceptance Testing feedback, the application is ready for deployment or extension with advanced features.

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The URL Shortener Web Application developed using **ASP.NET Core MVC** successfully addresses the modern challenges associated with sharing long and complex URLs by providing a simple, secure, and user-friendly platform. The project demonstrates essential concepts in full-stack web development, including:

- **MVC architecture**
- **User authentication and session management**
- **Form validation**
- **Routing and controller logic**
- **Responsive UI design using Bootstrap 5**
- **RESTful API design principles**

The application allows registered users to shorten long URLs, track their activity during a session, and interact with the system through a responsive interface. By using an in-memory data store (FakeUserStore), the project remains lightweight while simulating real-world operations. The system has been thoroughly tested for functionality, usability, and performance, making it reliable for demonstration or educational use.

Additionally, the project serves as a practical learning platform for students and developers seeking hands-on experience with .NET-based web applications. It balances simplicity and functionality, offering a solid foundation for future enhancements and deployment in real-world environments.

6.2 Future Scope

The current system offers core functionality but holds strong potential for further development. Future enhancements include integrating a real database SQL Server to persist user data across sessions. An analytics dashboard can be introduced to track URL clicks, traffic sources, and user engagement. The system could also support custom short codes, allowing users to create memorable branded URLs. Features like link expiration, manual link management, and advanced security (CAPTCHA, rate limiting, encryption) would improve safety and control. Expanding to mobile apps and browser extensions will increase accessibility and convenience. Multi-language support can make the platform usable by a global audience. Role-based access control can be added for administrative oversight. Finally, integration with third-party tools like email and social media platforms can enhance sharing and campaign tracking capabilities.