A Project Report On

AI Shortlinks

Submitted in partial fulfillment of the requirement for the award of the degree

MASTER OF COMPUTER APPLICATIONS (M.C.A.)

Academic Year 2025 – 26

Deep Fichadiya (92400582013)

Internal Guide
Prof. Divyakant Meva sir

Rajkot-Morbi Road, At & PO : Gauridad, Rajkot 360 003. Gujarat. India.

Faculty of Computer Applications (FCA)

Certificate

This is to certify that the project work entitled

AI Shortlinks

submitted in partial fulfillment of the requirement for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

(M.C.A.)

of the

Marwadi University

is a result of the bonafide work carried out by

Deep Fichadiya (92400584013)

during the academic year 2025-26

Faculty Guide	HOD	Dean

CERTIFICATE

DECLARATION

I / We hereby declare that this project work entitled **AI Shortlinks** is a record done by me/us.

I / We also declare that the matter embodied in this project is genuine work done by me / us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of any course of study.

Place:

Date:

Deep Fichadiya (92400584013) Signature: Deep M Fichaidya.

ACKNOWLEDGEMENT

It is indeed a great pleasure to express my/our thanks and gratitude to all those who helped me/us. No serious and lasting achievement one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

I/We am/are very thankful to my/our faculty guide **Prof. Divyakant Meva sir,** the person who makes me/us follow the right steps during project work. I/we express my/our deep sense of gratitude to his/her guidance, suggestions and expertise at every stage. Apart from that his/her valuable and expertise suggestion during documentation of my/our report indeed helped me/us a lot.

Thanks to my/our friends who have been a source of inspiration and motivation that helped me/us during my/our project work. I/we am/are heartily thankful to our external guide **<Name of External Guide>** for providing me/us an opportunity to work over this report and for their endless and great support. I/we am/are also thankful to my/our Head **Dr.Sunil Bajeja** and Dean **Dr. R. Sridaran** for their support as and when required. I/we would like to extend my/our gratitude towards all who directly or indirectly supported and helped me/us to accomplish my/our project work.

Deep Fichadiya (92400584013) Signature: Deep M Fichaidya.

COMPANY PROFILE

About Us – Smart10x

Smart10x is a forward-thinking IT solutions company focused on delivering innovative, reliable, and high-impact digital products. We design and develop scalable, secure, and performance-driven applications that help startups, enterprises, and digital businesses grow faster and smarter. Our goal is to multiply business efficiency and digital success by 10x through technology.

Our Services

1. **Web Development**
   We build modern, responsive, and high-speed websites and web applications using the latest technologies. Our solutions include custom websites, web apps, dashboards, admin panels, and API-based systems tailored to business needs.
2. **Mobile App Development**
   Smart10x develops powerful Android, iOS, and cross-platform mobile applications with a strong focus on performance and user experience. We handle the entire development lifecycle from idea to launch and maintenance.
3. **Web Scraping & Automation**
   We offer smart data extraction and automation solutions that help businesses collect, analyze, and utilize structured web data. Our systems are secure, efficient, and highly customizable.
4. **Desktop Application Development**
   We create robust desktop applications for Windows, macOS, and Linux with clean UI design, high performance, and strong offline capabilities.
5. **SEO & Digital Optimization**
   Smart10x helps businesses strengthen their online presence through advanced SEO strategies, keyword research, content optimization, technical audits, and performance tracking to improve search rankings and visibility.

Our Vision

To become a trusted technology partner that transforms businesses through smart, scalable, and future-ready digital solutions.

Our Mission

To provide innovative, efficient, and cost-effective IT services that empower clients to achieve sustainable growth in the digital ecosystem.

Why Choose Smart10x?

• Experienced team of skilled developers and tech specialists

• Timely project delivery with clear and transparent communication

• High-quality coding practices and scalable system design

• Client-focused approach with reliable long-term support

This version keeps the same professional structure as PraxisCore but adds a more growth-driven, performance-oriented identity suitable for the name **Smart10x**.

CONTENTS

FIGURE INDEX

TABLE INDEX

1. INTRODUCTION TO PROJECT DEFINITION

This project is a full-stack URL Shortener SaaS Platform designed to allow users to create short, secure links while providing administrators with tools to monitor and manage all URL activity. The system is built using Next.js 14 (App Router) for the frontend and backend, with Drizzle ORM + PostgreSQL as the database layer.

User Features:

Users can register, log in, shorten long URLs, create custom aliases, edit or delete their links, and view analytics such as click count, device information, and redirect history. A built-in URL safety check system analyzes links and flags unsafe or suspicious URLs before they are shared.

Admin Features:

Administrators have access to a dedicated dashboard where they can manage all users, update user roles, monitor all shortened URLs, review flagged or unsafe URLs, and take actions such as approving, blocking, or removing harmful links. Admins can also access system-wide analytics and perform database seeding for initial setup.

Authentication & Security:

The system uses NextAuth for authentication, session management, and role-based access control (RBAC). Additional security measures include server-side validations using Zod, URL safety scanning, and protection against invalid or malicious URLs.

Database Structure:

The backend is powered by Drizzle ORM, with structured schemas for users, URLs, analytics, and safety checks. The database supports relational mapping, efficient queries, and reliable storage required for SaaS operations.

Purpose:

The project aims to provide a modern, scalable, and user-friendly URL management service that enables individuals to shorten and track URLs easily, while giving administrators full control over safety, user roles, and platform oversight. It delivers a complete SaaS solution suitable for real-world deployment and business use.

2. PREAMBLE

This project presents a full-stack **URL Shortener SaaS Platform** designed to simplify link management for users and provide administrators with complete oversight of URLs, analytics, and safety controls. The system ensures secure URL handling, efficient database management, and a modern user dashboard built with Next.js 14.

2.1 General Introduction

URL shortening has become essential for improving link usability, especially when sharing long or complex URLs across digital platforms. This system allows users to generate short URLs, customize aliases, and track analytics while ensuring link safety. The platform includes a dedicated admin panel for managing users, URLs, roles, and flagged content, making it suitable for real-world SaaS deployment.

2.2 Statement of Problem

Long URLs are difficult to share, track, and manage. Traditional URL shorteners often lack built-in security, analytics, or administrative control. Users face challenges such as identifying unsafe URLs, tracking performance, and managing multiple links. Administrators lack visibility into abusive or harmful content shared through shortened links.

This project addresses these issues by providing a secure, analytics-enabled, and admin-controlled URL shortening platform.

2.3 Objective and Scope of the Study

Objectives:

- To develop a secure platform for generating short URLs.
- To allow users to manage their URLs and track detailed analytics.
- To integrate a URL safety detection system for identifying harmful links.
- To provide an admin dashboard with role management and monitoring features.
- To implement authentication and role-based access control for security.

Scope:

The system covers URL creation, user authentication, analytics tracking, safety checks, and admin-level management of users and URLs. It operates as a SaaS platform suitable for businesses, developers, or general users requiring reliable URL management.

2.4 Module Description with Functionality

1. Authentication Module

Handles user registration, login, sessions, and role-based access using NextAuth and Zod validations.

2. URL Management Module

Allows users to shorten URLs, create custom aliases, edit or delete links, check safety, and view analytics.

3. Redirection Module

Resolves short URLs, verifies safety status, records click analytics, and redirects users to the original link.

4. Admin Module

Provides functionality to manage all users, update roles, monitor URLs, and review or block unsafe links.

5. Database Module

Built using Drizzle ORM and PostgreSQL to store users, URLs, analytics, and flagged data with efficient relational queries.

6. Analytics Module

Tracks link performance, including click count, device type, browser, and time of access.

2.5 Methodology

The project follows a modular, full-stack development approach using:

- Next.js 14 (App Router): For frontend UI, backend server actions, and routing.
- Drizzle ORM + PostgreSQL: For secure and structured data handling.
- NextAuth: For authentication and RBAC.
- Zod: For server-side input validation.
- Tailwind + ShadCN: For clean and responsive user interface.

Development steps include requirement analysis, module design, database schema creation, coding, integration, testing, and deployment.

2.6 Feasibility Study

Technical Feasibility:
The system is built using widely adopted technologies (Next.js, PostgreSQL, Drizzle) that ensure scalability, maintainability, and performance.

Operational Feasibility:
Users and admins can easily interact with the system through dedicated dashboards. Workflows like URL creation, monitoring, and management are simple and intuitive.

3. REVIEW OF LITERATURE

The study of existing URL shortening systems provides insights into current industry standards, common features, limitations, and opportunities for improvement. Modern URL shorteners focus on usability, speed, analytics, and reliable redirection. However, many platforms lack built-in safety checks or an integrated admin dashboard for monitoring misuse. This project analyzes leading URL shortener solutions to identify essential functionalities and gaps that can be addressed through an optimized SaaS model.

3.1 Study of Existing Similar Types of Systems

1. Bitly

Bitly is one of the most widely used URL shorteners and offers features such as link shortening, branded domains, and detailed analytics. It provides enterprise-level tracking but does not offer an open safety scanning system or an integrated admin panel for monitoring user behavior. Many advanced features require paid subscriptions.

2. TinyURL

TinyURL is a simple link-shortening service that provides quick, no-login URL generation. However, it lacks modern features such as analytics, user accounts, role management, and URL safety detection. The absence of an admin monitoring system limits its security capabilities.

Summary:

Existing systems offer strong URL shortening and analytics features, but most lack:

- Built-in URL safety scanning
- A full admin dashboard
- Role-based access control
- Modern Next.js-based architecture

The present project addresses these gaps by combining ease of use, analytics, strong security features, and admin-level oversight into a unified SaaS platform.

4. TECHNICAL DESCRIPTION

This section outlines the technical requirements and specifications necessary for the successful development, testing, and deployment of the URL Shortener SaaS platform. The system is built using a modern full-stack architecture that leverages Next.js for the frontend and backend, PostgreSQL for database storage, and Drizzle ORM for schema management. The technical description covers both hardware and software requirements essential for running the application efficiently during development as well as production deployment.

4.1 Hardware Requirements

Development Environment:

- Processor: Intel i5 / AMD Ryzen 5 or higher
- RAM: Minimum 8 GB (16 GB recommended for smoother builds)
- Storage: At least 20 GB free space
- Operating System: Windows 10/11, macOS, or Linux
- Internet Connection: Required for package installation, authentication, deployment, and API interactions
- Server / Deployment (Cloud Hosting):
- Cloud Platform: Vercel, Railway, Render, Supabase, or any VPS
- CPU: 1 vCPU or higher
- RAM: 1–2 GB for small deployments
- Storage: 5–10 GB for logs, database, and project files
- Database Hosting: Managed PostgreSQL instance (Neon, Supabase, Railway, etc.)

These specifications ensure stable performance for both development and hosting.

4.2 Software Requirements

Frontend & Backend Stack:

- Next.js 14 (App Router) – Main framework for UI, routing, and server actions
- React – Component-based frontend development
- TypeScript – Strongly-typed scripting language
- Tailwind CSS – Styling and responsive UI
- ShadCN UI – Pre-built UI components

Backend Functional Tools:

- Node.js (v18 or higher) – Runtime environment
- NextAuth – Authentication and session management
- Zod – Validation of forms and server actions
- Drizzle ORM – Database schema and query management

PostgreSQL – Relational database for storing users, URLs, analytics, and safety logs

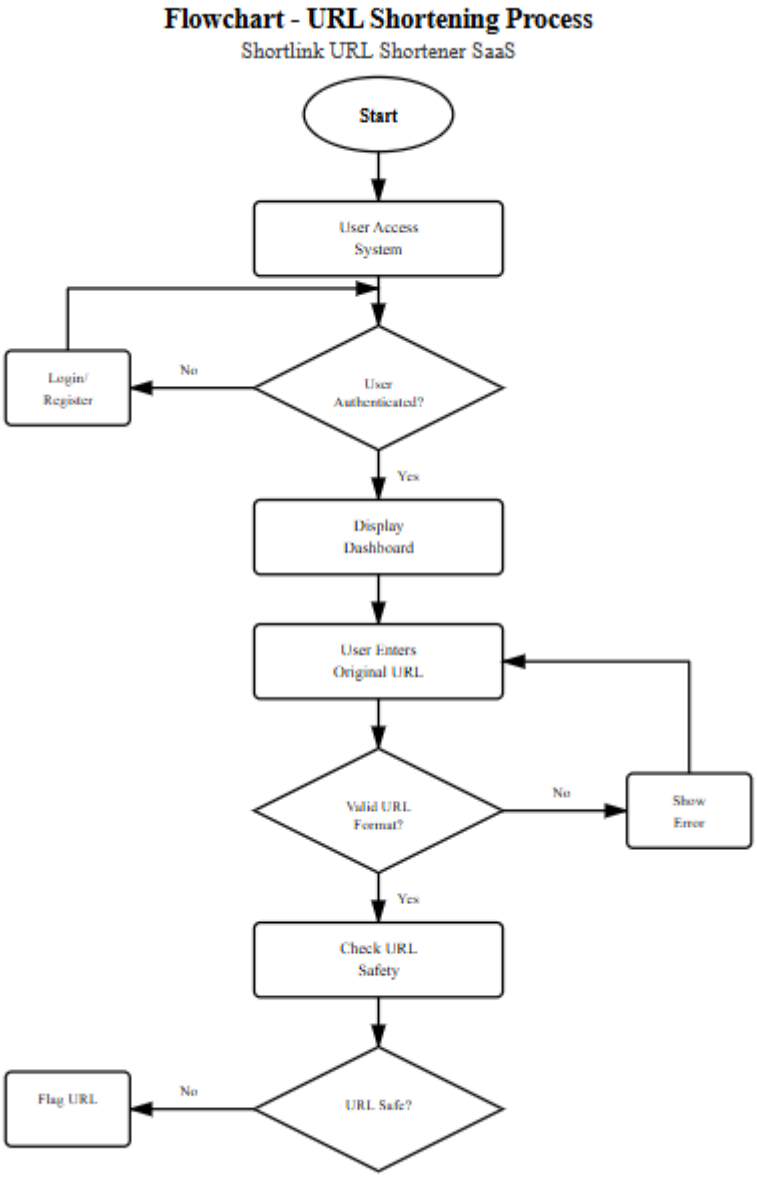
Deployment Software:

- Vercel (recommended) for hosting the Next.js application

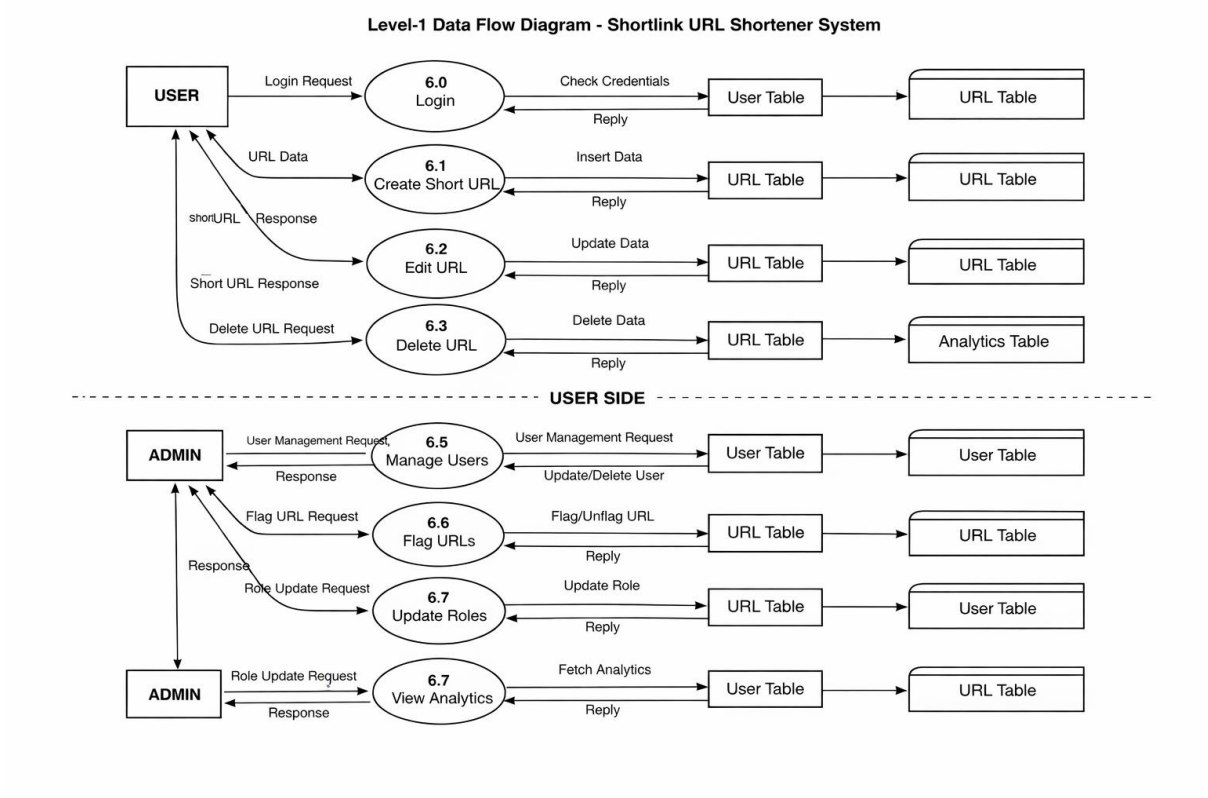
5. SYSTEM DESIGN AND DEVELOPMENT

This section outlines the architectural structure and system flow of the URL Shortener SaaS platform. The design demonstrates how users, administrators, backend services, and the database interact to provide secure URL shortening, analytics tracking, and system monitoring.

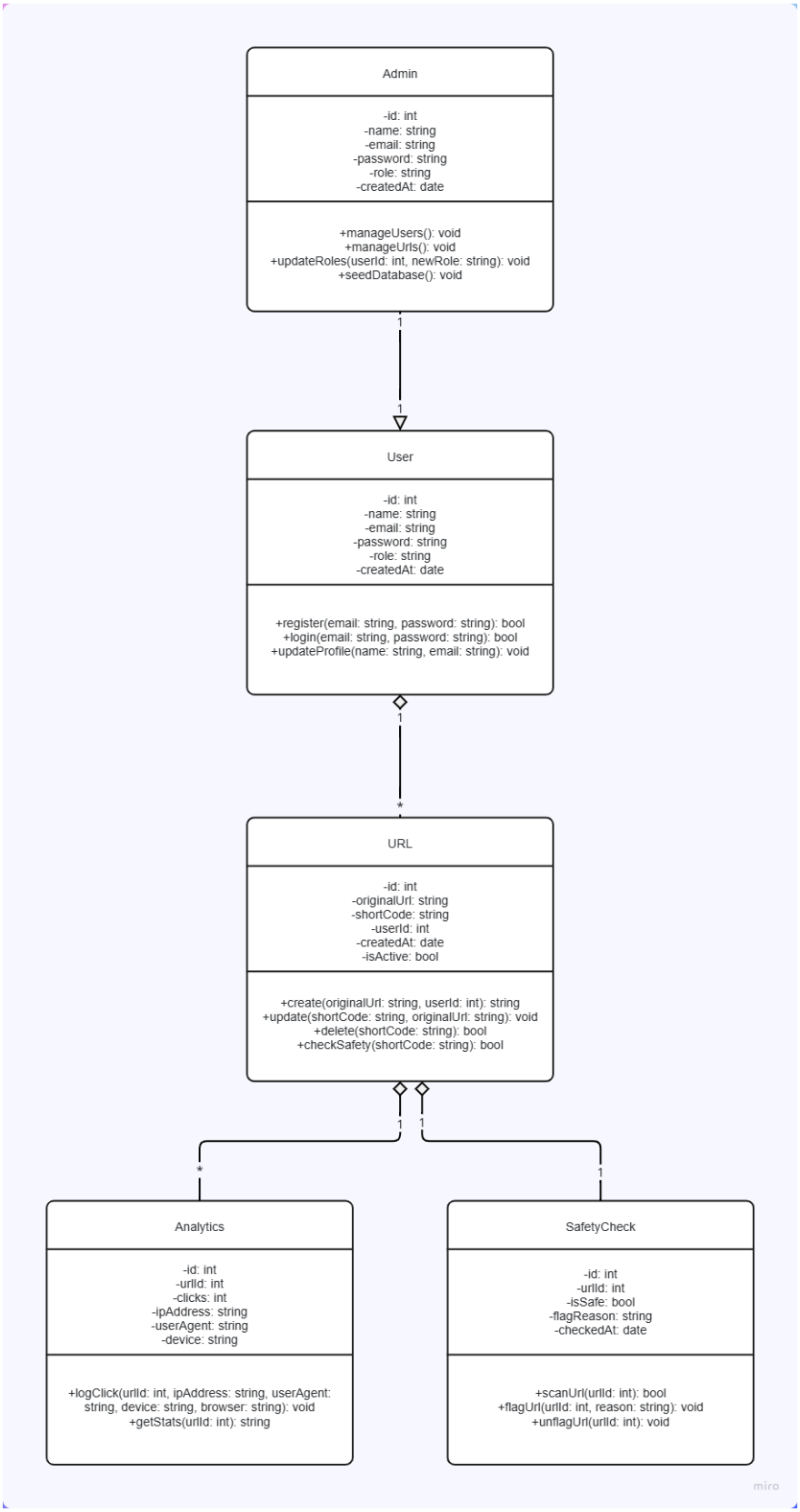
5.1 System Flow Chart



5.2 Data Flow Diagram

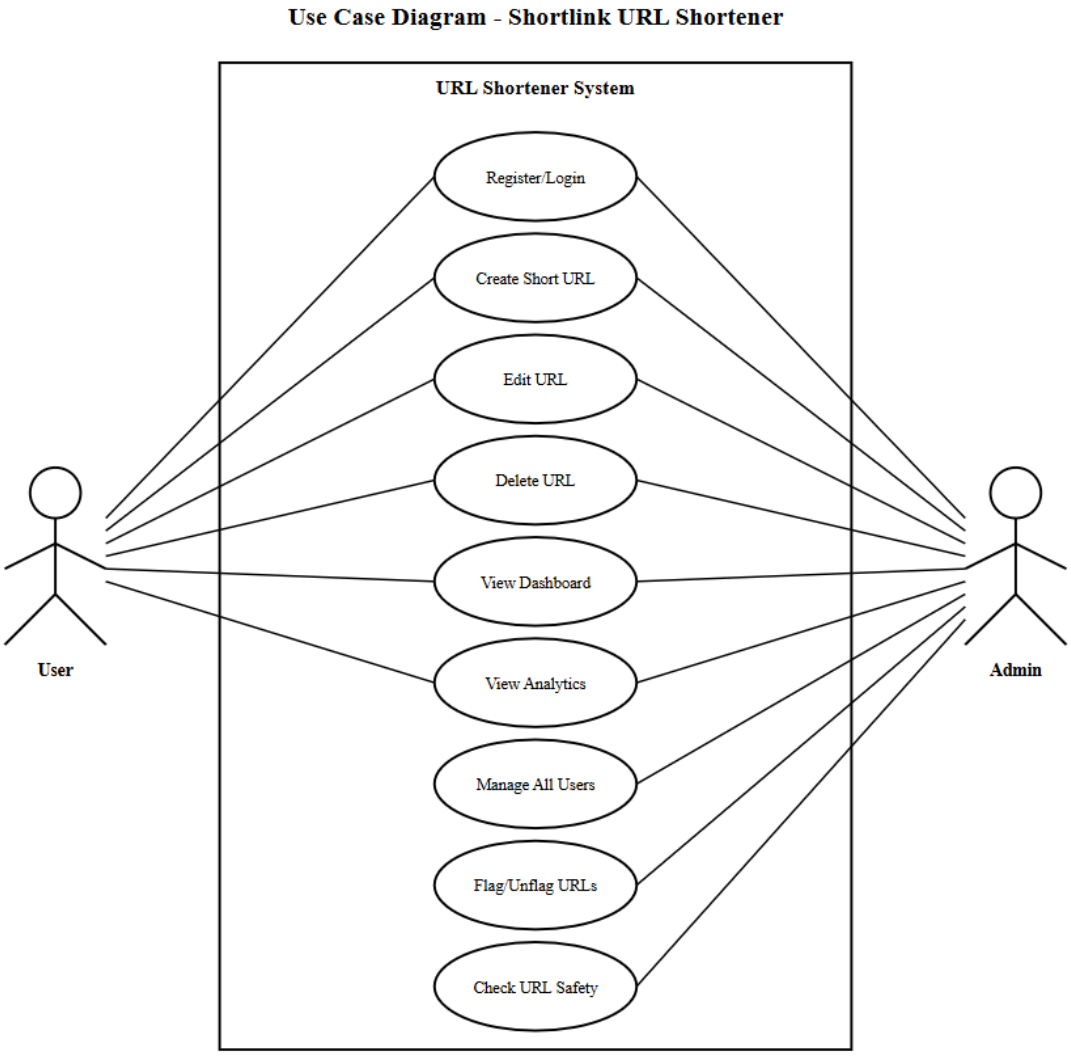


5.3 Class Diagram

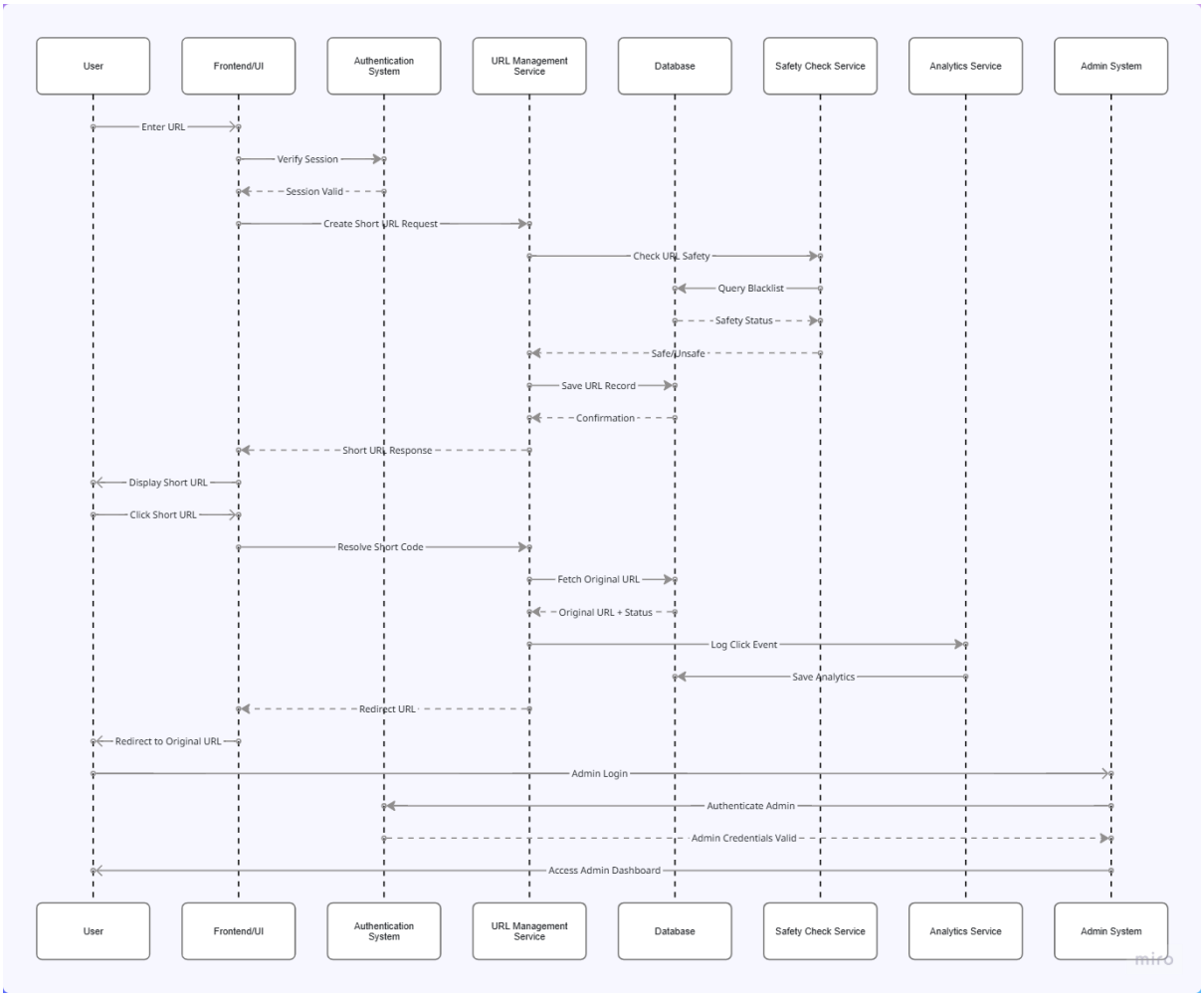


5.4 Behavioural Modeling

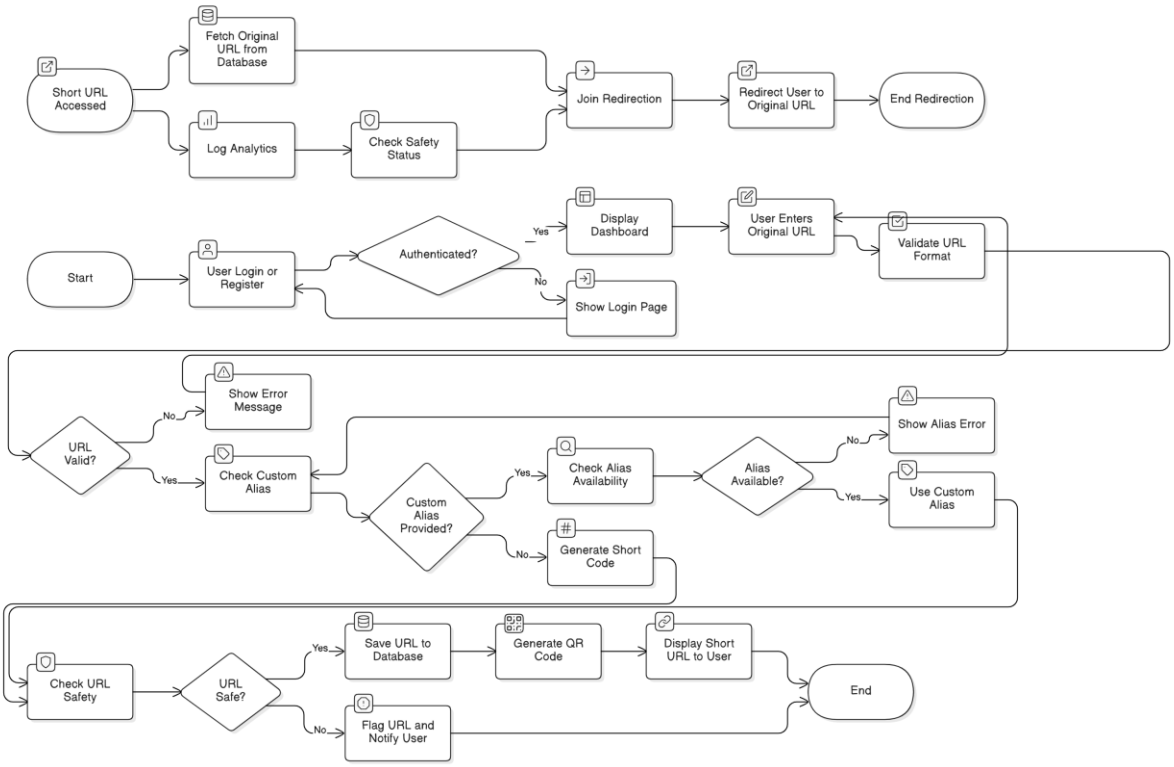
5.4.1 Use Case Diagram



5.4.2 Sequence Diagram



5.4.3 Activity Diagram



eraser

5.5 Database Design

5.5.1 Table Structure :-

table 5.5.1.1: users

Column Name	Data Type	Description
Id	VARCHAR(255)	Unique identifier
Name	VARCHAR(255)	User's full name
Email	VARCHAR(255)	User's email address
emailVerified	TIMESTAMP	Date and time when email was verified
image	TEXT	URL to user's profile image
password	TEXT	Hashed password for authentication
role	ENUM	User role - either 'user' or 'admin'
createdAt	TIMESTAMP	Account creation timestamp

table 5.5.1.2: accounts

Column Name	Data Type	Description
userId	VARCHAR(255)	Reference to user ID
type	VARCHAR(255)	Type of account
provider	VARCHAR(255)	OAuth provider name
providerAccountId	VARCHAR(255)	User's ID with the OAuth provider
refresh_token	TEXT	Refresh token
access_token	TEXT	Aceess token
expires_at	INTEGER	Token expiration timestamp
token_type	VARCHAR(255)	Type of token (Bearer, etc.)
scope	VARCHAR(255)	OAuth permission scopes granted
Id_token	TEXT	OpenID Connect ID token
Session_state	VARCHAR(255)	OpenID Connect ID token

table 5.5.1.3: sessions

Column Name	Data Type	Description
sessionToken	VARCHAR(255)	Unique session identifier
userId	VARCHAR(255)	Reference to user Id
expires	TIMESTAMP	Session expiration date and time

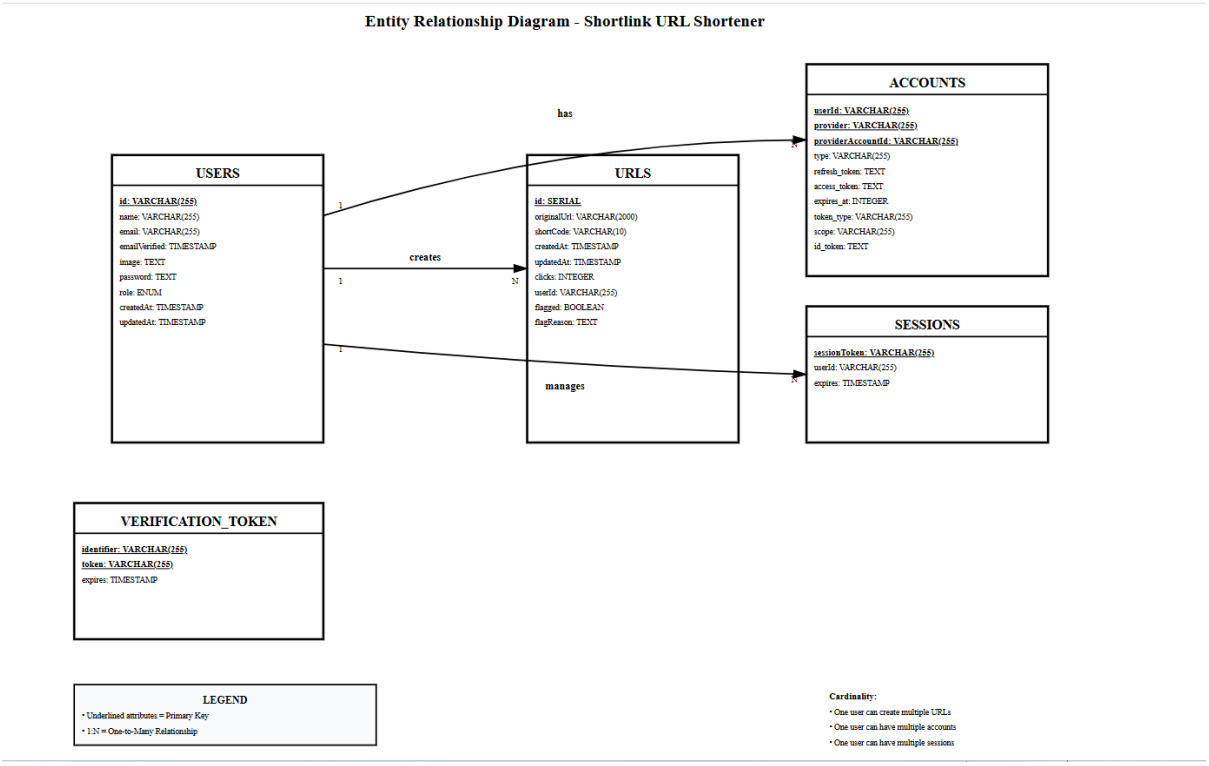
table 5.5.1.4: verification_token

Column Name	Data Type	Description
identifier	VARCHAR(255)	Unique identifier
verification token	VARCHAR(255)	Unique verification token
expires	TIMESTAMP	Token expiration date and time

table 5.5.1.5: urls

Column Name	Data Type	Description
id	SERIAL	Auto-incrementing unique identifier
originalUrl	VARCHAR(255)	Original long URL to be shortened
shortCode	VARCHAR(255)	Unique short code for the URL
createdAt	TIMESTAMP	URL creation timestamp
updatedAt	TIMESTAMP	Last URL update timestamp
clicks	INTEGER	Total number of clicks on short UR
userId	INTEGER	Reference to user who created URL
flagged	BOOLEAN	Whether URL is flagged as unsafe
flagReason	VARCHAR(255)	Reason for flagging the URL (if flagged)

5.5.2 E.R Diagram



5.6 Screen Design

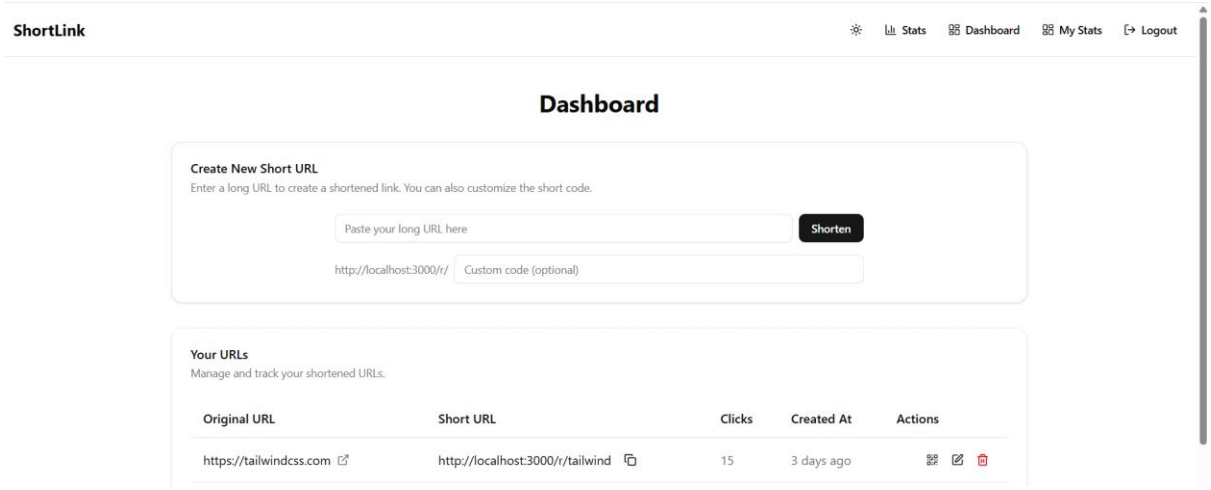


Figure 5.6.1 User Dashboard

This screen allows users to create new short URLs and optionally customize the short code. Below the form, users can view and manage their existing shortened links along with click counts and actions.

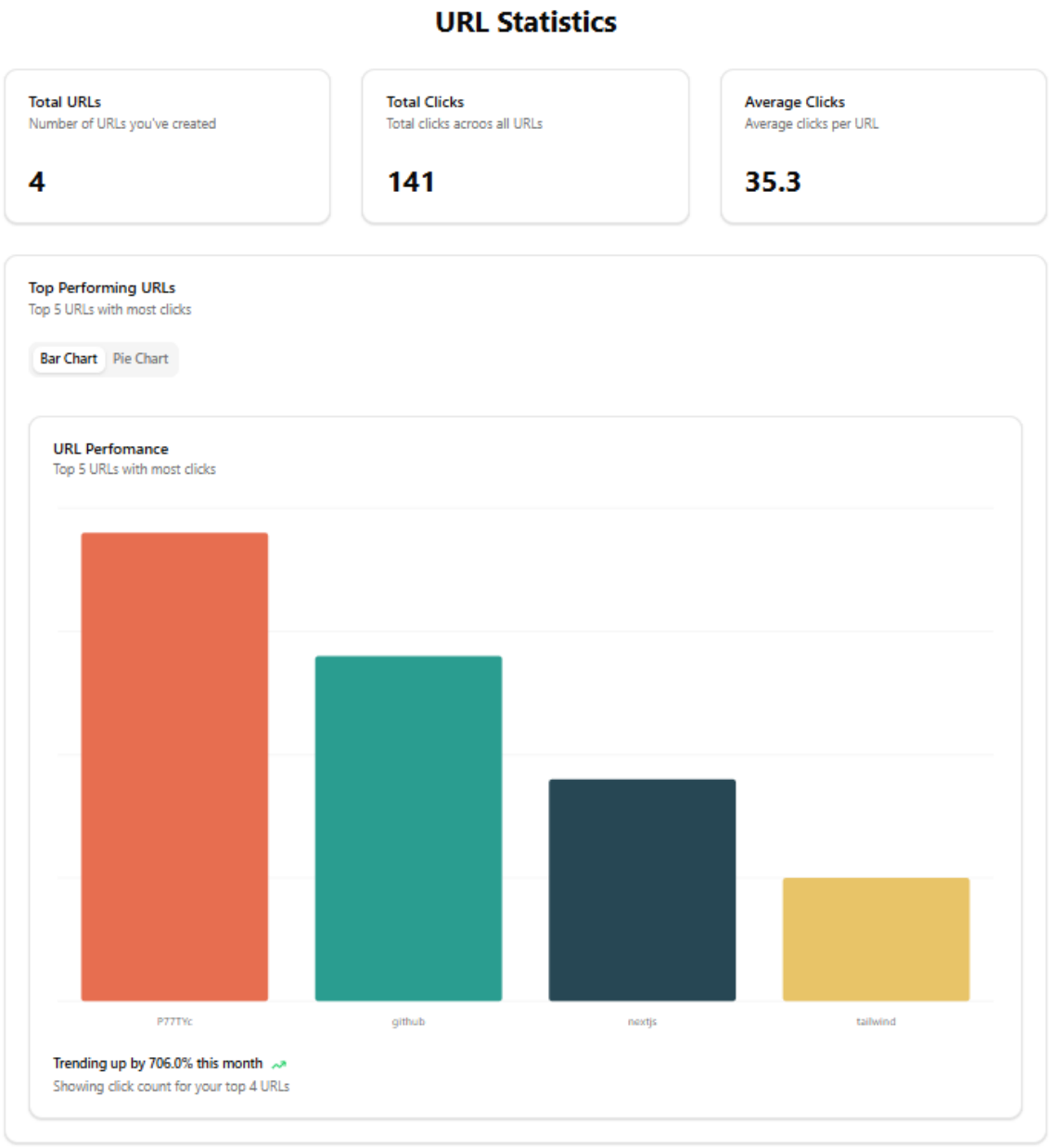


Figure 5.6.2 URL Stat Screen :-

This screen displays overall analytics, including total URLs, total clicks, and average clicks per URL. It also showcases the top-performing links through bar and pie charts, helping users quickly understand their URL performance trends.

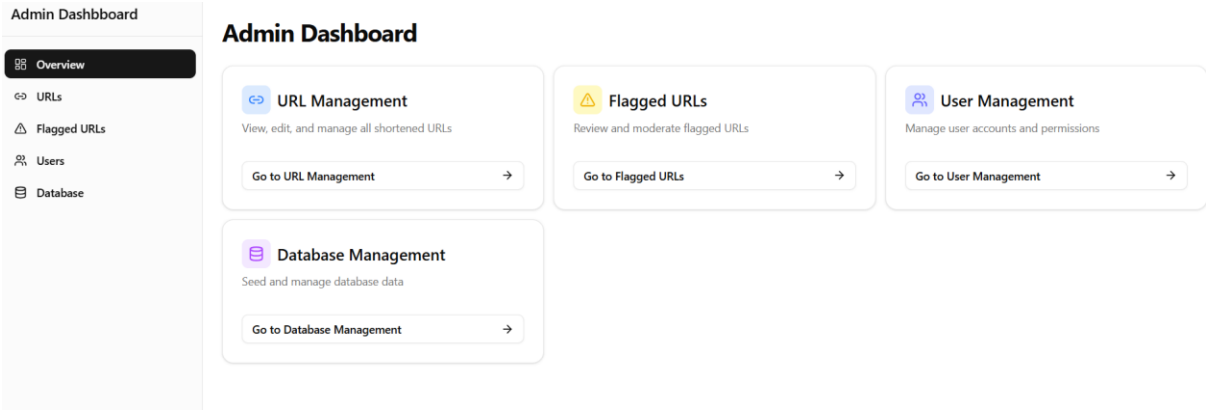


Figure 5.6.3 Admin Dashboard :-

The central hub for administrators to monitor and control platform operations. It provides quick access to URL management, flagged content review, user controls, and database maintenance, all from a single interface.

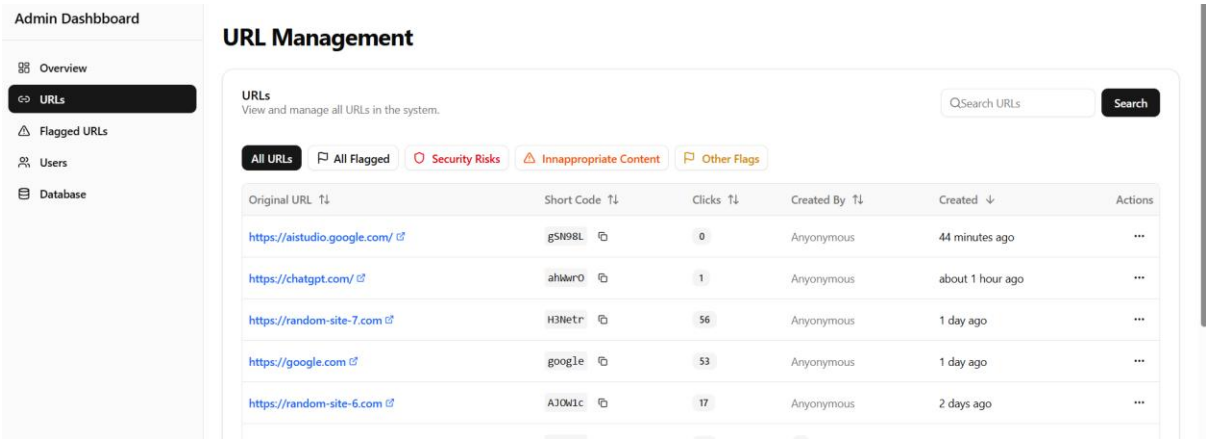


Figure 5.6.4 Admin URL Management :-

This section provides a detailed view and management controls for all shortened URLs in the system. Administrators can filter links by status such as "All Flagged," "Security Risks," or "Inappropriate Content," and review key metrics like clicks, creator, and creation date. Each entry includes actionable options for moderation and oversight.

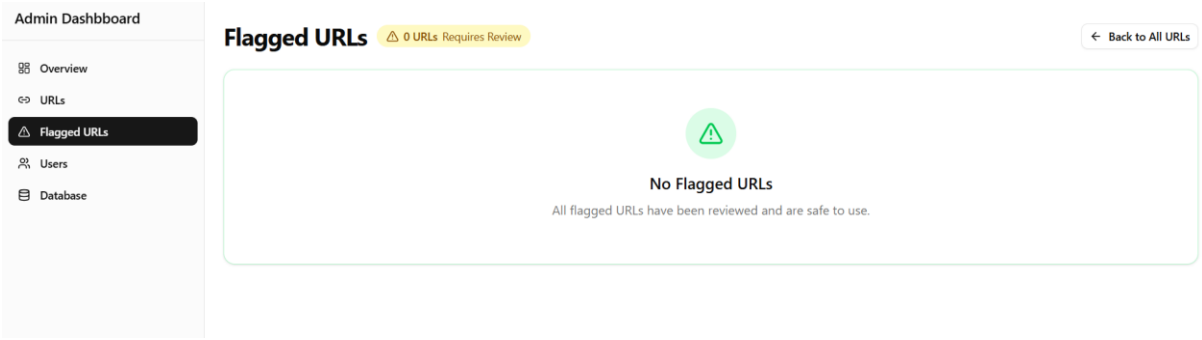


Figure 5.6.5 Admin Flagged URL Management :-

This screen displays all URLs that have been flagged for review due to security, content, or policy concerns. Currently, no URLs require attention, indicating all flagged links have already been moderated and resolved. Admins can quickly navigate back to view all URLs if needed.

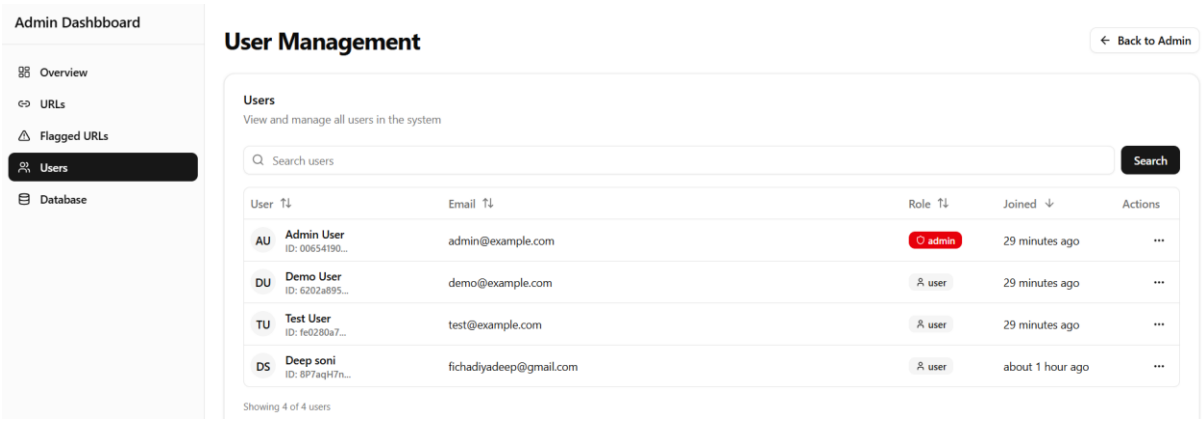


Figure 5.6.6 Admin Users Management :-

This panel allows administrators to view, search, and manage all user accounts within the system. Admins can assign or revoke roles—such as promoting a user to admin or demoting an admin to a regular user—to control platform access and permissions efficiently.

6. SYSTEM TESTING

This section covers the testing activities performed to ensure the URL Shortener SaaS platform functions correctly, securely, and efficiently. It includes the overall testing approach, implementation process, and methods used to validate different modules of the system.

6.1 Testing and Implementation

Testing was carried out throughout development to verify that each feature—such as URL shortening, authentication, analytics tracking, and admin controls—performed as expected. Both user-side and admin-side functionalities were implemented gradually and tested immediately to avoid integration issues. The system was then deployed in a test environment to confirm smooth operation across devices and browsers.

6.2 Testing Methodology

A combination of **unit testing**, **integration testing**, and **manual functional testing** was used.

Each module (authentication, URL creation, database operations, and admin actions) was tested individually and then validated together to ensure proper workflow. Edge cases such as invalid URLs, duplicate custom codes, and unauthorized access were also tested to ensure system reliability and security.

6.3 Test Case Design

No.	Test Scenario	Test Steps	Expected Result	Actual Result	Status
1	User Registration with Valid Data	1. Go to registration page 2. Enter name, email, password 3. Click Register	Account created, redirected to dashboard	Account created, redirected to dashboard	Pass
2	User Login with Valid Credentials	1. Go to login page 2. Enter valid email and password 3. Click Login	User logged in, session created	User logged in, session created	Pass
3	User Login with Invalid Password	1. Go to login page 2. Enter valid email but wrong password 3. Click Login	Login fails, error message shown	Login successful without validation	Fail
4	Create Short URL with Valid URL	1. Login 2. Enter original URL in form 3. Click Shorten button	Short URL created with unique code	Short URL created with unique code	Pass
5	Create Short URL with Custom Alias	1. Login 2. Enter URL and custom alias "mylink" 3. Click Shorten	Short URL created with custom alias	Short URL created with custom alias	Pass
6	Edit Existing Short URL	1. Login 2. Go to dashboard 3. Click Edit 4. Update URL 5. Save changes	URL updated successfully	URL updated successfully	Pass
7	Redirect Using Valid Short Code	1. Access short URL in browser 2. Verify redirection 3. Check click count	Redirected to original URL, clicks incremented	Redirected but clicks not incremented	Fail
8	View Analytics for URLs	1. Login 2. Go to dashboard 3. Click View Analytics 4. Check data	Analytics displayed with clicks, device, browser info	Analytics displayed with clicks, device, browser info	Pass
9	Detect Suspicious URL	1. Login 2. Enter suspicious URL 3. Click Shorten 4. Check for warning	Warning shown, URL flagged, admin notified	URL created without warning or flag	Fail
10	Admin Flag Unsafe UR	1. Admin login 2. Go to Manage URLs 3. Select URL 4. Flag with reason 5. Confirm	URL flagged, warning shown to user	URL flagged, warning shown to user	Pass

6.4 System Implementation Strategy

The implementation followed an incremental development model, allowing each module of the system to be built, tested, and improved step by step. Core backend functionalities such as database schema creation, server actions, and authentication workflows were implemented first to establish a solid foundation. Once the backend structure was stable, frontend components were developed to interact seamlessly with the underlying logic through API routes and server actions.

To maintain consistency and reduce integration issues, the system used a unified technology stack (Next.js, Drizzle ORM, PostgreSQL), ensuring both the client and server layers operate within the same framework. This improved communication between components and allowed faster implementation of features like URL creation, analytics tracking, and role-based admin operations. Each feature was deployed to a staging environment for real-time validation before merging into the main system.

Additionally, security and scalability were prioritized during implementation. Input validation, error handling, protected routes, and role-based access control were integrated early to prevent vulnerabilities during later stages of development. Once all modules were functional, the system underwent full integration testing, UI refinement, and optimization before being deployed to the production environment. This structured implementation strategy ensured a reliable, maintainable, and scalable SaaS application.

7 CONCLUSION

This project successfully developed **Shortlink**, a fully functional and secure **URL Shortener SaaS Platform** designed to simplify and safeguard the process of link sharing in today's digital landscape. By integrating modern web development tools and methodologies, the system provides users with an intuitive interface to create, manage, and track shortened URLs, while offering administrators powerful tools for content moderation, user management, and system oversight. The platform stands as a robust, scalable, and production-ready solution that addresses both usability needs and security concerns inherent in URL shortening services.

Built on the **Next.js 14 framework** using **TypeScript**, **Tailwind CSS**, and **ShadCN UI**, the frontend delivers a responsive and accessible user experience. On the backend, **Next.js Server Actions**, **Drizzle ORM**, and **PostgreSQL** ensure efficient data handling, secure transactions, and scalable performance. The inclusion of **real-time safety scanning**, **role-based access control (RBAC)**, **JWT authentication**, and **analytics tracking** highlights the project's emphasis on security, transparency, and user trust—features often lacking in conventional URL shorteners.

Through systematic implementation of modular components such as **Authentication**, **URL Management**, **Admin Panel**, and **Analytics Tracking**, this project not only meets academic and functional requirements but also aligns with industry standards. The completed system demonstrates proficiency in full-stack development, database design, API integration, and UI/UX best practices. Future enhancements may include expanded API support, browser extensions, machine learning-based threat detection, and multi-region deployment—each of which would further elevate the platform's utility and reach.

In conclusion, **Shortlink** exemplifies how contemporary web technologies can be harnessed to create secure, user-centric, and administratively controlled digital tools. It reinforces the critical role of safety and usability in link management systems and serves as a foundational model for future innovations in the domain of URL shortening and web resource sharing.

8. LEARNING DURING PROJECT WORK

The development of the Shortlink URL shortener platform provided significant practical learning across multiple domains of modern web development. Through hands-on implementation, several technical and conceptual insights were gained:

Technical Stack Mastery
Working with Next.js 14 (App Router) and Server Actions provided a deep understanding of full-stack React development, including server-side rendering, static generation, and seamless backend integration within the frontend framework. Using TypeScript throughout the project improved code reliability, type safety, and maintainability. The integration of Drizzle ORM with PostgreSQL offered valuable experience in database modeling, schema migrations, and performance optimization in a TypeScript-native environment.

Security and Authentication Implementation
Implementing NextAuth.js with role-based access control (RBAC) provided practical exposure to secure authentication flows, session management, and authorization patterns. Developing the URL safety scanning module highlighted the importance of proactive security measures in user-generated content platforms, including validation, sanitization, and moderation workflows.

System Architecture and State Management
The project reinforced principles of modular architecture, separation of concerns, and scalable design. Experience was gained in structuring a monorepo-like application with clearly divided modules (authentication, URL management, admin panel, analytics) that communicate efficiently. Managing application state across server and client components in Next.js 14 provided insight into modern React state patterns without relying heavily on external libraries.

Project Management and Collaboration
Beyond coding, the project underscored the importance of systematic planning, version control with Git, consistent documentation, and iterative development. Writing clean, maintainable code and establishing clear conventions for folder structure, naming, and imports became essential practices for project scalability and team collaboration readiness.

Problem-Solving and Debugging
Challenges such as handling redirects safely, managing database transactions, optimizing query performance, and implementing real-time analytics fostered stronger debugging and systematic problem-solving skills. Learning to read logs, trace errors in full-stack environments, and write comprehensive tests contributed to a more resilient development mindset.

Overall, this project served as a comprehensive learning platform that bridged theoretical knowledge with real-world application, preparing for professional software development environments where performance, security, usability, and maintainability are equally critical.

9 FUTURE ENHANCEMENTS

While the Shortlink platform is fully functional and production-ready, several future enhancements could significantly expand its capabilities and user value. These improvements focus on feature enrichment, technical optimization, and enterprise scalability.

Feature Expansion & User Experience
Future versions could include custom domain support, allowing users to brand their short links, along with QR code generation and bulk URL shortening via CSV uploads. An official public API would enable developers to integrate Shortlink into third-party applications, while browser extensions could provide one-click shortening directly from the browser toolbar. Advanced analytics with interactive charts and geographic heatmaps would offer deeper insights into link performance.

Technical & Performance Improvements
To enhance performance and security, integration with Redis for caching frequently accessed redirects and implementing edge computing via Vercel Edge Functions or Cloudflare Workers would reduce latency globally. Machine learning-based safety detection could replace or supplement the current rule-based system for more accurate phishing and malware identification. Real-time click tracking
using WebSockets would provide instant analytics updates without page refreshes.
Scalability & Enterprise Features

For larger-scale deployments, migrating to a microservices architecture would improve maintainability and scalability. Support for Single Sign-On (SSO) via providers like Okta or Azure AD would cater to organizational users, while team collaboration features with role-based permissions would make the platform suitable for business use. Comprehensive audit logging and compliance reporting tools would help meet regulatory requirements such as GDPR and CCPA.
Mobile & Integration Ecosystem

A dedicated mobile application for iOS and Android could extend accessibility, while webhook support would allow notifications for events like clicks, flags, or expirations. Integration with marketing tools, CRM platforms, and social media schedulers would position Shortlink as a central tool in digital marketing and communication workflows.

Each of these enhancements would build upon the current solid foundation, transforming Shortlink from a capable URL shortener into a comprehensive link management platform suitable for individual, business, and enterprise use cases.

10. BIBLIOGRAPHY

- Next.js Documentation – Vercel, 2024
- PostgreSQL Official Docs – PostgreSQL Group, 2024
- React: The Complete Reference – Banks & Porcello, 2023
- TypeScript Handbook – Microsoft, 2024
- Drizzle ORM Documentation – 2024
- Clean Architecture – Robert C. Martin, 2017
- Web Security for Developers – Malcolm McDonald, 2022

10.2 ONLINE REFERENCES

- Next.js App Router – https://nextjs.org/docs/app
- Drizzle ORM – https://orm.drizzle.team
- NextAuth.js – https://next-auth.js.org
- Tailwind CSS – https://tailwindcss.com/docs
- ShadCN UI – https://ui.shadcn.com
- TypeScript Docs – https://www.typescriptlang.org/docs
- MDN Web Docs – https://developer.mozilla.org
- OWASP Security – https://owasp.org

10.3 OFFLINE REFERENCES

- University SE & DBMS Course Material
- IEEE/ACM Papers on URL Shortening & Security
- Project Guidance – Faculty Notes
- Previous FYPs – Library Archives
- Printed PostgreSQL & Next.js Guides