**SUMMER TRAINING/INTERNSHIP**

**PROJECT REPORT**

(Term June-July 2025)

**URL SHORTENER SYSTEM**

Submitted by:

**Rakshanda Talwekar**

**Registration Number: 12309743**

**Vani Bhardwaj**

**Registration Number: 12308150**

**Anupama Dey**

**Registration Number: 12306578**

**Mohit Agrawal**

**Registration Number: 12319895**

**Agarveer Sandhu**

**Registration Number: 12320018**

**Course Code: PETV51**

Under the Guidance of:

**Dr. Chirag Sharma (Associate Professor, Lovely Professional University)**

**Er. Nitish Kumar  
(Assistant Professor, Lovely Professional University)**

**School of Computer Science and Engineering**

**CERTIFICATE**This is to certify that the project report entitled "URL Shortener Web Application" is a bonafide work carried out by Rakshanda Talwekar (12309743), Vani Bhardwaj (12308150), Anupama Dey (12306578), Mohit Agrawal (12319895) and Agarveer Sandhu (12320018), student of B. Tech CSE, School of Computer Science and Engineering, during the summer internship term June–July 2025, under my supervision.

Dr. Chirag Sharma and Er. Nitish Kumar  
(Mentor)

i

**ACKNOWLEDGEMENT**We take this opportunity to express our heartfelt gratitude to our mentor Dr. Chirag Sharma and Er. Nitish Kumar for their invaluable guidance and support throughout the training period. We are also thankful to the faculty and staff of the School of Computer Science and Engineering for providing the infrastructure and a conducive learning environment. Lastly, we would like to thank our teammates for their cooperation and collective efforts in successfully completing this summer training project.

ii

|  |
| --- |
| CHAPTER 1: INTRODUCTION - Company Profile - Overview of Training Domain - Objective of the Project |
| CHAPTER 2: TRAINING OVERVIEW - Tools & Technologies Used - Areas Covered During Training - Daily/Weekly Work Summary |
| CHAPTER 3: PROJECT DETAILS - Title of the Project - Problem Definition - Scope and Objectives - System Requirements - Architecture Diagram - Data Flow / UML Diagrams |
| CHAPTER 4: IMPLEMENTATION - Tools Used - Methodology - Modules / Screenshots - Code Snippets |
| CHAPTER 5: RESULTS AND DISCUSSION - Output / Report - Challenges Faced - Learnings |
| CHAPTER 6: CONCLUSION - Summary |

**TABLE OF CONTENT**

iii

**CHAPTER 1: INTRODUCTION**

**Company Profile:**

This project was independently executed as part of the LPU Summer Training Program under the guidance of the School of Computer Science and Engineering. LPU promotes hands-on learning and emphasizes real-world project development through mentorship and structured training. This project focuses on building a URL Shortener Web Application that demonstrates key concepts in backend development, data storage, and user interaction.

**Overview of Training Domain:**

The training focused on Full Stack Web Development with a primary emphasis on backend technologies. Students learned Python programming and built web applications using the Flask micro-framework. Training sessions covered HTTP protocols, RESTful APIs, and rendering templates with HTML. Students also explored basic frontend styling and interactivity using CSS and JavaScript. By the end of the training, students were capable of deploying complete web applications and integrating backend services with frontend interfaces.

**Objective of the Project:**

The objective of this project is to build a working URL Shortener that reduces long, complex URLs into shorter aliases. Users can either generate a random alias or define a custom alias for their URL. The application must ensure alias uniqueness, persistently store mappings, and perform accurate redirections when a short link is accessed.

**CHAPTER 2: TRAINING OVERVIEW**

**Tools & Technologies Used:** - Python   
- Flask:   
- HTML/CSS  
- JavaScript  
- JSON  
- Visual Studio Code  
- Postman  
- Git

**Areas Covered During Training:** - Basics of Python and Flask for backend development.  
- Setting up and routing in a Flask web server.  
- Rendering HTML pages using Jinja templates.  
- Creating and consuming RESTful APIs.  
- Handling form inputs and integrating frontend with backend.  
- File handling in Python and managing persistent storage using JSON.  
- Testing APIs with tools like Postman.  
- Basic frontend styling and responsiveness using HTML and CSS.

**Daily/Weekly Work Summary:** Week 1:  
- Introduction to Flask and Python basics.  
- Setting up a basic Flask server with routes.  
Week 2:  
- Designing REST API endpoints.  
- Handling POST requests and processing form inputs.

2

Week 3:  
- Implementing alias generation logic.  
- Creating a persistent JSON file to store URLs.  
- Error handling and uniqueness validation for custom aliases.  
Week 4:  
- Building and connecting the HTML frontend.  
- Integrating JavaScript with Flask routes.  
- Full feature testing and resolving bugs.  
Week 5:  
- Final UI refinements and input validations.  
- Conducting test cases using Postman.  
- Project documentation and report compilation.

3

**CHAPTER 3: PROJECT DETAILS**

**Title of the Project:** URL Shortener Web Application

**Problem Definition:** Long URLs can be difficult to share, remember, or print, especially when they include complex query parameters. This creates usability challenges for users, particularly when accessing resources across platforms. URL shorteners address this issue by transforming long URLs into compact, human-friendly versions that redirect to the original address. This project aims to implement such a system with the added ability for users to specify a custom alias for the shortened URL.

**Scope and Objectives:** The scope of this project includes the following objectives:  
- Develop a backend system to generate and store short URLs.  
- Accept and validate custom aliases to avoid duplication.  
- Use a JSON file as a persistent, lightweight storage solution.  
- Design a simple HTML frontend for user interaction.  
- Enable redirection to the original URL when a short alias is accessed.  
- Ensure the solution works on localhost and is testable with API tools.

**System Requirements:** Software Requirements:  
- Python 3.11 or higher  
- Flask  
- Web browser (Chrome, Firefox)  
- Visual Studio Code (IDE)  
- Postman (for API testing)  
Hardware Requirements:  
- A system with at least 4 GB RAM

4

- Localhost environment (no internet needed for local deployment)

**Architecture Diagram:**   
User → Web Interface → Flask Backend → JSON Storage  
↓  
Redirection on short URL access

**Data Flow / UML Diagrams:** 1 User enters long URL + optional alias on frontend.

2. Data is sent via POST to Flask backend.

3. Flask checks for alias uniqueness and saves data to JSON.

4. Flask returns short URL to frontend.

5. On accessing short URL, Flask reads JSON and redirects to long URL.

5

**CHAPTER 4: IMPLEMENTATION**

**Tools Used:**

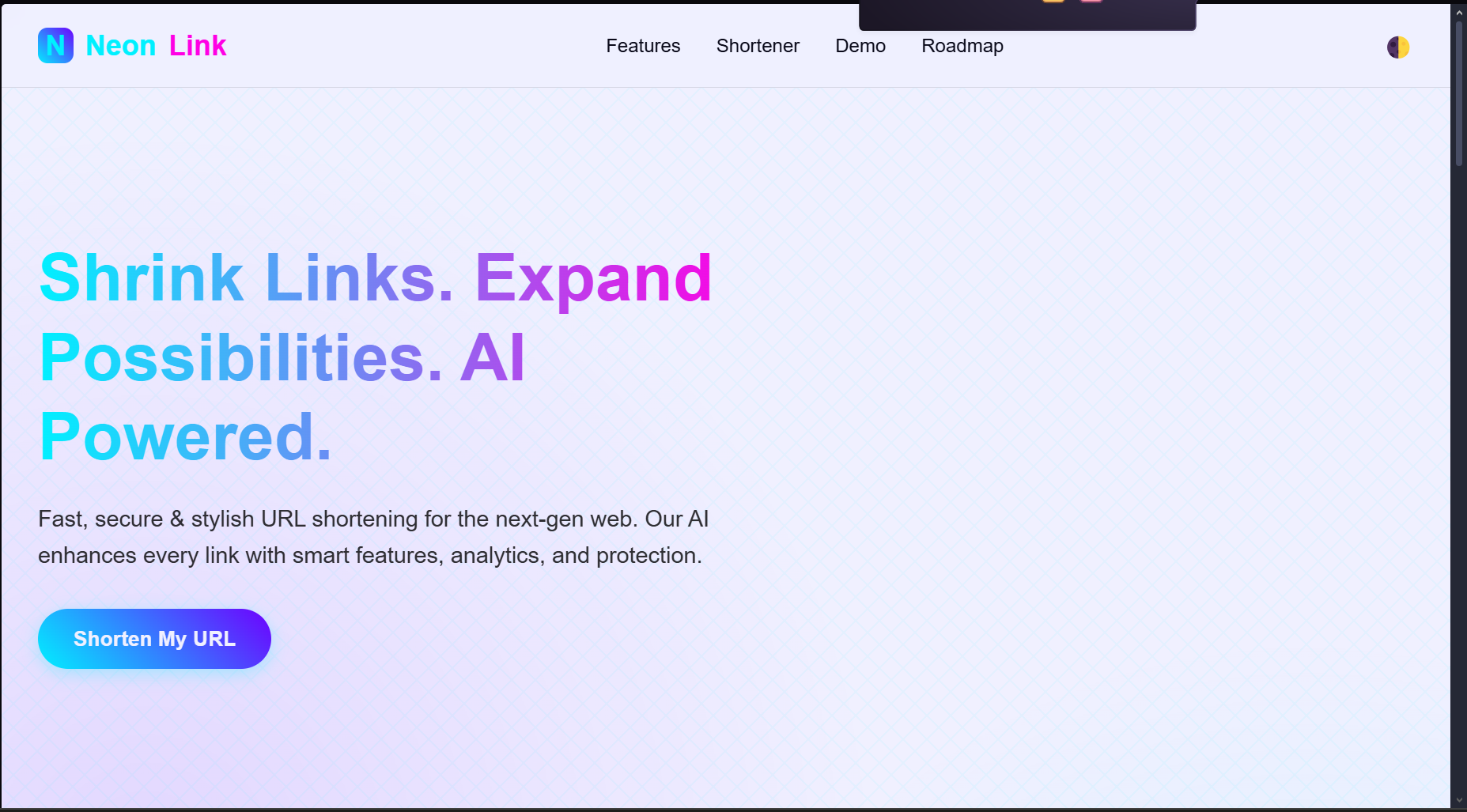
- Python 3.11  
- Flask framework  
- HTML/CSS/JavaScript  
- JSON for data storage  
- Postman  
- Visual Studio Code

**Methodology:**

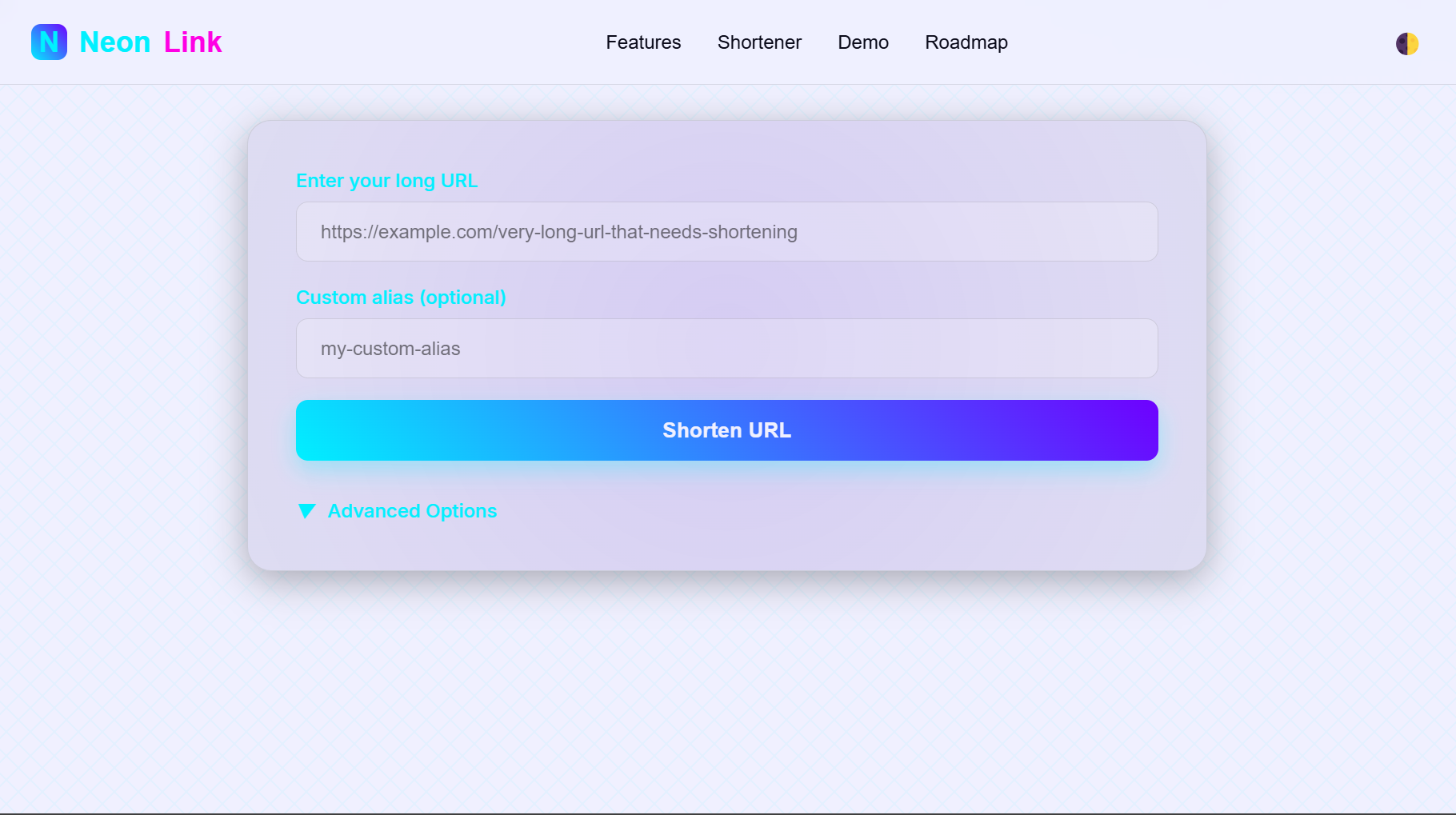
The application follows a RESTful architecture using Flask for backend services. User inputs are handled using a simple HTML form that interacts with the backend via POST requests. The system checks for duplicate aliases, stores mappings in a JSON file, and responds with a complete short URL. When a short URL is visited, Flask retrieves the corresponding long URL from the JSON store and redirects the user.

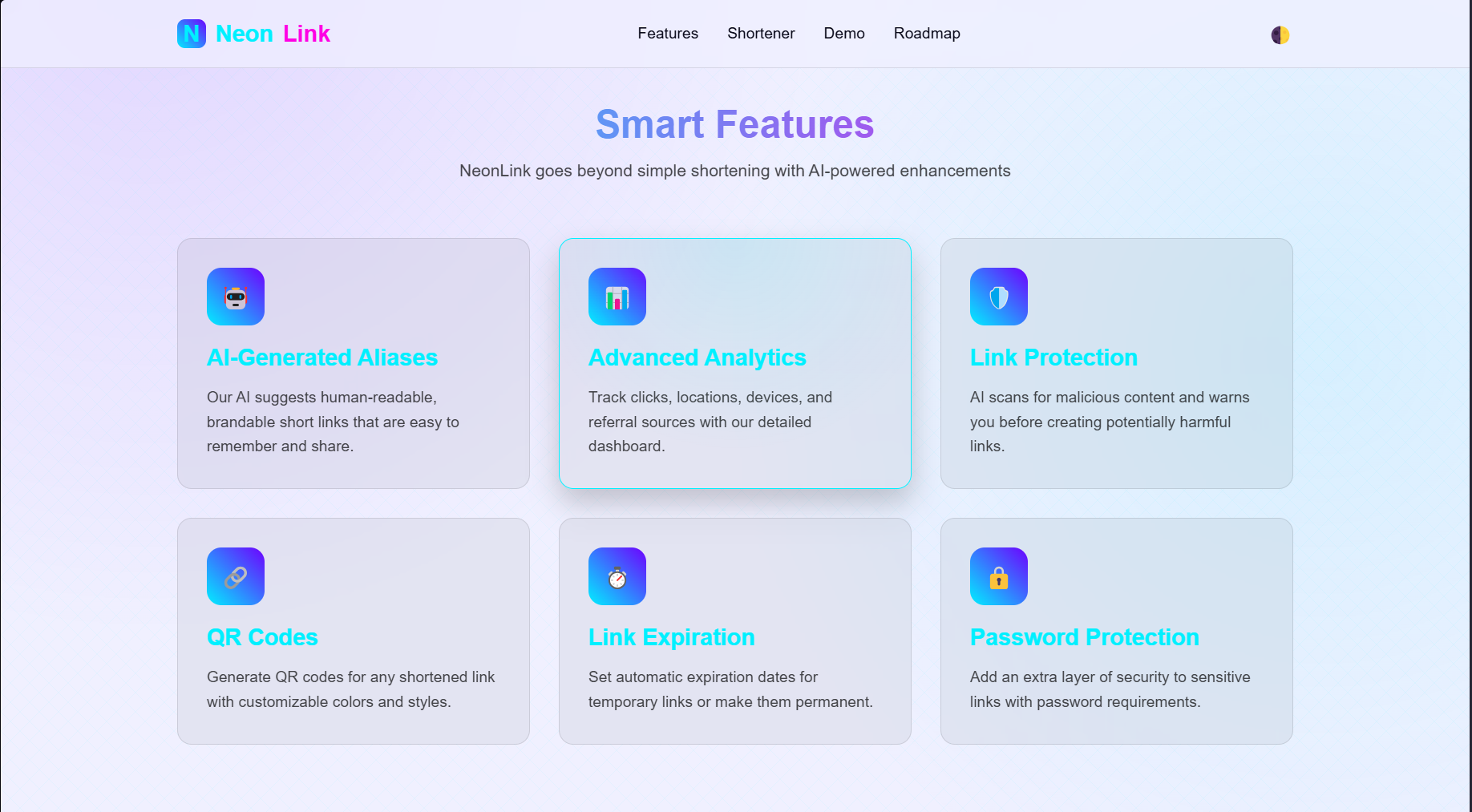
**Modules / Screenshots:**

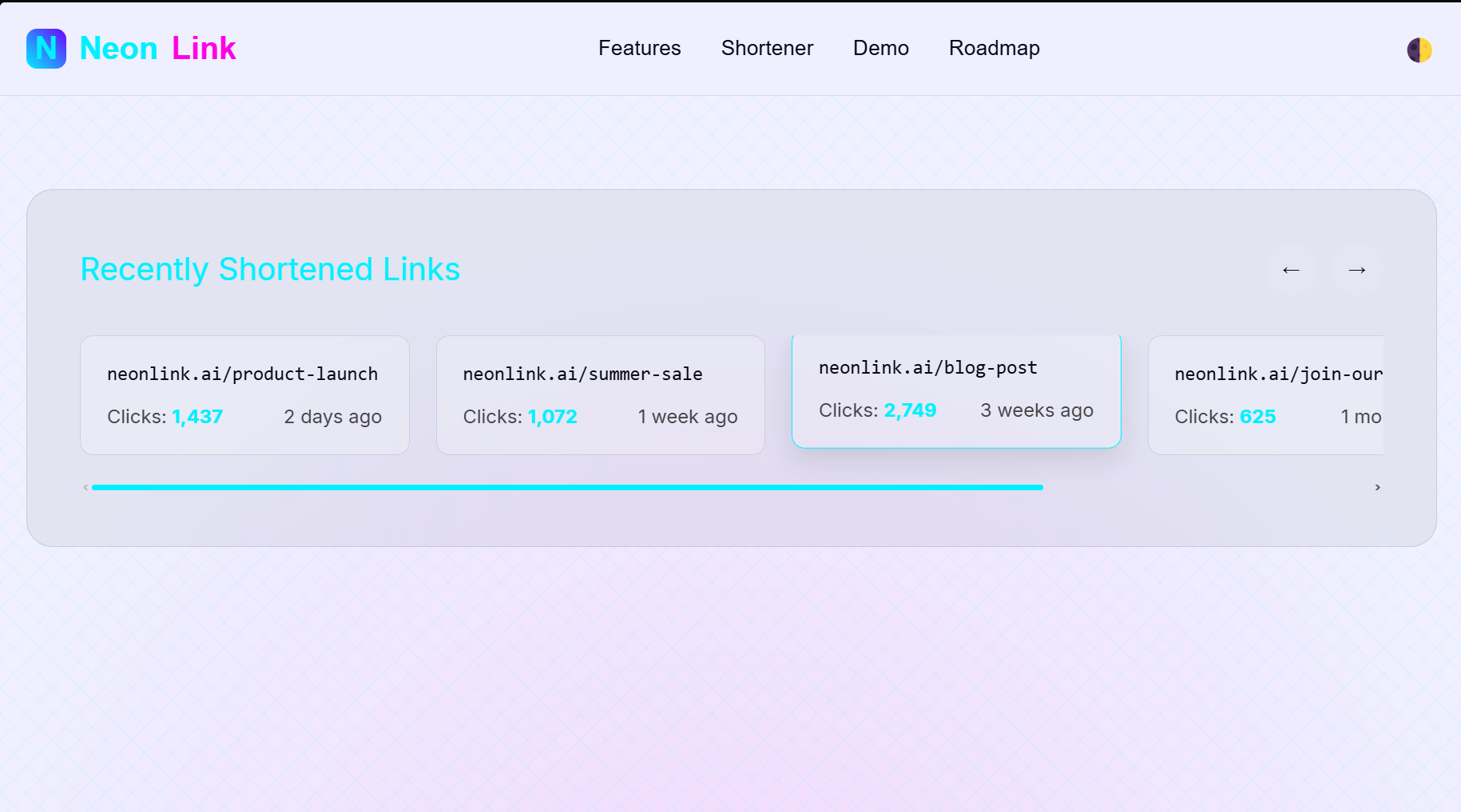
1. Home Page: HTML form for inputting long URLs and aliases.  
2. Shortening Endpoint: Flask POST route that processes form data.  
3. JSON Storage File: Maintains alias-to-URL mappings persistently.  
4. Redirect Route: Dynamically redirects based on alias access.

 Figure 1: Home page of the website

6

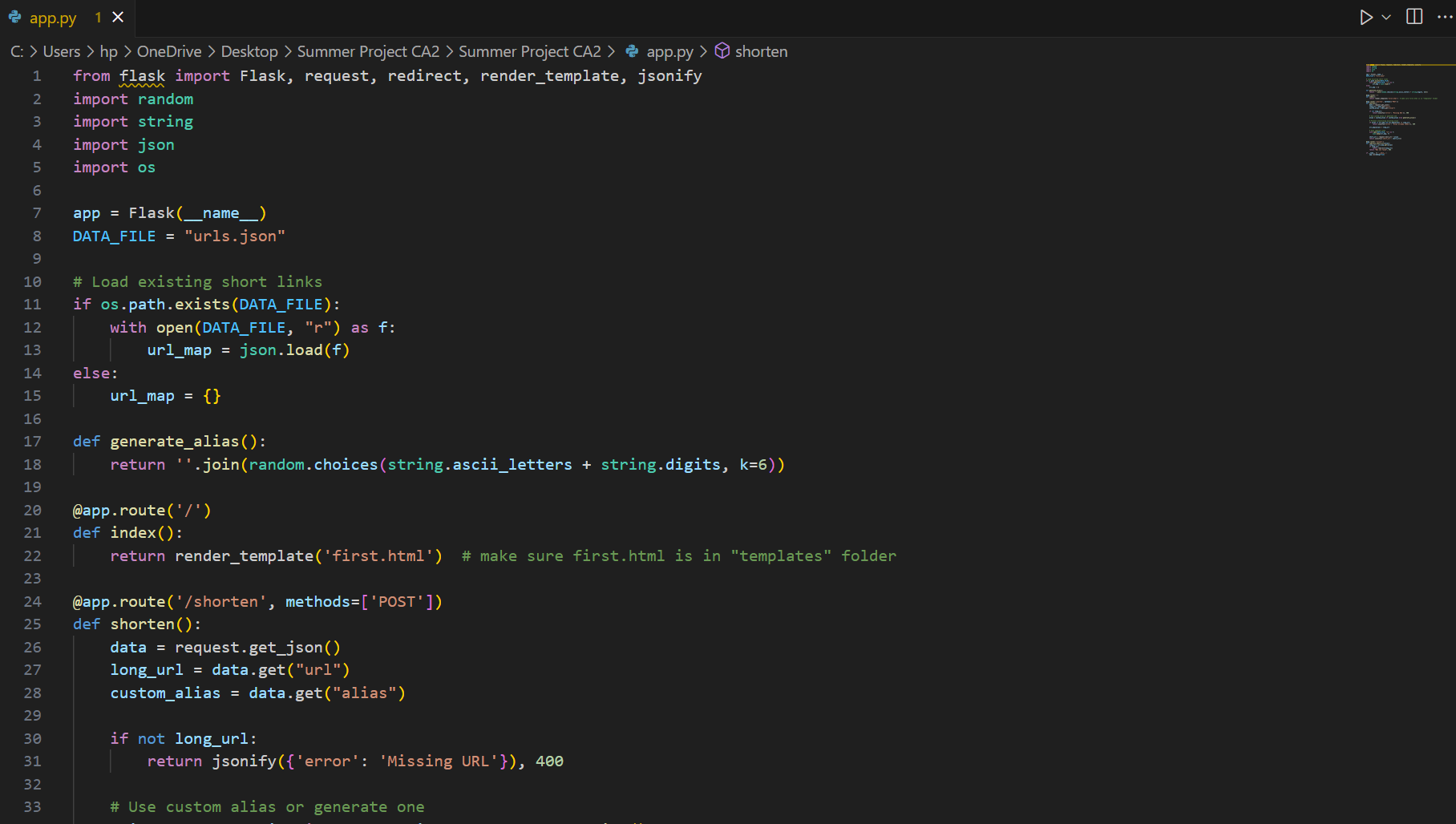
 Figure 2: Main page of the website

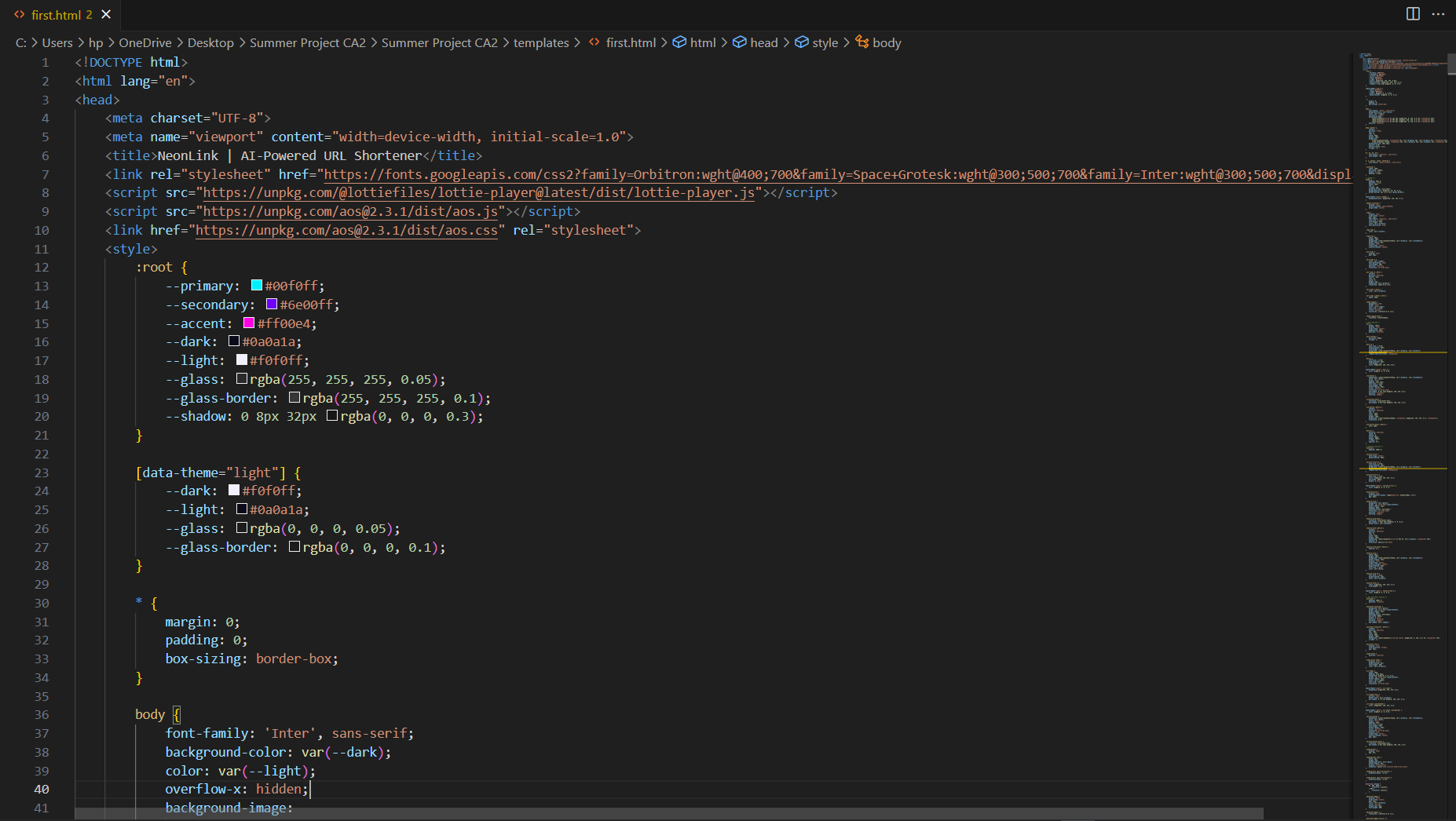
 Figure 3: Smart features of the website

 Figure 4: Link shortening demo

7

**Code Snippets:**

 Figure 5: Python Code

 Figure 6: HTML, CSS and Javascript Code

8

**CHAPTER 5: RESULTS AND DISCUSSION**

**Output / Report:**

- The application successfully shortens long URLs and optionally uses custom aliases.  
- All alias-URL pairs are stored persistently in a JSON file.  
- Visiting a valid short alias redirects the user to the original URL.  
- Error handling is implemented for missing inputs and duplicate aliases.

**Challenges Faced:**

- Handling file write/read operations concurrently.  
- Ensuring uniqueness of randomly generated aliases.  
- Testing redirection logic and confirming proper data storage.  
- Managing HTML form data and converting to JSON for backend use.

**Learnings:**

- Hands-on understanding of Flask routing and JSON handling.  
- Developed REST APIs and understood HTTP request/response cycles.  
- Integrated frontend and backend systems.  
- Debugged and tested dynamic redirection logic.  
- Practiced modular coding and error handling strategies.

9

**CHAPTER 6: CONCLUSION**

This project was a comprehensive learning experience, bridging core programming principles with real-world application design. By building a URL shortener, I learned how to manage backend services, handle form data, generate dynamic links, and persist data effectively using JSON. The implementation demonstrated my ability to integrate Flask with HTML and test the system using API tools. Challenges like alias collisions, file handling, and redirect accuracy were addressed through research, debugging, and testing.  
  
The URL Shortener Web Application can serve as a base for more advanced systems incorporating analytics, user authentication, and database storage. Overall, the internship strengthened my foundational and applied skills in web development, preparing me for larger and more complex software engineering tasks.

10