Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

This document outlines the requirements for a web-based application that shortens URLs. The goal is to provide users with a simple way to convert lengthy web addresses into compact links, with options for setting expiration times and custom codes.

1.2 Scope

The project includes a user interface built with React and a backend service. Users can enter several URLs, define how long each link should remain active, and optionally choose a custom code for their short link. The backend processes these requests and returns the shortened links.

1.3 Key Terms

- URL Shortener: A tool that creates a brief link pointing to a longer web address.
- Shortcode: A user-defined or system-generated identifier for the short link.
- Validity: The duration (in minutes) that a short link remains usable.

2. System Overview

2.1 Context

This is a standalone web application. The frontend uses React and Material UI for the interface, and Axios for communicating with the backend API.

2.2 Main Features

- Users can input multiple URLs at once (up to five).
- Each URL can have a custom expiration time and shortcode.
- The app sends the data to the backend, which returns the shortened links.
- Shortened links are displayed to the user.
- The app provides feedback for errors or invalid input.

2.3 Intended Users

- Anyone who needs to generate short, shareable links from long URLs.

2.4 Supported Platforms

- Modern web browsers (e.g., Chrome, Firefox, Edge)
- Node.js and npm for development
- Backend server for API processing

2.5 Technical Constraints

- The frontend is built with React, Material UI, and Axios.
- The backend must expose a `/shorten` endpoint for POST requests.

2.6 User Help

- On-screen instructions and prompts are provided within the app.

3. Requirements

3.1 Functional Requirements

- The app allows up to five URLs to be entered at once.
- Each URL is checked to ensure it is valid.
- Users can set how long each short link should work (in minutes).
- Users can request a specific shortcode for each link.
- The app sends all entered data to the backend in a single request.
- The app shows the resulting short links after processing.
- The app displays clear error messages for invalid input or server issues.

3.2 Non-Functional Requirements

- The app should respond to user actions within two seconds.
- The app must work on all major browsers.
- The interface should be responsive and accessible.

4. Interfaces

4.1 User Interface

- Form fields for entering URLs, validity, and shortcode
- Button to add more URL input fields (up to five)
- Button to submit the form
- Section to display the generated short links and any error messages

4.2 Hardware Interface

- No special hardware required; runs in a web browser

4.3 Software Interface

- Backend API endpoint: `POST /shorten`
- Request format: `{ urls: [{ longUrl, validity, shortcode }] }`
- Response format: `[ { shortUrl } ]`

4.4 Communication

- Uses HTTP or HTTPS for all client-server interactions

5. Additional Notes

- Both frontend and backend should be runnable on a local machine for development.
- The frontend should handle cases where the backend is unavailable and inform the user appropriately.