Web Development Project

Task 1: Link Shortener

In this project, you will be required to make an API to build short URLs. The functionality will be similar to bitly. Using Node, Express, and MongoDB you can make your own URL Shortener service. However, you can use any backend language also depending on your expertise. The project is not specific to any particular backend language.

Skills Required – Node, MongoDB, JavaScript

Creating a URL shortener service involves building an API to handle shortening long URLs and redirecting users to the original URLs when they access the shortened version. Below is a step-by-step guide using Node.js, Express, and MongoDB to create a basic URL shortener service.

Step 1: Setting Up Your Project

1. **Initialize Node.js Project:** Create a new directory for your project and run **npm init -y** to initialize a new Node.js project.
2. **Install Dependencies:** Install necessary packages such as Express, Mongoose (for MongoDB), and shortid (for generating short IDs) using npm:

npm install express mongoose shorted

3. **Set Up MongoDB:** Make sure you have MongoDB installed and running on your machine or use a cloud MongoDB service like MongoDB Atlas.

Step 2: Create Your Express App

Create an **index.js** file to set up your Express application.

```
// index.js

const express = require('express');

const mongoose = require('mongoose');

const shortid = require('shortid');

const app = express();

app.use(express.json());

// Connect to MongoDB

mongoose.connect('mongodb://localhost/urlShortener', {

  useNewUrlParser: true,

  useUnifiedTopology: true,

});

const Url = mongoose.model('Url', new mongoose.Schema({

  _id: { type: String, default: shortid.generate },
```

```
  originalUrl: String,
}));
// Endpoint to shorten URLs
app.post('/shorten', async (req, res) => {
  const { originalUrl } = req.body;
  try {
    let url = await Url.findOne({ originalUrl });
    if (url) {
      res.json(url);
    } else {
      const newUrl = new Url({ originalUrl });
      url = await newUrl.save();
      res.json(url);
    }
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Server error' });
  }
});
// Redirect shortened URLs
app.get('/:shortUrl', async (req, res) => {
  const { shortUrl } = req.params;
  try {
    const url = await Url.findById(shortUrl);
    if (url) {
      res.redirect(url.originalUrl);
    } else {
      res.status(404).json({ error: 'URL not found' });
    }
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Server error' });
```

```
  }
});
const PORT = 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Step 3: Run Your Server

Start Node.js server by running:

```
node index.js
```

Step 4: Test Your API

Use tools like Postman or curl to test API endpoints:

1. **Shorten URL:** Send a POST request to **http://localhost:5000/shorten** with JSON body:

```
{ "originalUrl": "https://www.example.com/very/long/url/to/be/shortened" }
```

1. This will return a JSON response containing the shortened URL.
2. **Access Shortened URL:** Access **http://localhost:5000/{shortUrl}** in browser, where **{shortUrl}** is the short ID generated by service. This should redirect to the original long URL.