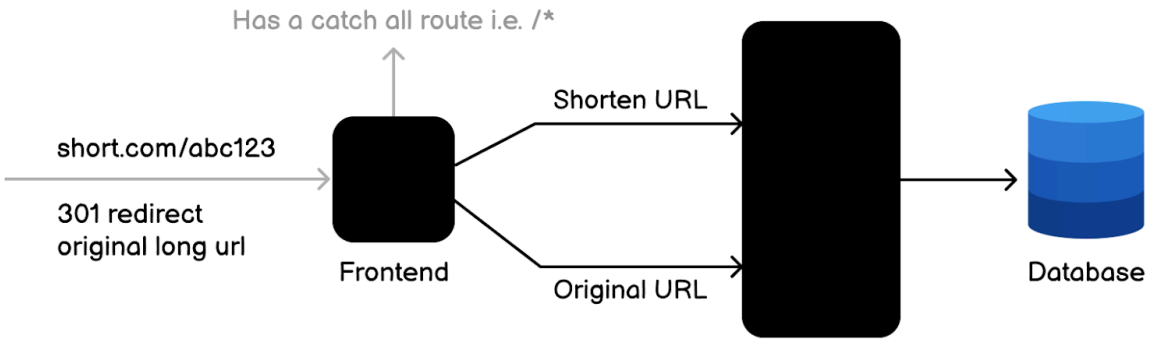
innovaxel
23 CCA
DHA Phase 8 (Ex-Park View)
Lahore, PK

Take-Home Assignment

URL Shortening Service

1. Overview

You must create a simple **RESTful API** that allows users to shorten long URLs. The API should provide endpoints to create, retrieve, update, and delete short URLs. It should also provide statistics on the number of times a short URL has been accessed.



2. Requirements

You should create a RESTful API for a URL shortening service. The API should allow users to perform the following operations:

- **Create** a new short URL
- **Retrieve** an original URL from a short URL
- **Update** an existing short URL
- **Delete** an existing short URL
- **Get** statistics on the short URL (e.g., number of times accessed)

You can optionally set up a minimal frontend to interact with the API and set up redirects for the short URLs to the original URLs.

3. API Endpoints

The details for each API operation are given below:

3.1. Create Short URL

Create a new short URL using the **POST** method

```
POST /shorten
{
  "url": "https://www.example.com/some/long/url"
}
```

The endpoint should validate the request body and return a **201 Created** status code with the newly created short URL, i.e.

```
{
  "id": "1",
  "url": "https://www.example.com/some/long/url",
  "shortCode": "abc123",
  "createdAt": "2021-09-01T12:00:00Z",
  "updatedAt": "2021-09-01T12:00:00Z"
}
```

or a **400 Bad Request** status code with error messages in case of validation errors. Short codes must be unique and should be generated randomly.

3.2. Retrieve Original URL

Retrieve the original URL from a short URL using the **GET** method

```
GET /shorten/abc123
```

The endpoint should return a **200 OK** status code with the original URL i.e.

```
{
  "id": "1",
  "url": "https://www.example.com/some/long/url",
  "shortCode": "abc123",
  "createdAt": "2021-09-01T12:00:00Z",
  "updatedAt": "2021-09-01T12:00:00Z"
}
```

or a **404 Not Found** status code if the short URL was not found. Your frontend should be responsible for retrieving the original URL using the short URL and redirecting the user to the original URL.

3.3. Update Short URL

Update an existing short URL using the **PUT** method

```
PUT /shorten/abc123
{
  "url": "https://www.example.com/some/updated/url"
}
```

The endpoint should validate the request body and return a 200 OK status code with the updated short URL, i.e.

```
{
  "id": "1",
  "url": "https://www.example.com/some/updated/url",
  "shortCode": "abc123",
  "createdAt": "2021-09-01T12:00:00Z",
  "updatedAt": "2021-09-01T12:30:00Z"
}
```

or a 400 Bad Request status code with error messages in case of validation errors. It should return a 404 Not Found status code if the short URL was not found.

3.4. Delete Short URL

Delete an existing short URL using the **DELETE** method

```
DELETE /shorten/abc123
```

The endpoint should return a **204 No Content** status code if the short URL was successfully deleted or a **404 Not Found** status code if the short URL was not found.

3.5. Get URL Statistics

Get statistics for a short URL using the **GET** method

The endpoint should return a **200 OK** status code with the statistics, i.e.

```
{
  "id": "1",
  "url": "https://www.example.com/some/long/url",
  "shortCode": "abc123",
  "createdAt": "2021-09-01T12:00:00Z",
  "updatedAt": "2021-09-01T12:00:00Z",
  "accessCount": 10
}
```

or a **404 Not Found** status code if the short URL was not found.

4. Tech Stack

Feel free to use any programming language, framework, and database of your choice for this project. Here are some suggestions:

- If you are using **JavaScript**, you can use **Node.js** with **Express.js**
- If you are using **Python**, you can use **Flask** or **Django**
- If you are using **Java**, you can use **Spring Boot**
- If you are using **Ruby**, you can use **Ruby on Rails**
- If you are using **PHP**, you can use **Laravel** or any other relevant framework

For databases, you can use:

- **MySQL** if you are using **SQL**
- **MongoDB** if you are using **NoSQL**

Your job is to implement the core functionality of the API, focusing on creating, retrieving, updating, and deleting short URLs, as well as tracking and retrieving access statistics. You don't have to implement authentication or authorization for this project.

5. Submission Guidelines

- **Deadline:** Please ensure compliance with all submission guidelines and complete the assessment by the specified deadline mentioned in the email.
- **Repository:**
    - Create a GitHub repository named firstname-innovaxel-lastname.
        - **Example:** junaid-innovaxel-husssnain.
    - Use the dev branch for main application code.
    - Keep the main branch limited to a README.md with setup instructions and notes.
    - Ensure at least 15 meaningful commits. Repositories with fewer than 10 commits will not be reviewed.
    - Commit messages must be concise (7 words max).
- **Code Review:** Add the following reviewers to your repository:
    - `Junaid Hussnain`
- **Public Repository:**
    - Keep the repository public for review.
    - After adding Junaid as reviewer, please inform us by sending an email to `HR Innovaxel` with a link to your repository.

- You are free to use any inspirations or online resources as long as you understand every line of code you write. During the interview, you will be expected to describe each aspect of your solution in detail and defend the decisions and lines of code you have written. This is your opportunity to showcase your problem-solving, technical skills, and thought process.

GOOD LUCK!