URL Shortener Service

Design a server capable of generating, tracking and managing shortened URLs with specific functionalities. The server should offer various endpoints for interaction:

- An endpoint to generate the shortened URL. This endpoint should account for the following:
    - **Custom** aliases (custom URLs suggested by the user).
    - Generate an **optimal** unique alias if not provided by the user. If the same URL is passed for shortening again, ensure that a new shortened URL is generated every time.
    - **TTL (Time to live)**: Each alias will have a TTL. The duration for which can be specified by the user. If not specified, the default TTL should be of 120 seconds.
- An endpoint which **redirects** the user to the long url based on the alias provided.
- An endpoint to get the **analytics** of an active alias. This endpoint will return basic information about the shortened URLs such as the number of visits and the last 10 access times. For better understanding, check the example in the API specification.
- An endpoint to update the Custom alias, TTL of the specified existing shortened URL. When an alias is updated using Custom alias, the analytics data of the old alias should be deleted.
- An endpoint to delete the specified existing shortened URL

Apart from the above endpoints, the mapping for the shortened URLs must be deleted from the datastore once the TTL duration has passed.

Constraints

- Ensure quick redirections with low latency.
- Don't use any databases. Implement the solution with in-memory variables.
- The complexity of endpoint requests should be aimed at O(log n) or O(1) for scalability and efficiency.

API Specification

1. Shortening URLs

Endpoint: `POST /shorten`

Description: Generate an optimal unique alias for a given long URL. Optionally, a custom alias and a time-to-live (TTL) can be specified.

Request Body:

```
{
  "long_url": "https://www.example.com/some/very/long/url",
  "custom_alias": "myalias",    // Optional
  "ttl_seconds": 60             // Optional
}
```

Response:

```
{
  "short_url": "http://localhost:<port>/myalias"    // or a generated
alias if      no custom_alias is provided
}
```

2. Redirecting

Endpoint: `GET /:alias`

Description: Redirect to the original long URL associated with the given alias.

Path Parameters:

- `alias`: The shortened URL alias.

Response:

- 302 Temporary Redirect: Redirects to the original long URL.
- 404 Not Found: Alias does not exist or has expired.

3. URL Analytics

Endpoint: `GET /analytics/:alias`

Description: Retrieve access statistics for a given shortened URL alias.

Path Parameters:

- `alias`: The shortened URL alias.

Response:

```
{
  "alias": "myalias",
  "long_url": "https://www.example.com/some/very/long/url",
  "access_count": 150,
  "access_times": [
    "2024-07-21T12:34:56Z",
    "2024-07-21T12:35:56Z",
    ...
  ]
}
```

4. Updating URLs

Endpoint: `PUT /update/:alias`

Description: Update the custom alias and TTL for a given alias.

Request Body:

```
{

  "custom_alias": "newalias",    // Optional

  "ttl_seconds": 90             // Optional

}
```

Response:

- `200 OK`: Successfully updated.
- `400 Bad Request`: Invalid request.
- `404 Not Found`: Alias does not exist or has expired.

5. Deleting URLs

Endpoint: `DELETE /delete/:alias`

Description: Delete the shortened URL data for the given alias.

Path Parameters:

- `alias`: The shortened URL alias.

Response:

- `200 OK`: Successfully deleted.
- `404 Not Found`: Alias does not exist or has expired.