Project Documentation for Round-Robin Coupon Distribution System

1. Project Overview

The Round-Robin Coupon Distribution System is a backend server designed to generate, store, distribute, and track unique coupon codes using a round-robin approach. The system ensures that each user receives a unique coupon and cannot request another one within a restricted time period (e.g., 1 hour).

2. Technology Stack

- Backend: Node.js with Express.js

- Database: MongoDB (without Mongoose)

- Middleware: cors, dotenv, cookie-parser

- Hosting: Server hosting on Vercel

3. Features

- Generates unique coupon codes in the format `RRC-XXXXXX`.

- Ensures coupons are distributed in a round-robin fashion.

- Implements a cooldown period to prevent users from claiming multiple coupons in a short period.

- Stores new coupons separately from claimed coupons.

- Uses cookies and IP tracking to enforce claim restrictions.

- Dynamically generates new coupons when all are exhausted.

4. Live Site and Server URIs

Live Site URI: https://round-robin-coupon-client.vercel.app

Server-side URI: https://round-robin-coupon-server.vercel.app

5. GitHub Repositories

Client Code: https://github.com/sikderfahad/round-robin-coupon-client

Server Code: https://github.com/sikderfahad/round-robin-coupon-server

6. Setup & Installation

Prerequisites

Ensure you have the following installed:

- Node.js (v16+ recommended)

- MongoDB instance (local or cloud)

Installation Steps

1. Clone the repository:

```
git clone https://github.com/sikderfahad/round-robin-coupon-server

cd round-robin-coupon
```

2. Install dependencies:

```
yarn
```

3. Create a .env file and configure:

```
PORT=3000 / 5000

MONGO_URI=your_mongodb_connection_string
```

4. Start the server:

```
nodemon start
```

The server should now be running at http://localhost:5000

7. API Endpoints

1. Health Check

GET /

Response: "Coupon server is running..."

2. Get a Coupon

GET /get-coupon

Response:

```
{"success": true, "coupon": "RRC-123456"}
```

If the user requests a coupon within the cooldown time:

```
{"msg": "Server restriction: Please wait 59 minutes before requesting another coupon."}
```

8. Workflow

1. A user requests a coupon via /get-coupon.

2. The server checks cookies & IP address to see if the user is eligible.

3. If eligible, the server fetches the next available coupon from new-coupons.

4. The coupon is transferred from new-coupons to old-coupons.

5. If no new coupons are available, the server generates new coupons.

6. The coupon is sent to the user, and the request timestamp is stored in cookies.

9. Database Schema

New Coupons Collection:

```
{ "_id": "ObjectId", "coupons": ["RRC-123456", "RRC-987654"] }
```

Old Coupons Collection:

```
{ "_id": "ObjectId", "coupons": ["RRC-654321", "RRC-112233"] }
```

10. Abuse Prevention Strategies

- **Cooldown Period**: Each user can only claim one coupon per restricted time (e.g., 1 hour). This is enforced by checking cookies and IP addresses.

- **IP Tracking**: The system tracks users based on their IP address to prevent multiple requests from the same user in a short time.

-**Use localstorage with cookies also for preventing unnecessary Api calls. The time restricted user can’t be hit the server before the exact remaining time over.

- **Dynamic Coupon Generation**: The system generates new coupons when all the available coupons are claimed, ensuring that there are always new coupons for eligible users.

Thank You