Playvalve RoR Test

Objective

Create a `POST /v1/user/check_status` endpoint in a Ruby on Rails API-only application. This endpoint will perform a series of security checks to determine the ban status of a user.

Create a local git repository for your solution and send it to us in zip format

Request Format

- Endpoint: `POST /v1/user/check_status`
- Content-Type: `application/json`

Request Body:

```
{
  "idfa": "8264148c-be95-4b2b-b260-6ee98dd53bf6",
  "rooted_device": false
}
```

Response Format

Response Body:

```
{
  "ban_status": "not_banned" // or "banned"
}
```

Security Checks

1. CF-IPCountry Header Whitelisting: Ban the user if the `CF-IPCountry` header value is not in the Redis country whitelist. Note: The application is behind a Cloudflare proxy, which passes this header.
2. Rooted Device Check: Ban if `rooted_device` is `true`.
3. IP Check for Tor/VPN: Ban if the IP is identified as Tor or VPN. Use VPNAPI (https://vpnapi.io/api-documentation) for this check. Cache the VPNAPI responses in Redis for 24 hours. If VPNAPI check fails (rate limit, server error), consider the check as passed.

User Record Handling in PostgreSQL

- Create a new user record if `IDFA` does not exist.

- Update the user record if `IDFA` exists.
- If an existing user is already banned, skip the check chain and return "banned" status.
- Re-run the checks for returning "not_banned" users.
- Include the following fields in `User` model:
  - `idfa`
  - `ban_status` (initial values: `banned`, `not_banned`)
  - `created_at`
  - `updated_at`

Integrity Log Record in PostgreSQL

- Create a log record when a new user is created or an existing user's `ban_status` changes.
- Include the following fields in the `IntegrityLog` model:
  - `idfa`
  - `ban_status`
  - `ip`
  - `rooted_device`
  - `country`
  - `proxy`
  - `vpn`
  - `created_at`

Future-Proofing

- Implement a service for the integrity logger to allow future re-routing of logs to other data sources.
- Design the `User` model to accommodate additional `ban_status` values in the future.

Technical Stack

- Ruby on Rails 7 (API only)
- PostgreSQL
- Redis
- RSpec/FactoryBot for testing

Testing Requirements

- Provide comprehensive tests using RSpec.