Technical Documentation

Architecture Overview

- **Technology**: Laravel (PHP Framework)
- **Database**: MySQL

Features

Authentication

- Sign Up: Users can access the system by signing up with their phone number and verification through an OTP code. Then, the user continues to fill in their data.
- Sign In: Users navigate to the home page by signing in with an existing account, having completed all steps with their phone number**.**

Categories Displayed on Home Screen

- Trending Events: Displays events with the most bookings, indicating popularity**.**
- Featured Events: Highlighted by app administrators for special visibility**.**
- Local Events: Shows events based on the user's city, providing localized options**.**
- Followed Organizers: Events organized by entities that the user follows, offering personalized suggestions.
- Offer Events: Shows events that are marked as offering with a percent discount**.**
- Organizers Section: Displays all organizers within the system.

Venue and Location

- Each event is associated with a venue that includes details like name, location, and capacity, enhancing the event discovery experience.

Reels Feature

- Users can watch short video clips ("reels") related to events, venues, and other users, providing a dynamic way to explore what's happening.
Event Request Feature

Event Request Feature :

- Enables users to request organization services for their private events, such as birthday parties, by choosing venues, service providers, date & time, number of people, personal information, and any additional notes.

Ticketing System

- Each ticket should contain the person's information (first & last name, phone number, age, coupon code if exists), then they confirm the booking by using MTN E-payment APIs for secure and efficient ticket transactions.

Notification System

- Utilizes the pushy.me platform to send timely notifications to users, ensuring they stay informed about relevant events and updates.

User Types

- Organizer: Responsible for managing events, and each event has its own organizer.
- Service Provider: Offers services (e.g., DJ, bartender) for events**.**
- Normal User: Engages with the app by booking tickets, following organizers, and participating in events.
- Guest User: Shows the home page and some features**.**
- Admin: Manages the dashboard, overseeing the application's operations, including CRUD operations for everything in the application.

Friends

- Each user can see all other users and send friend requests to them.
- Each user can assign tickets and invite their friends.

Direction and Location

- Each user can see the event location on Google Maps and get directions from their location to the event**.**

Coupons

- These exist by adding new ones from the admin panel; users get their coupons by applying some filters inside the panel.

Development Environment:

Venue and Location
Each event is associated with a venue that includes details like name, location, and capacity, enhancing the event discovery experience.
Reels Feature
Users can watch short video clips ("reels") related to events, providing a dynamic way to explore what's happening.
Event Request Feature
Enables users to request organization services for their private events, such as birthday parties, by choosing venues and service providers.
Ticketing System
Integrates with MTN E-payment API's for secure and efficient ticket transactions.
Notification System
Utilizes pushy.me platform to send timely notifications to users, ensuring they stay informed about relevant events and updates.
User Types

- **Organizer**: Responsible for managing events and every events has own organizer .
- **Service Provider**: Offers services (DJ , lighting , e.g.) for events.
- **Normal User**: Engages with the app by booking tickets, following organizers, and participating in events.
- **Admin**: Manages the dashboard, overseeing the application's operations (CRUD operation for all thing in the application).

Development Environment

- **Laravel Version**: 10.0x

Required Dependencies

- PHP: Version ^8.1
- **beyondcode/laravel-websockets: Version ^1.14**

- guzzlehttp/guzzle: Version ^7.2
- laravel/framework: Version ^10.10
- laravel/sanctum: Version ^3.2
- laravel/socialite: Version ^5.9
- laravel/tinker: Version ^2.8
- pusher/pusher-php-server: Version ^7.2
- spatie/laravel-permission: Version ^6.1
- ext-openssl: Any version (*)

Development Dependencies

- fakerphp/faker: Version ^1.23
- laravel/pint: Version ^1.0
- laravel/sail: Version ^1.18
- mockery/mockery: Version ^1.4.4
- nunomaduro/collision: Version ^7.0
- phpunit/phpunit: Version ^10.1
- spatie/laravel-ignition: Version ^2.0

API Documentation

- **Authentication**: Endpoints for user registration, login, and authentication.
- **Events**: APIs for event listing, detail retrieval, creation, and management.
- **Payments**: Integration details with MTN Mobile Money for handling ticket sales and refunds.
- **User Profiles**: Management of user profiles and preferences.

Security Measures Implemented

1. Environment File Security

- **.env Protection**: Confirmed that the **.env** file is not publicly accessible. Verified Apache's default configuration blocks access.
- **Secure Database Credentials**: Implemented strong, unique passwords for database and service access.

2. HTTPS Implementation

- **SSL/TLS Certificate**: Secured the application with an SSL/TLS certificate from Let's Encrypt to ensure encryption of data in transit. Implemented rigorous data validation and sanitization.

3. CSRF Protection

- **CSRF Tokens**: Leveraged Laravel's built-in protection against Cross-Site Request Forgery (CSRF) attacks. Confirmed that all forms and AJAX requests utilize CSRF tokens.

4. XSS and SQL Injection Prevention

- **Escape Output**: Adopted Blade's **{{ }}** syntax for automatic output escaping, preventing XSS attacks.
- **Use of Eloquent and Query Builder**: Ensured that all database interactions use Eloquent or Query Builder, utilizing prepared statements to mitigate SQL injection risks.

5. File Uploads Security

- **Validation and Storage**: Enforced strict validation and sanitization for all user-uploaded files. Configured file storage outside of the public directory, serving them securely through Laravel.

Laravel Project Deployment Guide

Pre-Deployment Checklist

- Confirmed the VPS meets Laravel's PHP (^8.1) and other server requirements.
- Ensured Apache is configured with **mod_rewrite** enabled for pretty URLs.

- Installed and accessible MySQL or a compatible database service.
- Established SSH access to the VPS.

Environment Setup:

Install PHP and Extensions

- Executed: **sudo apt-get install php8.1 php8.1-cli php8.1-common php8.1-mbstring php8.1-gd php8.1-imagick php8.1-xml php8.1-mysql php8.1-zip php8.1-bcmath php8.1-soap php8.1-intl php8.1-xmlrpc php8.1-curl**

Install Composer

- Followed the official Composer installation guide and executed: **curl -sS https://getcomposer.org/installer | php; mv composer.phar /usr/local/bin/composer**

Setup Database

- Created a new database and recorded credentials for the environment file configuration.

Deploying Your Laravel Application

Upload Project

- Utilized SCP for project upload: **scp -r /path/to/laravel/project user@your-vps-ip:/path/to/webroot**

Configure Environment

- Copied and updated environment variables:

cp .env.example .env

nano .env # Updated APP_KEY, DB_DATABASE, DB_USERNAME, DB_PASSWORD, etc.

Install Dependencies

- Ran: **composer install --optimize-autoloader --no-dev**

Generate Application Key

- Executed: **php artisan key:generate**

Run Migrations

- Applied database migrations: **php artisan migrate**

Set Directory Permissions

- Adjusted permissions:

- sudo chown -R evento:www-data /home/evento
- sudo chmod -R 775 /home/evento/evento/storage
- sudo chmod -R 775 /home/evento/evento/bootstrap/cache