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.

Notification System

Utilizes the pushy.me platform to send timely notifications to users.

Categories Displayed on Home Screen:

- For you: followed restaurants
- New Opening: new opening restaurants
- Featured: All Featured restaurants.
- Offers: show all offers

Following: this section contai followed restaurants

Discover: all nearest restaurants to the user location

Explore: explore all restaurants with their location.

Profile: in the profile section user can see his bookings (upcoming and history), his notifications, can change his password, see his achievements and coupon codes, change app language (Arabic or English), also there is help center and terms of use, and he can delete his account or logout.

User Types:

- 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.

Development Environment

- **Laravel Version:** 10.0x

Required Dependencies

```
"require": {
    "php": "^8.1",
    "beyondcode/laravel-websockets": "^1.14",
    "guzzlehttp/guzzle": "^7.2",
    "laravel/framework": "^10.10",
    "laravel/passport": "*",
    "laravel/sanctum": "^3.2",
    "laravel/telescope": "^4.17",
    "laravel/tinker": "^2.8",
    "laravel/ui": "^4.2",
    "laravelcollective/html": "^6.4",
    "predis/predis": "^2.2",
    "pusher/pusher-php-server": "^7.2",
    "ramsey/uuid": "^4.7",
    "spatie/laravel-permission": "^6.0"
  },
  "require-dev": {
    "fakerphp/faker": "^1.9.1",
    "laravel/dusk": "*",
    "laravel/pint": "^1.0",
    "laravel/sail": "^1.18",
    "mockery/mockery": "^1.4.4",
    "nunomaduro/collision": "^7.0",
    "phpunit/phpunit": "^10.1",
    "spatie/laravel-ignition": "^2.0"
  }
```

API Documentation

- Authentication: Endpoints for user registration, login, and authentication.
- Restaurants: APIs for Restaurants listing, Restaurant detail.
- Reservation: APIs for choose table in a restaurant and reserve it.
- Filters: APIs for filter restaurants depend on distance, price, cuisine, date, time, guests, and table attributes.
- Payments: Integration details with MTN Mobile Money for handling ticket sales and refunds.
- User Profiles: Management of user profiles and preferences.

Security Measures

1. Environment File Security
   - Protecting .env: The full application configuration are in the .env file, which is protected by Apache by default as it blocks access to it.
   - Securing Database Credentials: We have implemented rules on passwords to ensure the use of strong and unique passwords and to protect user data. In addition, we use strong passwords to protect all external accounts used in the app development process.
2. Use HTTPS
   - SSL/TLS Certificate: We have used an SSL/TLS certificate to encrypt data in transit.

3. To prevent common web vulnerabilities, we perform rigorous data validation and sanitization.

4. CSRF Protection CSRF Tokens: In app development, we use the Laravel framework, which automatically protects against Cross-Site Request Forgery (CSRF) attacks. Ensure forms and AJAX requests utilize CSRF tokens.

5. Output escape from XSS and SQL injection prevention: To do this we use:

- Blade's {{ }} syntax which automatically escapes output to prevent XSS attacks.
- Eloquent and Query Builder: These automatically use prepared statements, which make SQL injection attacks vastly more difficult

6. File Uploads Security

Validation: Strictly validate and sanitize all user-uploaded files.

Deployment

• For deploying your Laravel project, I'll draft a deployment guide that covers essential steps and best practices. This guide assumes you are deploying to a VPS (Virtual Private Server) running a LAMP (Linux, Apache, MySQL, PHP) stack, as mentioned earlier with your hosting on a Syrian ISP and using Apache as the web server

Laravel Project Deployment Guide

1. Pre-Deployment Checklist

Before deploying, ensure that:

- Your VPS meets the Laravel requirement for PHP (^8.1) and other server requirements.
- Apache is configured with mod_rewrite enabled for pretty URLs.
- MySQL or a compatible database service is installed and accessible.
- SSH access to your VPS is set up.

2. Environment Setup

Install PHP and Extensions

Ensure PHP 8.1 or higher and the required PHP extensions (including ext-openssl) are installed. You can install PHP and the necessary extensions using your Linux distribution's package manager.

Install Composer

Composer is required for managing Laravel's dependencies. If it's not already installed, you can follow the official Composer installation instructions.

Setup Database

Create a new database for your Laravel application and note the credentials, as you will need them for configuring your environment file.

3. Deploying Your Laravel Application

Upload Your Project

Upload your Laravel project files to your VPS. This can be done via **SCP, SFTP**, or directly pulling from a **Git** repository.

Configure Environment

Copy **.env.example** to **.env** and update the environment variables, especially **APP_KEY**, database credentials (**DB_DATABASE, DB_USERNAME, DB_PASSWORD**), and any other services your application uses.

Install Dependencies

Run composer **install --optimize-autoloader --no-dev** in the root directory of your Laravel project to install required dependencies without development packages.

Generate Application Key

If not already set, generate a new application key with **php artisan key:generate**.

Run Migrations

Apply your database migrations with **php artisan migrate**. This will set up your application's database schema.

Directory Permissions

Ensure the storage and bootstrap/cache directories are writable by the web server. A common approach is to use:

```
sudo chown -R www-data:www-data storage bootstrap/cache

sudo chmod -R 775 storage bootstrap/cache
```