## Project Overview

This project is a comprehensive backend system built using the Laravel PHP framework, designed exclusively to support a web-based application. The main objective was to create a secure, scalable, and efficient backend architecture that manages user data, authentication, business logic, and reporting within a web environment.  
  
The backend serves as the foundation for the web application, providing RESTful APIs, dynamic data management, and robust administrative control. It ensures consistent performance, maintainability, and flexibility for future module integration or feature expansion.

Key Objectives:

- Develop a robust backend architecture using Laravel.  
- Enable secure authentication and user management.  
- Implement role-based access and permissions.  
- Provide APIs for web-based frontend integration.  
- Ensure reliable data storage, reporting, and export capabilities.

## Technical Details

Technology Stack:

- Framework: Laravel (PHP 10.x compatible)  
- Database: MySQL   
- Frontend Integration: JSON response format  
- Authentication: Laravel Passport / Sanctum for token-based authentication  
- Server: Apache   
- Version Control: Git & GitHub for source management  
- Environment Management: .env configuration for secure deployment  
- PDF & Report Generation: dompdf / barryvdh-laravel-dompdf  
- Scheduler & Queue Handling: Laravel Scheduler, Redis / Queue Jobs  
- Deployment Environment: On-premise or cloud-based web server (Windows)

## Project Setup and Run Instructions

The project was deployed on a client infrastructure consisting of two separate servers: one for the application and another for the database. This setup provides improved performance, security, and scalability by isolating the backend logic from the data layer.

### 1. Database Server Setup

Steps to configure the Database Server (SQL):  
  
a. Install MySQL:  
 sudo apt update  
 sudo apt install mysql-server  
  
b. Create a new database and user:  
 CREATE DATABASE laravel\_backend;  
 CREATE USER 'laravel\_user'@'%' IDENTIFIED BY 'YourStrongPassword';  
 GRANT ALL PRIVILEGES ON laravel\_backend.\* TO 'laravel\_user'@'%';  
 FLUSH PRIVILEGES;  
  
c. Edit the MySQL configuration file (/etc/mysql/mysql.conf.d/mysqld.cnf) to allow remote connections:  
 bind-address = 0.0.0.0  
  
d. Restart MySQL service:  
 sudo systemctl restart mysql  
  
e. Allow only the Application Server’s IP to access port 3306 on the Database Server using firewall rules.

### 2. Application Server Setup

Steps to set up the Laravel backend on the Application Server:  
  
a. Install necessary packages:  
 sudo apt update  
 sudo apt install apache2 php php-mbstring php-xml php-bcmath php-curl php-zip unzip git composer  
  
b. Clone the repository:  
 git clone https://github.com/your-repo/laravel-backend.git  
 cd laravel-backend  
  
c. Install dependencies:  
 composer install  
  
d. Copy and edit the .env file:  
 cp .env.example .env  
  
Update the database configuration as below:  
 APP\_NAME="Laravel Backend"  
 APP\_ENV=production  
 APP\_KEY=base64:GeneratedKeyHere  
 APP\_DEBUG=false  
 APP\_URL=http://yourdomain.com  
 DB\_CONNECTION=mysql  
 DB\_HOST=(DataBase Port.NO)  
 DB\_PORT=3306  
 DB\_DATABASE=laravel\_backend  
 DB\_USERNAME=laravel\_user  
 DB\_PASSWORD=YourStrongPassword  
  
e. Generate the application key:  
 php artisan key:generate  
  
f. Run database migrations:  
 php artisan migrate --seed  
  
g. Set correct permissions:  
 sudo chmod -R 775 storage bootstrap/cache  
  
h. Configure Apache or Nginx to serve the Laravel public directory.  
  
i. Start the Laravel application:  
 php artisan serve --host=0.0.0.0 --port=8000

### 3. Security and Networking Notes

- Only allow port 3306 communication between Application Server and Database Server.  
- Restrict all external access to the Database Server using firewall rules.  
- Configure SSL (HTTPS) on the Application Server for secure communication.  
- Ensure SSH access is limited to authorized IP addresses only.

### 4. Maintenance and Logs

- Logs are stored at /storage/logs/laravel.log.  
- Use the following commands for maintenance tasks:  
 php artisan cache:clear  
 php artisan config:clear  
 php artisan route:clear  
 php artisan optimize  
- Regularly monitor disk space, backup the database, and update Composer dependencies.

## Deliverables

1. Backend Application Codebase – Laravel project source code with modular structure.  
2. Database Schema & Migrations – Complete migration files for users, records, logs, and analytics tables.  
3. API Endpoints Documentation – RESTful API documentation for web frontend integration.  
4. Admin Dashboard Backend Layer – Controllers for user management, leaderboard, and summary data.  
5. PDF & Report Generation Module – Export features for dashboards, user summaries, and logs.  
6. Setup Documentation – Full guide for deploying on separate Application and Database servers.  
7. Maintenance & Update Support – Procedures for security patches and performance optimization.

## Challenges & Solutions

|  |  |  |
| --- | --- | --- |
| **Challenge** | **Description** | **Solution Implemented** |
| Database Connectivity | Connecting Laravel application to a remote MySQL server securely. | Configured .env with remote host IP, enabled MySQL remote access, and used firewall rules for controlled access. |
| Data Accuracy in Reports | Inconsistent progress and time calculations for users. | Introduced a centralized CalculationService for accurate data aggregation. |
| PDF Export Formatting | Misalignment in generated reports. | Developed dedicated Blade templates for each report type. |
| Role-Based Data Access | Difficulty managing admin and user permissions. | Implemented middleware with Laravel Gates and Policies. |
| Large Data Processing | Slow performance with large datasets. | Applied query optimization, eager loading, and pagination techniques. |