# RS-Express Bus Booking System - Installation Guide

## System Requirements

### Minimum Requirements

- PHP 8.1 or higher
- MySQL 5.7 or higher
- Node.js 16.0 or higher
- npm 8.0 or higher
- Web server (Apache or Nginx)
- Composer 2.0 or higher

### Recommended Server Specifications

- CPU: 2+ cores
- RAM: 4GB or higher
- Storage: 20GB or higher
- Operating System: Ubuntu 20.04 LTS or Windows Server 2019

## Installation Steps

### Backend Installation (Laravel)

#### 1. Clone the Repository

```
git clone https://github.com/your-organization/rs-express.git
cd rs-express/Back-end(working)
```

#### 2. Install PHP Dependencies

```
composer install --no-dev --optimize-autoloader
```

### 3. Environment Configuration

```
cp .env.example .env
```

Edit the `.env` file with appropriate values:

```
APP_NAME="RS-Express"
APP_ENV=production
APP_KEY=
APP_DEBUG=false
APP_URL=https://your-domain.com

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=rs_express
DB_USERNAME=your_db_username
DB_PASSWORD=your_db_password

MAIL_MAILER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=your_mail_username
MAIL_PASSWORD=your_mail_password
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=noreply@your-domain.com
MAIL_FROM_NAME="${APP_NAME}"

SESSION_DOMAIN=your-domain.com
SANCTUM_STATEFUL_DOMAINS=your-domain.com
```

### 4. Generate Application Key

```
php artisan key:generate
```

### 5. Create Database

Log in to MySQL and create a database:

```sql
CREATE DATABASE rs_express CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
GRANT ALL ON rs_express.* TO 'your_db_username'@'localhost' IDENTIFIED BY
'your_db_password';
FLUSH PRIVILEGES;
```

#### 6. Run Database Migrations and Seed Data

```
php artisan migrate --seed
```

#### 7. Link Storage

```
php artisan storage:link
```

#### 8. Set Directory Permissions

```
chmod -R 775 storage bootstrap/cache
chown -R www-data:www-data storage bootstrap/cache
```

#### 9. Cache Configuration (for production)

```
php artisan config:cache
php artisan route:cache
php artisan view:cache
```

### Frontend Installation (React)

#### 1. Navigate to Frontend Directory

```
cd ../bus(working)
```

#### 2. Install JavaScript Dependencies

```
npm install
```

#### 3. Create Production Build

```
npm run build
```

# Web Server Configuration

## Apache Configuration

Create a new virtual host configuration:

```
<VirtualHost *:80>
    ServerName your-domain.com
    ServerAlias www.your-domain.com
    DocumentRoot /path/to/rs-express/Back-end(working)/public

    <Directory /path/to/rs-express/Back-end(working)/public>
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/rs-express-error.log
    CustomLog ${APACHE_LOG_DIR}/rs-express-access.log combined
</VirtualHost>
```

Enable the virtual host and restart Apache:

```
a2ensite rs-express.conf
systemctl restart apache2
```

## Nginx Configuration

Create a new server block:

```
server {
    listen 80;
    server_name your-domain.com www.your-domain.com;
    root /path/to/rs-express/Back-end(working)/public;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-Content-Type-Options "nosniff";

    index index.php;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    error_page 404 /index.php;

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.(?!well-known).* {
        deny all;
    }
}
```

Test and restart Nginx:

```
nginx -t
systemctl restart nginx
```

## Frontend Deployment

**Option 1: Serve from the same domain (subfolder)**

Copy the built frontend files to a subfolder in the Laravel public directory:

```
cp -r bus(working)/build/* Back-end(working)/public/app/
```

Access the frontend at `https://your-domain.com/app`

**Option 2: Serve from a subdomain**

Create a new Nginx server block for the frontend:

```
server {
    listen 80;
    server_name app.your-domain.com;
    root /path/to/rs-express/bus(working)/build;

    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
        expires max;
        log_not_found off;
    }
}
```

## SSL Configuration (Recommended)

### Using Certbot (Let's Encrypt)

**For Apache:**

```
apt-get update
apt-get install certbot python3-certbot-apache
certbot --apache -d your-domain.com -d www.your-domain.com
```

**For Nginx:**

```
apt-get update
apt-get install certbot python3-certbot-nginx
certbot --nginx -d your-domain.com -d www.your-domain.com
```

## Post-Installation Steps

### Create Admin User

If the default admin wasn't created during seeding, you can create one:

```
cd Back-end(working)
php artisan tinker
```

Inside tinker:

```
$user = new App\Models\User();
$user->name = "Admin User";
$user->email = "admin@example.com";
$user->password = Hash::make("secure_password");
$user->role = "admin";
$user->save();
exit
```

### Verify Installation

1. Access the backend API at: `https://your-domain.com/api`
2. Access the frontend at: `https://your-domain.com` (or the configured subdomain)
3. Log in with the admin user created above

## Troubleshooting

### Common Issues

#### 1. 500 Internal Server Error

- Check Laravel logs at `storage/logs/laravel.log`
- Ensure proper permissions on `storage` and `bootstrap/cache` directories
- Verify `.env` configuration

#### 2. Database Connection Error

- Verify database credentials in `.env`
- Check if MySQL service is running

#### 3. API Requests Failing from Frontend

- Verify CORS settings in `config/cors.php`
- Ensure API URL is correctly set in frontend environment

#### 4. File Upload Issues

- Check PHP `upload_max_filesize` and `post_max_size` in php.ini
- Verify storage symlink is created

#### 5. Application Key Error

- Ensure `APP_KEY` is set in `.env` by running `php artisan key:generate`

## Maintenance

### Regular Maintenance Tasks

1. Update dependencies:

   ```
   composer update
   npm update
   ```

2. Clear cache after updates:

   ```
   php artisan cache:clear
   php artisan config:clear
   php artisan view:clear
   ```

3. Database backups:

   ```
   mysqldump -u username -p rs_express > backup_$(date +%Y%m%d).sql
   ```

### Automatic Backups (Optional)

Set up a cron job for regular backups:

```
0 2 * * * mysqldump -u username -p'password' rs_express >
/path/to/backups/backup_$(date +\%Y\%m\%d).sql
```

### Log Rotation

Ensure Laravel logs are properly rotated by configuring `config/logging.php` or using the system's logrotate.

## Updating the Application

### Backend Updates

```
cd Back-end(working)
git pull origin main
composer install --no-dev --optimize-autoloader
php artisan migrate
php artisan config:cache
php artisan route:cache
php artisan view:cache
```

### Frontend Updates

```
cd bus(working)
git pull origin main
npm install
npm run build
# Copy the build files to their deployment location
```

## Security Recommendations

1. Enable HTTPS for all traffic
2. Set secure permissions on server files
3. Keep all dependencies updated
4. Use strong passwords for database and admin accounts
5. Configure a firewall to limit access to server ports
6. Enable rate limiting for API endpoints (already configured in Laravel)
7. Regularly back up your database
8. Monitor server logs for suspicious activity