Personal Info Management System - Complete Setup Guide

This guide will help you set up the Personal Info Management System from scratch on a new laptop/computer.

System Requirements

Hardware Requirements

- **RAM**: Minimum 8GB, Recommended 16GB or higher
- **Storage**: Minimum 10GB free disk space
- **Processor**: Modern dual-core processor or better

Software Requirements

1. PHP

- **Version**: PHP 8.1 or higher (8.2 recommended)
- **Required Extensions**:

- php-cli

- php-fpm

- php-mysql (or php-pgsql for PostgreSQL)

- php-xml

- php-curl

- php-zip

- php-gd

- php-mbstring

- php-bcmath

- php-json

- php-tokenizer

- php-ctype

- php-fileinfo

- php-opcache (recommended for performance)

2. Web Server

- **Apache 2.4+** with mod_rewrite enabled OR
- **Nginx 1.18+** with proper rewrite rules OR
- Use PHP's built-in development server (for development only)

3. Database

- **MySQL 5.7+** or **MariaDB 10.3+** OR
- **PostgreSQL 12+** OR
- **SQLite 3.8+** (for development)

4. Node.js & NPM

- **Node.js**: Version 18.x or 20.x (LTS versions recommended)
- **NPM**: Version 9.x or 10.x (comes with Node.js)

5. Composer

- **Composer**: Version 2.0 or higher

6. Git

- **Git**: Version 2.30 or higher (for version control)

Installation Steps

Step 1: Install Prerequisites

Windows Installation

- **Install XAMPP** (Recommended for Windows)

- Download from: https://www.apachefriends.org/

- Install XAMPP which includes Apache, MySQL, PHP, and phpMyAdmin

- Start Apache and MySQL services from XAMPP Control Panel

- **Install Node.js**

``cmd

Download and install from https://nodejs.org/

Choose the LTS version

Verify installation:

node --version

npm --version

`

- **Install Composer**

`cmd`

Download and install from https://getcomposer.org/download/

Verify installation:

composer --version

`

- **Install Git**

`cmd`

Download and install from https://git-scm.com/download/win

Verify installation:

git --version

`

macOS Installation

- **Install Homebrew** (if not installed)

`bash`

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

`

- **Install PHP, Composer, Node.js, and MySQL**

`bash`

brew install php composer node mysql

`

- **Start MySQL**

`bash`

brew services start mysql

`

Linux (Ubuntu/Debian) Installation

- **Update System**

`bash

sudo apt update && sudo apt upgrade -y

`

- **Install PHP and Extensions**

`bash

sudo apt install php8.2 php8.2-fpm php8.2-mysql php8.2-xml php8.2-curl php8.2-zip php8.2-gd php8.2-mbstring php8.2-bcmath php8.2-json php8.2-tokenizer php8.2-ctype php8.2-fileinfo php8.2-opcache -y

`

- **Install Composer**

`bash

curl -sS https://getcomposer.org/installer | php

sudo mv composer.phar /usr/local/bin/composer

sudo chmod +x /usr/local/bin/composer

`

- **Install Node.js**

`bash

curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -

sudo apt install nodejs -y

`

- **Install MySQL**

`bash

sudo apt install mysql-server -y

sudo mysql_secure_installation

`

- **Install Git**

`bash

sudo apt install git -y

`

Step 2: Clone the Project

- **Navigate to your projects directory**

```bash
cd C:\Projects # on Windows
# or
cd ~/Projects # on macOS/Linux
```

- **Clone the repository**

```bash
git clone https://github.com/totbenz/personal-info-management-system.git
cd personal-info-management-system
```

Step 3: Install PHP Dependencies

- **Install Composer dependencies**

```bash
composer install
```

If you encounter memory limit errors, run:

```bash
composer install --memory-limit=2G
```

Step 4: Setup Environment Configuration

- **Copy environment file**

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

- **Generate Application Key**

```bash
php artisan key:generate
```

- **Edit .env file** (Configure according to your database setup)

`env

APP_NAME="Personal Info Management System"

APP_ENV=local

APP_KEY=base64:YOUR_GENERATED_KEY_HERE

APP_DEBUG=true

APP_URL=http://localhost:8000

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=personal_info_system

DB_USERNAME=root

DB_PASSWORD=your_mysql_password

Other settings...

`

Step 5: Create Database

- **Create MySQL database**

- Using phpMyAdmin:

- Open http://localhost/phpmyadmin

- Click "New" to create a new database

- Name: personal_info_system

- Click "Create"

- Or using MySQL command line:

`sql

mysql -u root -p

CREATE DATABASE personal_info_system;

EXIT;

`

Step 6: Run Database Migrations

- **Run migrations**

`bash

php artisan migrate

`

- **Seed the database** (if seeders are available)

`bash

php artisan db:seed

`

Step 7: Install Node.js Dependencies

- **Install NPM packages**

`bash

npm install

`

Step 8: Build Frontend Assets

- **Compile assets for development**

`bash

npm run dev

`

For production:

`bash

npm run build

`

Step 9: Set File Permissions

Linux/macOS

```
sudo chown -R $USER:$USER storage bootstrap/cache
sudo chmod -R 775 storage bootstrap/cache
```

Windows

File permissions are usually not an issue on Windows, but ensure:

- IIS/IUSR or Apache user has read/write access to storage `and` bootstrap/cache `folders`

Step 10: Start the Development Server

- **Using PHP's built-in server**

`` `bash ``

php artisan serve

`

Then visit: http://localhost:8000

- **Using Apache/Nginx** (Production setup)

- Point your web server's document root to the public `directory`

- Configure virtual host for the domain

Additional Configuration

PDF Generation Dependencies

The project uses PDF generation libraries. You may need:

- **For Windows**: Usually works out of the box
- **For Linux**: Install additional packages

`` `bash ``

sudo apt install wkhtmltopdf xvfb -y

`

Email Configuration (Optional)

Update these in your .env `file:`

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=your_email@gmail.com
MAIL_PASSWORD=your_app_password
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="${APP_NAME}"
MAIL_FROM_NAME="${APP_NAME}"
```

Storage Configuration (Optional)

For file uploads, ensure:

- storage/app/public `directory is writable`
- Run php artisan storage:link `to create symbolic link`

Common Issues & Solutions

1. Composer Memory Limit Error

```
composer install --memory-limit=2G
```

or

```
php -d memory_limit=2G /usr/local/bin/composer install
```

2. Node.js Version Mismatch

Install specific Node version using nvm

```
nvm install 20
nvm use 20
```

3. Database Connection Error

- Verify MySQL/MariaDB service is running
- Check database credentials in .env
- Ensure database exists

4. Permission Issues (Linux/macOS)

```
sudo chmod -R 775 storage bootstrap/cache
sudo chown -R www-data:www-data storage bootstrap/cache  # for web server user
```

5. Vite Build Issues

```
# Clear cache and rebuild
npm run build -- --force
# or
rm -rf node_modules package-lock.json
npm install
npm run build
```

6. Livewire Not Working

- Ensure assets are built: npm run dev
- Check that @livewireStyles `and` @livewireScripts `are in your layout`
- Verify routes are cached: php artisan route:clear

Development Workflow

Daily Development Commands

```
# Start development server
php artisan serve

# Watch for asset changes (in separate terminal)
npm run dev

# Clear caches if needed
php artisan cache:clear
php artisan config:clear
php artisan route:clear
php artisan view:clear
```

Git Workflow

```
# Check status
git status

# Add changes
git add .

# Commit changes
git commit -m "Your commit message"

# Push to remote
git push origin main
```

Production Deployment

1. Optimize for Production

```
# Install production dependencies
composer install --optimize-autoloader --no-dev

# Cache configuration
php artisan config:cache
php artisan route:cache
php artisan view:cache

# Build assets for production
npm run build -- --production
```

2. Set Up Cron Jobs

```
# Edit crontab
crontab -e

# Add Laravel scheduler
    * cd /path-to-your-project && php artisan schedule:run >> /dev/null 2>&1
```

3. Queue Worker (if using queues)

```
# Install supervisor to manage queue workers
sudo apt install supervisor

# Configure supervisor config in /etc/supervisor/conf.d/laravel-worker.conf
```

Security Considerations

- **Never commit .env `file`** to version control
- **Use HTTPS** in production
- **Set APP_DEBUG=false** in production
- **Regularly update dependencies**

`bash

composer update

npm update

`

- **Use strong passwords** for database
- **Enable firewall** on production server
- **Regular backups** of database and files

Support & Troubleshooting

If you encounter issues:

- Check Laravel logs: storage/logs/laravel.log
- Enable debug mode temporarily: APP_DEBUG=true `in` .env`
- Check PHP error logs
- Verify all requirements are met
- Check official documentation:

- Laravel: https://laravel.com/docs

- Livewire: https://livewire.laravel.com/docs

- Tailwind CSS: https://tailwindcss.com/docs

Quick Start Commands Summary

1. Clone project

```
git clone [repository-url]

cd personal-info-management-system
```

2. Install dependencies

```
composer install

npm install
```

3. Setup environment

```
cp .env.example .env

php artisan key:generate
```

4. Setup database

Create database manually

```
php artisan migrate

php artisan db:seed
```

5. Build assets

```
npm run dev
```

6. Start server

```
php artisan serve
```

Visit http://localhost:8000 to access your application!