GradeManagement — Beginner-friendly setup guide

Overview This guide explains, step-by-step, how to get the GradeManagement API running on a computer. It uses simple language and includes exact commands you can copy and paste. If a command asks for a password and you didn't change anything, try `root` as the password.

Prerequisites (what you need installed)

- PHP (run `php -v` to check)
- Composer (run `composer --version`)
- MySQL or MariaDB (or Docker if you prefer not to install MySQL locally)
- Optional: Git if you want to clone the repository

Quick summary

1. Install PHP dependencies
2. Ensure your database server is running (the app will auto-create the `grms` database, schema, and seed when you visit `/api`)
3. Configure the environment file
4. Start the API server and test

Detailed steps

1. Get the code

- If you already have the project files, skip this step.
- To clone the repository (example):

```
git clone https://example.com/your-repo.git
cd GradeManagement
```

2. Install PHP dependencies

- Move into the `Backend` folder and run Composer to install required PHP packages:

```
cd Backend
composer install
```

3. Configure environment variables

- Copy the example environment file and edit it to match your database settings:

```
cp example.env .env
# Open .env in a text editor and set DB_HOST, DB_PORT, DB_DATABASE, DB_USERNAME, DB_PASSWORD
```

- Example `.env` database settings (use these as a starting point):

```
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=grms
DB_USERNAME=root
DB_PASSWORD=root
```

4. Initialize the database (automatic)

- The application will automatically create the `grms` database, import the schema, and run the seed data when you visit the API root. To use this feature:

    1. Make sure your database server is running and `.env` contains correct DB credentials.
    2. Start the API server (see step 5).
    3. Open this URL in a browser: http://localhost:8001/api

    You should see a confirmation message in the browser or server logs once initialization completes. Manual SQL imports are not required for normal setup.

Notes:

- If you don't know the container name, run `docker ps` and look for a container that uses the `mysql` image.
- If the provided `docker-compose.yml` includes a phpMyAdmin service, you can also use it to import the `.sql` files via a web browser.

5. Start the API server

- Recommended (from project root): run the server from the `Backend` folder so PHP serves the `api` directory as the document root:

```
cd Backend
php -S localhost:8001 -t api
```

- Alternative (also valid): start the server from inside the `api` folder — all routes remain available and `/api` will still initialize the database:

```
cd Backend/api
php -S localhost:8001
```

- Open a browser and go to: http://localhost:8001/api to trigger initialization and view the API.
- If port `8001` is already in use, pick another port (for example `8002`) and run the same commands with that port.

If you use XAMPP or another Apache setup:

- Copy the contents of `Backend/api` to your XAMPP `htdocs` folder and follow your local server instructions.

6. Test a basic endpoint

- In your terminal, run:

```
curl http://localhost:8001/api
```

- Or open the URL in a web browser. For specific API endpoints and usage examples, see the API documentation in the `Documentation` folder.

Troubleshooting tips (common problems)

- "Command not found": install the missing tool (PHP, Composer, or Docker). Use your package manager (for example `sudo apt install php composer` on Debian/Ubuntu).
- "Access denied" when importing SQL: double-check `DB_USERNAME` and `DB_PASSWORD` in `.env`, and ensure the MySQL user exists.
- "Port already in use": pick another port for the PHP server.
- If composer fails, try `composer install --no-dev` or check your PHP version compatibility.

Last updated: January 2026