ASTUDIO Assessment - Laravel Project MHD HUSAM TAKAJI

This repository demonstrates a Laravel-based application for managing Projects, Timesheets, and dynamic attributes (EAV). It includes user authentication (via Laravel Passport), RESTful APIs, and advanced filtering (both on core fields and EAV attributes).

Github link: mhdhusamtakaji/astudio-assessment

Contents

Requirements

- PHP ^8.1
- Composer
- MySQL
- Laravel ^10.*
- Laravel Passport

Setup Instructions

1. Clone the Repository (or unzip the attached code file)

```
git clone https://github.com/mhdhusamtakaji/astudio-
assessment.git
cd astudio-assessment
```

2. Install Dependencies

```
composer install
```

3. Create and Configure .env

- Copy `.env.example` → `.env`
- Update your database credentials (DB_DATABASE, DB_USERNAME, DB_PASSWORD, etc.)
- Make sure `APP_URL` is set to something like `http://127.0.0.1:8000`

4. Generate App Key

```
php artisan key:generate
```

5. Run Migrations

```
php artisan migrate
```

6. Install Passport

```
php artisan passport:install
```

This command creates the encryption keys and OAuth client entries needed by Passport.

7. Seed the Database

```
php artisan db:seed
```

This populates a test user, example projects, attributes, and timesheets (optional) into your database.

8. Serve the Application

```
php artisan serve
```

By default, the application will be available at `http://127.0.0.1:8000`.

Database Migrations & Seeders

- **Migrations** create the schema for:
  - users, projects, timesheets, project_user, attributes, attribute_values.
- **Seeders** (via db:seed) create:
  - A test user with known credentials (see below).
  - Sample projects (Project A, Project B).
  - Common attributes (department, start_date, end_date).
  - An optional timesheet linking the test user to Project A.

Authentication

- **Laravel Passport** is used for token-based authentication.
- **Register:** POST /api/register
- **Login:** POST /api/login
- **Logout:** POST /api/logout (requires token in Authorization: Bearer <token> header)

Once logged in, you'll receive an `accessToken`. Include this token as a **Bearer token** in the `Authorization` header for all subsequent **protected** endpoints.

API Documentation

Here’s a brief overview of the major endpoints.
All **protected** routes require `Authorization: Bearer <token>`.

1. Auth Endpoints

Method	*URI*	*Description*
POST	/api/register	Create a new user (public)
POST	/api/login	User login, returns token (public)
POST	/api/logout	Revoke current token (requires auth)

2. Projects Endpoints

Method	*URI*	*Description*
GET	/api/projects	List projects, optional filters
GET	/api/projects/{id}	Show details of a single project + EAV data
POST	/api/projects	Create a new project, can set EAV attributes
PUT	/api/projects/{id}	Update project fields and EAV attributes
DELETE	/api/projects/{id}	Delete a project and its EAV data

3. Timesheets Endpoints

Method	*URI*	*Description*
GET	/api/timesheets	List timesheets belonging to the authenticated user
GET	/api/timesheets/{id}	Show a single timesheet (must own it)
POST	/api/timesheets	Create a new timesheet for the authenticated user + project
PUT	/api/timesheets/{id}	Update a timesheet (must own it)
DELETE	/api/timesheets/{id}	Delete a timesheet (must own it)

4. Attributes Endpoints

Method	*URI*	*Description*
GET	/api/attributes	List all Attributes
GET	/api/attributes/{id}	Show a single Attribute
POST	/api/attributes	Create a new Attribute
PUT	/api/attributes/{id}	Update an existing Attribute
DELETE	/api/attributes/{id}	Delete an Attribute and any related rows

---

Filtering

You can filter projects based on:

1. **Native columns**: name, status, created_at, etc.
2. **EAV attributes**: e.g., department, start_date.

Example:

```
GET /api/projects?filters[name][like]=Project&filters[department][=]=IT
```

- Filters where `projects.name LIKE "%Project%"` and the EAV attribute with name `department` equals `IT`.

Supported operators: `=`, `>`, `<`, `LIKE`.

Example Requests/Responses

1. **Register** (public):

```
POST /api/register
{
  "first_name": "John",
  "last_name": "Doe",
  "email": "john@example.com",
  "password": "secret",
  "password_confirmation": "secret"
}
```

2. **Login** (public):

   ```
   POST /api/login
   {
     "email": "john@example.com",
     "password": "secret"
   }
   ```

   Response:

   ```
   {
     "message": "Login successful.",
     "user": {
       "id": 2,
       "first_name": "John",
       "last_name": "Doe",
       "email": "john@example.com",
       ...
     },
     "token": "some-long-access-token"
   }
   ```

3. **Create Project** (requires auth):

   ```
   POST /api/projects
   Headers: Authorization: Bearer <token>

   {
     "name": "Project C",
     "status": "In Progress",
     "attributes": [
       { "attribute_id": 1, "value": "IT" },
       { "attribute_id": 2, "value": "2025-01-01" }
     ]
   }
   ```

4. **Filter Projects**:

   ```
   GET /api/projects?filters[status][=]=Open&filters[department][like]=IT
   ```

5. **Create Timesheet** (requires auth):

```
POST /api/timesheets
{
  "task_name": "Fix Bugs",
  "date": "2025-05-10",
  "hours": 4,
  "project_id": 1
}
```

Must be owned by the logged-in user.

Test Credentials

After running migrations and seeders, you have a test user:

- **Email**: `test@demo.com`
- **Password**: `password`

Use these credentials to log in (`POST /api/login`) and access protected endpoints quickly.

Postman Collections

We've included two **JSON** files in the repository for easy testing:

1. **Postman Collection**: `astudio-assessment.postman_collection.json`
   - Contains all requests for Auth, Projects, Timesheets, Attributes, with placeholders for the token.
2. **Postman Environment**: `astudio-assessment.postman_environment.json`
   - Defines environment variables like `{{base_url}}` and `{{token}}`.

How to use:

1. Import both `.json` files in Postman.
2. Select the environment (`astudio-assessment.postman_environment`).
3. Update the `base_url` variable if needed (e.g., `http://127.0.0.1:8000/api`).
4. Go to "Auth → Login User" request, fill in the credentials (or use the existing ones).

5. Click **Send**, and store the returned token in the environment variable: `{{token}}`.
6. You can now test all protected endpoints with `Bearer {{token}}`.

License

This project is provided as part of the ASTUDIO assessment.

Thank you for reviewing this project! If you have any questions or feedback, feel free to open an issue on the GitHub repository or contact the contributor.

MHD HUSAM TAKAJI
Software Engineer
Dubai, UAE

+971 505890129

LinkedIn: www.linkedin.com/in/mhd-husam-takaji-a1a074289

Github: mhdhusamtakaji (MHD HUSAM TAKAJI) (github.com)

Portfolio: ENG. MHD HUSAM TAKAJI