1. Basic Submission Information

Student Name - Joanna Masiewicz
Student ID - 250372209
Test User Credentials - Username: sam@test.com & joanna@test.com/ Password: password
Application Entry Page - http://127.0.0.1:8000/ (Redirects to Login)
Source Code Link - https://github.com/Joanna-devop/Back-end-project-manager-website

Database Data Submission - The database structure and test data are submitted using Laravel's file-based system. Structure Files (SQL equivalent): Located in the database/migrations/ folder. Data Files (SQL equivalent): Test users (Sam, Joanna, Dona) and demo projects are seeded via the database/seeders/DatabaseSeeder.php file.

2. Core Functional Requirements (CRUD)

The Project Manager application successfully implements all required CRUD (Create, Read, Update, Delete) functionality.

Create (Store): New projects are created and saved, automatically linking to the logged-in user's ID (uid). Source File: ProjectController.php (store method)

Read (Index): Projects are fetched and displayed in a filterable list, displaying only projects owned by the logged-in user. Source File: ProjectController.php (index method)

Update (Edit/Patch): Existing project details can be modified and saved only by the owner. Source File: ProjectController.php (update method)

Delete (Destroy): Projects can be permanently removed from the database only by the owner. Source File: ProjectController.php (destroy method)

Key Feature: Dynamic Search: Implemented dynamic search functionality by Title (fuzzy search) or Start Date (date picker). Source File: index.blade.php / ProjectController.php

3. Security Features Implemented

Security is enforced through middleware and a specific authorisation policy, critical for protecting user data.

Authentication (Login/Logout/Register): Standard Laravel/Breeze security ensures only registered users can access the application. Source File: routes/web.php (Middleware: auth)

Authorization (Project Policy): The core security feature, ensuring a user can only interact with projects they created.

Visibility Check: The list view query filters projects based on the current user's ID, denying unauthorized read access (demonstrated by Dona seeing an empty list).

Manipulation Check: The ProjectPolicy checks ownership before allowing edit or delete actions (demonstrated by the "403 UNAUTHORIZED" error when Dona attempts to edit Project ID 3).

Source File: app/Policies/ProjectPolicy.php (Methods: update, delete)

Data Validation: All incoming data via forms is validated to prevent common vulnerabilities (e.g., required fields, correct date format). Source File: ProjectController.php (Use of $request->validate(...))

4. Architectural Overview

Database Model Design
I used a relational database with a One-to-Many Relationship between users and projects. The Foreign Key uid in the projects table establishes the data ownership required for the security policy.

Advanced Routing and Control
Custom Primary Key: The projects table uses a custom primary key named pid instead of the default id.

Explicit Route Model Binding: The application required routes to be explicitly defined to look for the pid column (e.g., {project:pid}) when fetching a model from the URL. This streamlined controller methods by automatically providing the correct Project model instance based on the custom key.

5. Final Conclusion

The application successfully meets all core functional and security requirements. The implementation of a robust authorisation policy and the customization of Route Model Binding to accommodate the custom pid demonstrate a comprehensive understanding of secure, full-stack web development principles.