Requirements

- Php (8.4.7) - programming language
- Composer (2.8.9) - dependency manager
- Mysql (8.4.5 LTS) - RDBMS
- Mysql workbench (8.0.42) - Management Studio

MySQL Basic Operations

You can use **MySQL Workbench** or **MySQL CLI** to interact with the database, choose what option you like. Kindly follow the instructions:

- Open **MySQL Workbench** and key in the **root** password.
- Click on the **local instance** of MySQL.
- On the **navigator panel** (left panel), you will see 2 tabs named **Administration and Schema**. Click on Administration and click on the **Users and Privileges**.
- We will create a new user that will connect to this server on our laravel project. To do that, click on **Add Account** and fill out the following details:
    - Login Name
    - Authentication type: caching_sha2_password
    - Limit to Hosts Matching: %
    - Password and Confirm Password
- On the next tab, **Account Limits** just input **0** on all fields to denote that no max value for each field.
- On the next tab, **Administrative Roles**, click on **DBA** to check all items.
- Click on **Apply** to commit changes.

Kindly demonstrate how we create schema, tables and fields and how to insert, update and delete records using MySQL Workbench.

SQL statements for administration if you choose using **MySQL CLI** to manage your database server.

- **CREATE DATABASE** <*database_name*>
- **CREATE USER '**<*user_name*>**'@'**<*server_name*>**' IDENTIFIED BY '**<*password*>**'**
- **GRANT PRIVILEGES ON** <*database_name*> **. * TO '**<*user_name*>**'@'**<*server_name*>**'**

Eloquent Basic Operations

Kindly follow the instructions;

- Fork and clone this repository: https://github.com/rjasino/cs401-week-4-act-2
- After cloning open your terminal make sure that the current directory is the project folder and type **composer update** to restore the **vendor** folder.

- Make sure to rename the **.env.example**, remove the **.example** then open your terminal and type **php artisan key:generate** to generate the **APP_KEY** in your **.env** file.
- Configure your **.env** to use **mySQL** or **SQLite** depending on what works on your terminal.
- Demonstrate **Mysql Workbench** on how to create tables, columns and records.
- Consider the following table definition:

User

Id	Id	Primary key
Email	String	Unique
Password	String	
Name	String	***max(30)***
Remember_token	String	rememberToken()
Email_verified_at	Timestamp	Nullable
Registration_date	Timestamp	
Last_login_date	Timestamp	Nullable
Remove timestamps		

Role

Id	Id	Primary key
Role_name	String	***(A - Admin, C - Contributor, S - Subscriber) max(1)***
Description	String	
Timestamps	Timestamps	timestamps()

Post

Id	Id	Primary key
Title	String	
Content	***Text***	
Slug	String	
Publication_date	Timestamp	Nullable
Last_modified_date	Timestamp	Nullable

Status	String	(D - Draft, P - Published, I - Inactive) ***max(1)***
Featured_image_url	***Text***	
Views_count	Integer	default(0)

Category

Id	Id	Primary key
Category_name	String	(News, Review, Podcast, Opinion, Lifestyle, etc.) ***max(30)***
Slug	String	
Description	String	
Timestamps	Timestamps	timestamps()

Tag

Id	Id	Primary key
Tag_name	String	***max(45)***
Slug	String	
Timestamps	Timestamps	timestamps()

Comment

Id	Id	Primary key
Comment_content	***Text***	
Comment_date	Timestamp	
Reviewer_name	String	***Nullable***
Reviewer_email	String	***Nullable***
Is_hidden	Boolean	default(false)

Media

Id	Id	Primary key
File_name	String	

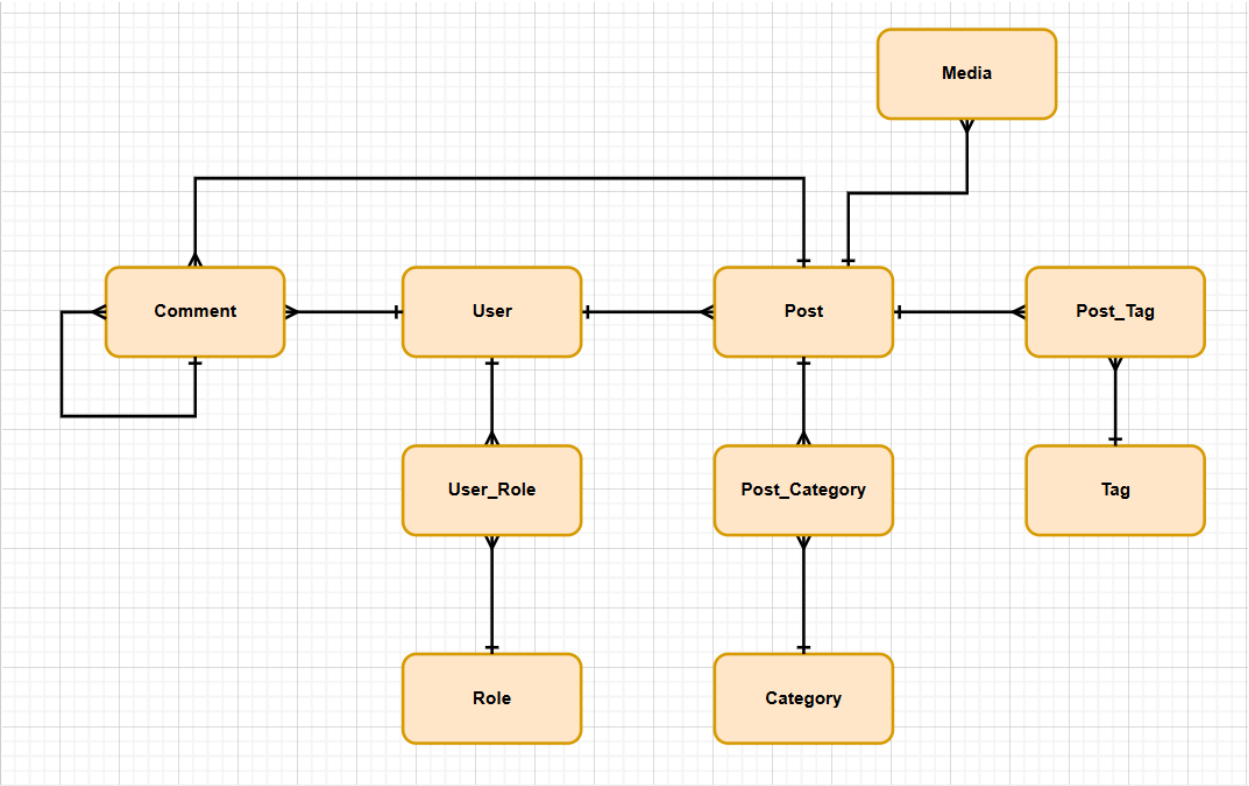
File_type	String	***max(10)***
File_size	Integer	***default(0)***
Url	***text***	
Upload_date	Timestamp	
Description	String	***Nullable***

Database Operation and Eloquent ORM

Migration is the Git of Database Management.

1. Create a model for each table on the list, open your terminal, make sure that it points to your project directory and type **php artisan make:model <*table_name*> -m.** This command will make the model at **app\Models** and migration at **database\migrations.**
2. Open up the migration file for each table and define its schema based on the table definition above. Make sure to define both the **up** and **down** function of the migration. **up** will be executed when migration will be executed and **down** for rollback. Side note, you can add comments to each field of your table to define use of a particular field.
3. After you define your schema, open your terminal and type **php artisan migrate** to execute the migration. You can open your mysql workbench to see if the tables were created.
4. If you want to make some changes, renaming, change of type, increase the size and any data definition action, just create a new migration. Open your terminal and type **php artisan make:migration <*name_of_migration*>**. Make the name of migration as verbose as possible to identify the action to perform.
5. If you want to squash migration use the command **php artisan schema:dump --prune** to dump and create a **.sql file** that will be run in the database server. With the **prune** option it will **delete** the migration file.

6. Create Eloquent model relationships for each table by considering the diagram below:



7. After you define the relationships, create a new migration for each of these relationships for the database to recognize.
8. One last thing before you execute the migration, check your database design and structure, check types, relationships, rules such as fillable, hidden and cast. Create a new migration if there are changes needed to correct at this point.
9. Create Seeders and Factories for each table by typing **php artisan make:seeders <*name*>Seeders** and **php artisan make:factory <*name*>Factory** respectively.
10. Seeders are functions that will execute the command to generate data on our table while Factories define what data to generate.

Migration Post Database Setup

During a project's development, it's common to find yourself needing to adjust your database's blueprint – perhaps adding a new table, tweaking a column's data type, or even removing a column entirely. However, once your project is live and in production, changing the database structure demands extreme caution. It's a critical decision that requires a lot of thought and planning because there's a significant risk of data loss or system disruption if not handled meticulously. While these changes are sometimes unavoidable, they are definitely not to be taken lightly.

- Adding a new column - this is generally safe.
- Renaming a column - requires careful handling of preserve data.

- Changing a column's data type - high risk of data loss if the existing data cannot be cast to the new type.
- Dropping a column - high risk of data loss.
- Adding/Dropping indexes or foreign keys: Generally safe, but consider performance implications.

Migration Common Commands

- **php artisan make:controller** <*controller_name*>**Controller -r -** make new controller with scaffolding, always put a suffix of Controller.
- **php artisan make:view** <*folder_name*>**.**<*view_name*> - folder name is optional, only if you want to organize your view under folder structure.
- **php artisan make:model** <*model_name*> **-m -** make new model with migration
- **php artisan make:migration** <*migration_file_name*> - make new migration, make it verbose as much as possible for clarity
- **php artisan schema:dump** - squash your migration into a single .sql file
- **php artisan schema:dump --prune** - squash your migration into a single .sql file and delete all migration files.
- **php artisan migrate** - run the migration
- **php artisan migrate:status** - check migrations have run thus far
- **php artisan migrate:rollback** - rollback the latest migration
- **php artisan migrate:rollback --step=***n* - rollback the last *n* migrations
- **php artisan migrate:rollback --batch=***n* - rollback the *n* batch migrations
- **php artisan migrate:reset** - rollback all migrations
- **php artisan migrate:refresh** - rollback and execute all your migrations
- **php artisan migrate:refresh --seed** - rollback, execute all your migrations and seed your database
- **php artisan migrate:fresh** - drop all tables from the database and then execute migrations
- **php artisan migrate:fresh --seed** - drop all tables from the database, execute migrations and then seed database
- **php artisan db:seed** - to seed your database
- **php artisan db:seed --class=**<*seeder_class*> - running specific seeder class