FILE SHARING APP USING PHP/LARAVEL

Project report (CA3) submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

RAKSHIT SINGH TOMAR

(12013985)

SUBJECT

INT221 - MVC PROGRAMMING

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

November, 2023

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROBLEM STATEMENT

In today's digital age, the need for efficient and secure file sharing and management has become increasingly crucial. Individuals, businesses, and organizations regularly deal with various types of files, including documents, images, audio, video, and more. Sharing and organizing these files often involves challenges such as data security, accessibility, collaboration, and the need for user-friendly interfaces. The project addresses these issues by developing a Laravel-based file sharing application with an array of features designed to simplify and enhance the file sharing and management process.

Key problems to address:

- Security: Ensuring files are shared securely, preventing unauthorized access.
- User-Friendly Interface: Providing a seamless and intuitive user experience for file sharing and management.
- File Organization: Enabling users to efficiently organize and categorize their files.
- Collaboration: Supporting user collaboration on shared files.
- Accessibility: Ensuring files are easily accessible and retrievable.
- File Preview and Playback: Enhancing user experience through in-app file previews and playback options for multimedia content.

1.2 WHY USE PHP AND LARAVEL?

The choice of PHP and the Laravel framework for the project is driven by several key factors:

- Robustness: PHP is a widely-used, battle-tested scripting language with a strong track record of web development. Laravel, built on PHP, extends this robustness by offering a wide range of features and tools.

- Scalability: Both PHP and Laravel are known for their scalability. This is crucial for the file sharing application, which must accommodate various user demands and growing data.
- Active Developer Community: The PHP and Laravel communities are vast and active, providing ongoing support, updates, and a wealth of resources for developers. This ensures the longevity and sustainability of the application.
- Rapid Development: Laravel's elegant syntax and built-in features allow developers to work quickly and efficiently, reducing development time and costs.

1.3 MVC ARCHITECTURE

The application adheres to the Model-View-Controller (MVC) architectural pattern. This approach separates the application into three interconnected components:

- Model: Represents the data and database interactions, responsible for managing data and business logic.
- View: Deals with the presentation and user interface, ensuring the content is appropriately displayed to users.
- Controller: Acts as an intermediary between the Model and View, processing user input and responding accordingly.

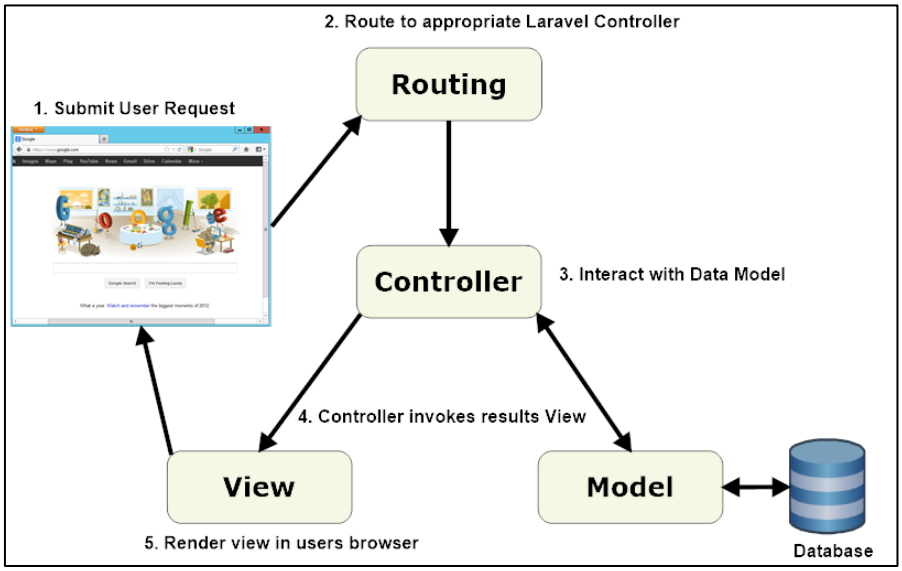

Figure 1: MVC architecture of the application

The Model-View-Controller (MVC) architecture is a design pattern commonly used in software development, particularly in the context of web applications. It separates the concerns and responsibilities of an application into three interconnected components: Model, View, and Controller. Each of these components has a distinct role in managing and delivering the functionality of the application.

Characteristics of Model:

- Manages data storage, retrieval, and updates.
- Contains the application's business logic, including validation rules and data transformation.
- Provides an interface for interacting with the database.
- Isolated from the user interface and presentation concerns.

Characteristics of View

- Renders and displays data to the user in a human-readable format.
- Receives user input, such as form submissions or user interactions.
- Often includes HTML templates, CSS, and other elements that control the visual representation of the application.
- Isolated from the application's business logic and data processing.

Characteristics of Controller:

- Receives and processes user requests, often through routing mechanisms.
- Interacts with the Model to retrieve or update data based on user input.
- Communicates with the View to ensure the appropriate presentation of data to the user.
- Orchestrates the flow of data and user interactions, ensuring that the Model and View remain isolated from each other.

Advantages of MVC:

- Separation of Concerns: MVC enforces a clear separation of concerns, allowing developers to work on individual components without interfering with the others. This makes code more modular, maintainable, and extensible.

- Code Reusability: Components within MVC can be reused in other parts of the application or in different applications, reducing redundancy and development time.
- Scalability: The modular structure of MVC makes it easier to scale and enhance specific components without impacting the entire application.
- Parallel Development: Developers can work in parallel on different components of the architecture, streamlining the development process.
- Testing: Each component can be tested in isolation, simplifying unit testing and quality assurance.

The project introduces a Laravel-based file sharing application to address the challenges associated with file management, focusing on security, user-friendliness, and collaboration. By leveraging PHP, Laravel, and the MVC architecture, the aim to deliver a robust, scalable, and maintainable solution that meets the evolving needs of users in various domains.

2. TECHNOLOGIES USED

2.1 PHP (Hypertext Preprocessor)

PHP is a widely-used server-side scripting language that is particularly well-suited for web development. It has several key features and characteristics:

- Dynamic Web Pages: PHP allows you to embed PHP code directly within HTML, making it easy to generate dynamic web pages. PHP code is executed on the server, and the resulting HTML is sent to the client's browser.
- Cross-Platform Compatibility: PHP is compatible with various operating systems, including Windows, Linux, macOS, and others, making it versatile for different hosting environments.
- Extensive Library Support: PHP has a vast standard library and an active community of developers who create open-source libraries and frameworks, enhancing its capabilities.
- Database Connectivity: PHP can connect to various databases, making it an ideal choice for building database-driven web applications. In your project, it's used to interact with the MySQL database.

2.2 LARAVEL (PHP Framework)

Laravel is a PHP web application framework known for its elegant syntax and robust features. It provides a structured and efficient way to develop web applications:

- Eloquent ORM: Laravel includes an ORM (Object-Relational Mapping) called Eloquent, which simplifies database interactions by allowing you to work with database tables using object-oriented models.
- Authentication and Authorization: Laravel provides built-in support for user authentication and authorization, streamlining the process of creating secure user systems.
- Routing: Laravel offers a clean and expressive way to define application routes, making it easy to create and manage URLs and associated actions.

- Blade Templating Engine: Blade is a powerful templating engine in Laravel, allowing you to create reusable views and layouts. It encourages the separation of business logic and presentation.
- Middleware: Middleware in Laravel enables you to filter HTTP requests entering the application. It's often used for tasks like authentication, logging, and CORS handling.

2.3 MYSQL (Relational Database Management System)

MySQL is a popular open-source relational database management system known for its reliability and performance:

- Relational Database: MySQL follows the relational data model, allowing you to structure data in tables with rows and columns. This makes it well-suited for storing structured data.
- ACID Properties: MySQL adheres to the ACID properties (Atomicity, Consistency, Isolation, Durability), ensuring data consistency and reliability, even in high-concurrency scenarios.
- Scalability: MySQL can handle both small-scale and large-scale applications, making it suitable for a wide range of projects.

2.4 TAILWIND CSS (CSS Framework)

Tailwind CSS is a utility-first CSS framework designed to simplify the process of building user interfaces:

- Modular and Reusable Components: Tailwind CSS promotes the creation of modular and reusable design components, making it easy to maintain a consistent design across an application.
- Customization: The framework allows you to customize and extend the default styles, tailoring the design to your specific project requirements.
- Responsive Design: Tailwind CSS includes responsive classes, enabling you to create adaptive layouts that work seamlessly on various screen sizes and devices.

3. KEY FEATURES OF THE APP

Laravel-based file sharing application is designed to offer a comprehensive set of features to meet the diverse needs of users and facilitate efficient file sharing and management. These features ensure that the application stands out in terms of usability, security, and collaboration.

Here are the core features:

- **User Registration:** Allow users to create accounts by providing essential information, including username, email address, and password.
- **File Uploading:** Enable registered users to upload various types of files, such as images, documents, audio, and video files.
- **File Downloading:** Provide a secure and user-friendly mechanism for downloading files, ensuring data integrity and access control.
- **File Editing:** Allow users to make modifications to the files they have uploaded, promoting collaboration and updates.
- **File Deleting:** Enable users to manage their uploaded content by allowing them to delete files as needed.
- **File Searching:** Implement a powerful search feature that lets users quickly find specific files based on various criteria, including file names and types.
- **User's Uploaded Files Page:** Create a dedicated user page where users can view and manage the files they have uploaded, offering easy access to their content.
- **Recently Uploaded Files Page:** Display a "Latest" section that showcases the most recently uploaded files, helping users discover and access new content.
- **File's Information Page:** Each file has a detailed information page, providing metadata such as file type, size, upload date, and comments from other users, enhancing file organization and information retrieval.
- **Preview for Image Files:** Offer an in-app image preview feature, allowing users to view image files without the need to download them separately.
- **Players for Audio and Video Files:** Incorporate built-in players for audio and video files, enabling users to listen to or watch content directly within the application, enhancing the user experience.

- **Nested Comments:** Implement nested comments, allowing users to leave comments and engage in discussions on files, fostering interaction and collaboration.
- **Adaptive Design:** Develop the application with a responsive and adaptive design, ensuring an optimal user experience on various devices, including desktops, tablets, and smartphones.
- **Flash Messages:** Use flash messages to provide real-time feedback to users about their actions within the application, ensuring a smooth and interactive experience.

These features collectively create a versatile and user-centric file sharing application, addressing the primary concerns of security, accessibility, and collaboration. The application is designed to meet the needs of a wide range of users, from individuals looking to share personal files to businesses and organizations requiring a secure and efficient platform for file management and collaboration.

4. MODULES

This section will outline key modules of our Laravel-based file sharing application, accompanied by code snippets and explanations.

Project Structure:

- Inside *app>Http>Controllers*, the *FileController.php* deals with uploading a file, displaying files, deleting files and displaying comments. While *UserController.php* deals with user registration and login.
- Inside *resources>views>components (php artisan make:component Name)*, The *layout.blade.php* file defines the layout of the app. It contains navbar, footer and other components. This layout component class can be used by other blade.php files via blade component tags *<x-layout> </x-layout>*

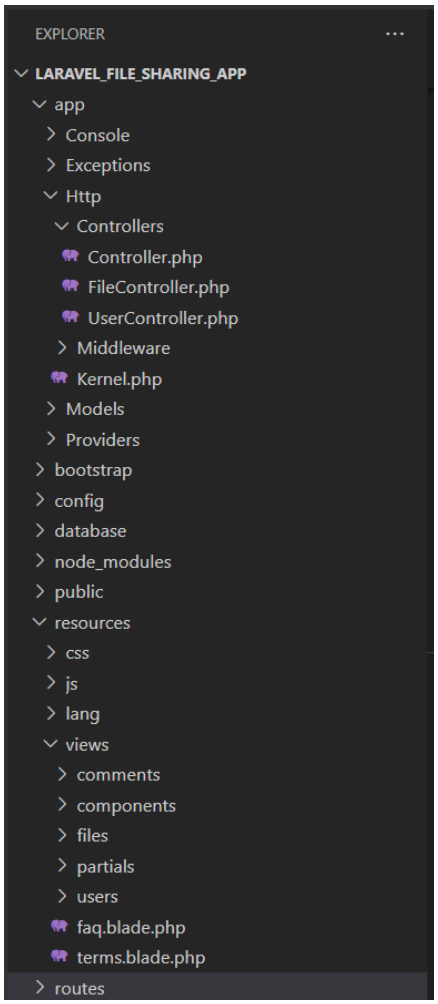

Figure 2: Project directory structure as seen in VS Code editor

4.1 USER REGISTRATION

```
public function register(Request $request)
{
    // Validation of user input
    $validatedData = $request->validate([
        'name' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:6|confirmed',
    ]);

    // Create a new user
    $user = User::create([
        'name' => $validatedData['name'],
        'email' => $validatedData['email'],
        'password' => Hash::make($validatedData['password']),
    ]);

    // Authentication and login
    Auth::login($user);

    // Redirect to the user's dashboard
    return redirect('/dashboard');
}
```

Explanation: This module handles user registration. It starts by validating user input to ensure it meets the required criteria. If the input is valid, it proceeds to create a new user, storing their name, email, and password hashed for security. The user is then authenticated and logged in, and they are redirected to their dashboard.

4.2 USER LOGIN

```
public function login(Request $request)
{
    // Validate user input
    $credentials = $request->validate([
        'email' => 'required|email',
        'password' => 'required',
    ]);

    // Attempt to authenticate the user
    if (Auth::attempt($credentials)) {
        // Authentication successful, redirect to the user's dashboard
        return redirect('/dashboard');
    } else {
        // Authentication failed, redirect back with an error message
        return back()->with('error', 'Invalid login credentials');
    }
}
```

Explanation:

- The login function handles user login by first validating the user's input (email and password) using Laravel's built-in validation methods.
- It then uses Laravel's Auth::attempt() method to attempt user authentication. If the provided credentials are correct, the user is logged in.

- If authentication is successful, the user is redirected to the dashboard. Otherwise, if authentication fails, the user is redirected back to the login page with an error message.

4.3 FILE UPLOAD

```
public function uploadFile(Request $request)
{
    $request->validate([
        'file' => 'required|file|mimes:jpeg,png,mp3,mp4,pdf',
    ]);

    $file = $request->file('file');
    $fileName = time() . '_' . $file->getClientOriginalName();
    $file->storeAs('uploads', $fileName);

    // Save file information to the database
    File::create([
        'user_id' => auth()->id(),
        'name' => $fileName,
        'type' => $file->getMimeType(),
        'size' => $file->getSize(),
    ]);

    return redirect('/dashboard')->with('success', 'File uploaded successfully.');
}
```

Explanation:

- The uploadFile function handles the file uploading process. It begins by validating the uploaded file to ensure it meets the specified criteria (file type, etc.).
- After validation, it generates a unique filename based on the current timestamp and the original filename to avoid conflicts.
- The file is then stored in the 'uploads' directory, which should be configured in Laravel's file storage settings.
- The file's information, including the user who uploaded it, filename, type, and size, is saved to the database using Laravel's Eloquent ORM.
- Finally, the user is redirected to their dashboard with a success message.

4.4 FILE MODIFICATION

```
public function editFile(Request $request, $fileId)
{
    // Validate and authorize the request to edit the file
    $file = File::find($fileId);

    if (!$file) {
        return abort(404);
    }

    if ($file->user_id !== auth()->id()) {
        return abort(403);
    }

    // Process file editing (update file details, etc.)
    // ...

    return redirect('/dashboard')->with('success', 'File edited successfully.');
}
```

Explanation:

- The editFile function handles file editing. It begins by validating and authorizing the request, ensuring that the user has the appropriate permissions to edit the file.
- It retrieves the file based on the provided $fileId, checks if the file exists, and verifies that the user attempting to edit the file is the file's owner.
- Once validation and authorization are successful, the code can be extended to process the file editing (e.g., updating file details, modifying content).
- After editing, the user is redirected to their dashboard with a success message.

5. WEBSITE SNAPSHOTS

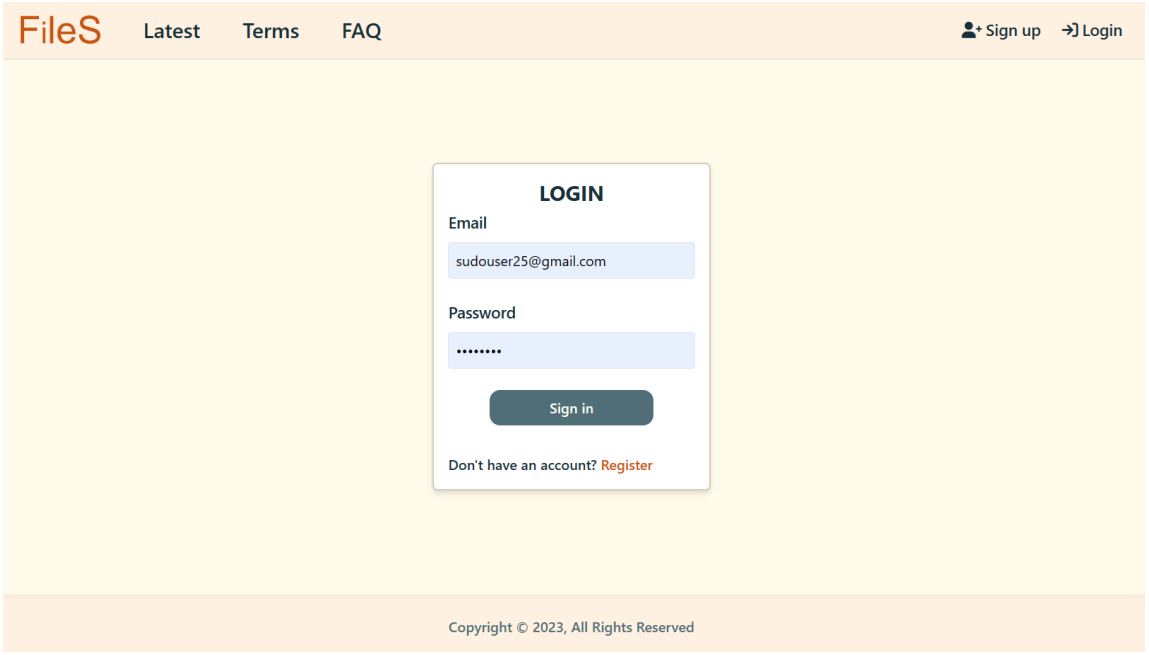

Figure 3: Login page

Figure 4: Uploading a file

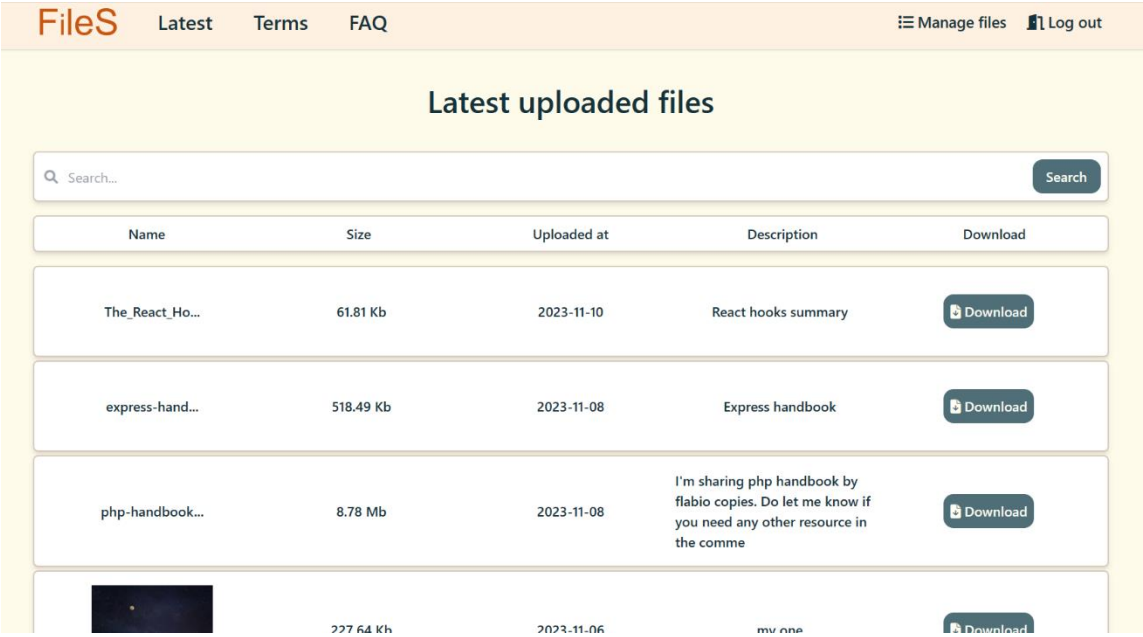

Figure 5: Latest uploaded files page

Figure 6: Manage files page

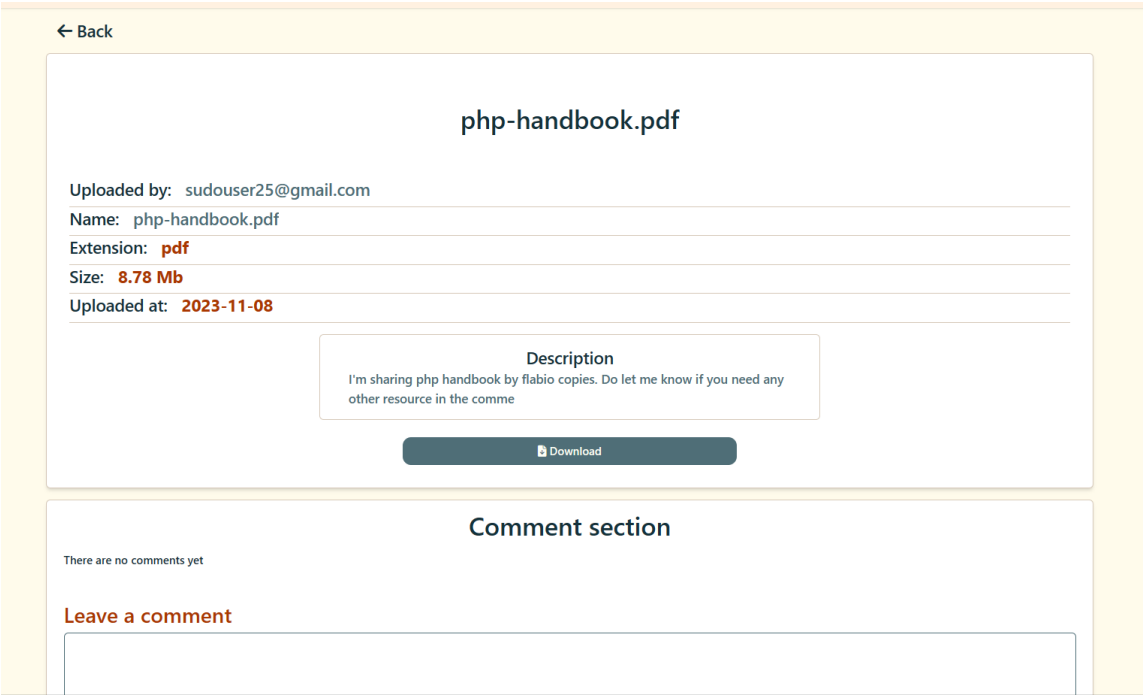

Figure 7: File info. page

6. GITHUB LINK

- The link to the GitHub repository is: https://github.com/Rmariner25/file_sharing_app

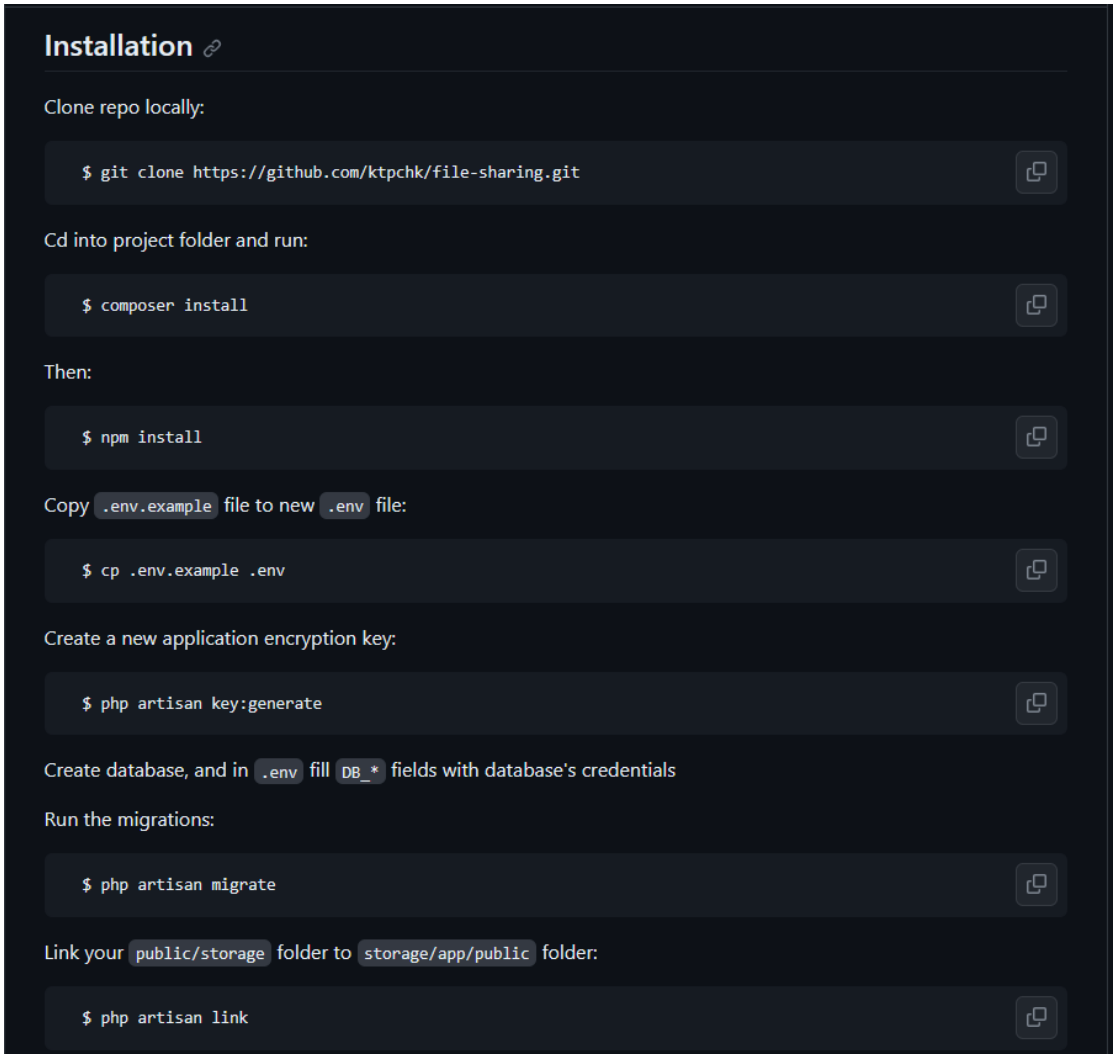

Figure 8: Steps for installation

7. LIST OF REFERENCES

- PHP Documentation: https://www.php.net/docs.php
- Laravel Documentation: https://laravel.com/docs/8.x
- MySQL Documentation: https://dev.mysql.com/doc/
- Tailwind CSS Documentation: https://tailwindcss.com/docs
- Laravel Flash Messages Tutorial: https://www.codechief.org/article/laravel-flash-messages-with-example
- Laravel File Upload Tutorial: https://www.positronx.io/laravel-file-upload-with-validation/
- Laravel Nested Comments Tutorial: https://www.itsolutionstuff.com/post/laravel-8-nested-comment-system-tutorialexample.html