**Image Hosting Document**

**Backend Project Documentation**

1. **First Step**

* Make a new laravel Project.
* Type => (**new laravel project\_name**).
* The new laravel project will be created in your system after few minutes.
* Then open the project and start working on it.

1. **Second Step**

* Link the project with github so we can upload the changes that we make changes repeatedly and save it online as a backup.
* GitHub is a confirmation that our code exists as a backup if we need it again if something goes wrong.

1. **Third Step**

* Get Done with the database.
* Install the setup for the database if not installed.
* After Installation of Database which will **MongoDB,** open cmd (Command Prompt) and check if the database has installed in our system or not by typing mongo on cmd.
* If some action (code of lines) is performed, then you have successfully installed mongodb else there was some issue in the installation of mongodb.
* Then merge the database with the project so further there will be no problems with the database.
* If there are some issue related to this go, see this MongoDB Documentation and video.
* Video Link: ([**https://www.youtube.com/watch?v=TOBqJa2GWQY/**](https://www.youtube.com/watch?v=TOBqJa2GWQY/)**)**.
* Documentation Link**:** ([**https://docs.mongodb.com/drivers/php/**](https://docs.mongodb.com/drivers/php/)).

1. **Fourth Step**

* Now start developing your project with respect to the requirements.

**-----------------------------------------------------------------------------------------------------**

1. **Signup API**

* **Signup Requirements** (profile picture, name, age, password, email) in **form data**.
* Make a Request file for User Signup Request and validate all inputs that are required.
* Use a model file for User.
* Make a Controller file for the User.
* Make middleware for UserSignup Api.
* Make Service for sending confirmation email.
  + Create **resources/views/testmail.blade.php**
  + Create **Mail/testmail.php**.
  + Make jobs for sending email (queue in backend) **job/SendEmailJob.**
  + Update in **(. env)** file **QUEUE\_CONNECTION=database**
  + Then run command (**php artisan queue:table**).
  + Then run this command (**php artisan migrate**)
  + Run this command in a new cmd/terminal to send email in queue jobs (**php artisan queue:listen**) or (**php artisan queue:work**).
  + Confirm the email is received.
* Make a route for Signup Api.
* Move to next step.

1. **Email Verify API**

* Confirm the email and verify the email.
* Once the email is verified the **entity (email\_verified\_at)** will be updated with date and time.

1. **Login API**

* **Login Requirements** (email, password) in **form data**.
* Make a Request file for User Login Request and validate all inputs that are required.
* Now install JWT Token Services from here ([**https://github.com/firebase/php-jwt/**](https://github.com/firebase/php-jwt/)).
  + **composer require firebase/php-jwt**
* Make a middleware for email verify in login Api.
* If your email is verified, then user will be logged in else user can’t logged in.
* After logging in the token will generated and status will be stetted to 1 in the database.

1. **Forgot Password API**

* **Forgot Password Requirements** (email) in **form data**.
* Make a Request file for User Forgot Password and validate all inputs that are required.
* Email will be generated with a verification token for forgot password.

1. **Change Password API**

* **Change Password Requirements** (email, otp, new password) in **form data**.
* Make a Request file for User Change Password and validate all inputs that are required.
* Verify the otp and Update User Password.

1. **User Update Profile API**

* **User Update Profile Requirements** (jwt\_token, profile picture, name, age, password, email) in **form data**.
* Make a Request file for User Update Profile and validate **only token** input that are required, rest are not required due to some reasons of (null values).
* Update the user credentials which are provided.

1. **Upload Photo Api**

* **Upload Photo Requirements** (jwt\_token, user\_id, photo/image, access (hidden, public, private)) in **form data**.
* Make a Request file for User Upload and validate all inputs that are required.
* Get image in form of encoded base64 and then decode and get credentials (extension and path with http server).
* Photo will be uploaded in databases.
* By default, image access type will be **hidden**.

1. **Delete / Remove Photo Api**

* **Remove Photo Requirements** (jwt\_token, user\_id, photo/image) in **form data**.
* Make a Request file for User Remove and validate all inputs that are required.
* Photo will be remove / deleted from databases.

1. **User Search Api**

* **User Search Requirements** (jwt\_token, user\_id, date, time, image\_name, image\_extension, image\_accessors) in **form data**.
* Make a Request file for User Search Photo and validate all inputs that are required.
* The searched photo can be searched on requirement basis.
* (jwt\_token, user\_id, date, time, image\_name, image\_extension, image\_accessors).

1. **User Make Photo Public Api**

* **User Make Photo Public Requirements** (jwt\_token, user\_id, photo\_id, access(public)) in **form data**.
* Make a Request file for User Search Photo and validate all inputs that are required.
* The photo access type will be converted to **public** from (**hidden / private**).
* The photo be updated in database.

1. **User Make Photo Hidden Again Api**

* **User Make Photo Public Requirements** (jwt\_token, user\_id, photo\_id, access(public)) in **form data**.
* Make a Request file for User Search Photo and validate all inputs that are required.
* The photo access type will be converted to **hidden** from by (**public / private**).
* The photo be updated in database.

1. **User Make Photo Private Api**

* **User Make Photo Private Requirements** (jwt\_token, user\_id, photo\_id, access (private), email (of users to access private photo)) in **form data**.
* Make a Request file for User Search Photo and validate all inputs that are required.
* The photo access type will be converted to **private** from by (**hidden / public**).
* The photo be updated in database.

1. **User Remove Email Access of Private Photo Api**

* **User Remove Email Access of Private Photo Requirements** (jwt\_token, user\_id, photo\_id, access (private), email (of user to remove access private photo)) in **form data**.
* Make a Request file for User Search Photo and validate all inputs that are required.
* The email will be removed for private photo access.
* The photo be updated in database.

1. **User Generate Link of Photo Api**

* **User Generate Link of Photo Requirements** (jwt\_token, photo\_id) in **form data**.
* Make a Request file for User Search Photo and validate all inputs that are required.
* The user\_id and photo access type will be get from the backed to match.
* Then the link will be generated.

1. **User Show Generated Link of Photo Api**

* **User Show Generate Link of Photo Requirements** (jwt\_token, photo\_link) in **form data**.
* Make a Request file for User Search Photo and validate all inputs that are required.
* All conditions will be checked by the access type (**hidden, public, private**) on that the picture will be viewed.

1. **User Logout Api**

* **User Logout Requirements** (jwt\_token) in **form data**.
* User will be logout and the token and status will set to null and 0.

**-----------------------------------------------------------------------------------------------------**

1. **Fifth Step**

* Now upload all your Api’s on ( <https://www.heroku.com/> ).

1. **Sixth Step**

* Now start integrating your Api’s with the fornt-end code one by one.
* Then your project will be completed.

**-----------------------------------------------------------------------------------------------------**