These are my api endpoint

*Route*::get('/get-random-users', [*GlobalController*::class, 'getRandomUsers']);

*Route*::get('/home', [*HomeController*::class, 'index']);

*Route*::middleware(['auth:sanctum'])->group(function () {

*Route*::get('/logged-in-user', [*UserController*::class, 'loggedInUser']);

*Route*::post('/update-user-image', [*UserController*::class, 'updateUserImage']);

*Route*::patch('/update-user', [*UserController*::class, 'updateUser']);

*Route*::get('/posts/{id}', [*PostController*::class, 'show']);

*Route*::post('/posts', [*PostController*::class, 'store']);

*Route*::delete('/posts/{id}', [*PostController*::class, 'destroy']);

*Route*::get('/profiles/{id}', [*ProfileController*::class, 'show']);

*Route*::post('/comments', [*CommentController*::class, 'store']);

*Route*::delete('/comments/{id}', [*CommentController*::class, 'destroy']);

*Route*::post('/likes', [*LikeController*::class, 'store']);

*Route*::delete('/likes/{id}', [*LikeController*::class, 'destroy']);

});

To Store A Post

public function store(*Request* $request)

{

$request->validate([

'video' => 'required|mimes:mp4,mov,webm',

'text' => 'required'

]);

try {

$post = new *Post*;

$post = (new *FileService*)->addVideo($post, $request);

$post->user\_id = auth()->user()->id;

$post->text = $request->input('text');

$post->save();

return response()->json(['success' => 'OK'], 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

*FileService Class have a function addVideo*

public function addVideo($model, $request)

{

$video = $request->file('video');

$extension = $video->getClientOriginalExtension();

$name = time() . '.' . $extension;

$video->move(public\_path() . '/files/', $name);

$model->video = '/files/' . $name;

return $model;

}

To Show a particular Post

public function show($id)

{

try {

$post = *Post*::where('id', $id)->get();

$posts = *Post*::where('user\_id', $post[0]->user\_id)->get();

$ids = $posts->map(function ($post) {

return $post->id;

});

return response()->json([

'post' => new *AllPostsCollection*($post),

'ids' => $ids

], 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

For the Json response

Inside the *AllPostsCollection*($post) class, I have a function toArray

public function toArray(*Request* $request)

{

return *$this*->collection->map(function ($post) {

return [

'id' => $post->id,

'text' => $post->text,

'video' => url('/') . $post->video,

'created\_at' => $post->created\_at->format(' M d Y'),

'comments' => $post->comments->map(function ($comment) {

return [

'id' => $comment->id,

'text' => $comment->text,

'created\_at' => $comment->created\_at->format(' M d Y'),

'user' => [

'id' => $comment->user->id,

'name' => $comment->user->name,

'image' => url('/') . $comment->user->image

],

];

}),

'likes' => $post->likes->map(function ($like) {

return [

'id' => $like->id,

'user\_id' => $like->user\_id,

'post\_id' => $like->post\_id

];

}),

'user' => [

'id' => $post->user->id,

'name' => $post->user->name,

'image' => url('/') . $post->user->image

]

];

});

}

ProfilePage

public function show($id)

{

try {

$posts = *Post*::where('user\_id', $id)->orderBy('created\_at', 'desc')->get();

$user = *User*::where('id', $id)->get();

return response()->json([

'posts' => new *AllPostsCollection*($posts),

'user' => new *UsersCollection*($user)

], 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

It returns all the user’s post and UserCollection(Data)

This is the toArray of the UserCollection

public function toArray(*Request* $request)

{

return *$this*->collection->map(function ($user){

return [

'id' => $user->id,

'name' => $user->name,

'bio' => $user->bio,

'image' => url('/') . $user->image,

'created\_at' => $user->created\_at,

'updated\_at' => $user->updated\_at,

];

});

}

**Like A Post**

To like a post is simple insert but return anarray **like[id, post\_id, user\_id]**

public function store(*Request* $request)

{

$request->validate(['post\_id' => 'required']);

try {

$like = new *Like*;

$like->user\_id = auth()->user()->id;

$like->post\_id = $request->input('post\_id');

$like->save();

return response()->json([

'like' => [

'id' => $like->id,

'post\_id' => $like->post\_id,

'user\_id' => $like->user\_id

],

'success' => 'OK'

], 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

**UnLike A Post**

To unlike a post is simple delete operation but return anarray **like[id, post\_id, user\_id]**

public function destroy($id)

{

try {

$like = *Like*::find($id);

if (count(collect($like)) > 0) {

$like->delete();

}

return response()->json([

'like' => [

'id' => $like->id,

'post\_id' => $like->post\_id,

'user\_id' => $like->user\_id

],

'success' => 'OK'

], 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

**Comment**

**Add A comment is a simple insert operation**

public function store(*Request* $request)

{

$request->validate([

'post\_id' => 'required',

'comment' => 'required',

]);

try {

$comment = new *Comment*;

$comment->post\_id = $request->input('post\_id');

$comment->user\_id = auth()->user()->id;

$comment->text = $request->input('comment');

$comment->save();

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

**These are other function**

public function loggedInUser()

{

try {

$user = *User*::where('id', auth()->user()->id)->get();

return response()->json(new *UsersCollection*($user), 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

/\*\*

\* Store a newly created resource in storage.

\*/

public function updateUserImage(*Request* $request)

{

$request->validate(['image' => 'required|mimes:png,jpg,jpeg']);

if ($request->height === '' || $request->width === '' || $request->top === '' || $request->left === '') {

return response()->json(['error' => 'The dimensions are incomplete'], 400);

}

try {

$user = (new *FileService*)->updateImage(auth()->user(), $request);

$user->save();

return response()->json(['success' => 'OK'], 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

/\*\*

\* Display the specified resource.

\*/

public function getUser($id)

{

try {

$user = *User*::findOrFail($id);

return response()->json([

'success' => 'OK',

'user' => $user,

], 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}

/\*\*

\* Update the specified resource in storage.

\*/

public function updateUser(*Request* $request)

{

$request->validate(['name' => 'required']);

try {

$user = *User*::findOrFail(auth()->user()->id);

$user->name = $request->input('name');

$user->bio = $request->input('bio');

$user->save();

return response()->json(['success' => 'OK'], 200);

} catch (*\Exception* $e) {

return response()->json(['error' => $e->getMessage()], 400);

}

}