How I start this project from zero.

Laravel:

1. Install Laravel 5 and Composer. Doc: <https://laravel.com/docs/5.5>, <https://getcomposer.org/>
2. Run command line/terminal “laravel new Amcisa-Laravel”.
3. Add the code below into Amcisa-Laravel/app/Providers/AppServiceProvider.php, register() function.

$this->app->bind('path.public', function() {

return base\_path('public\_html');

});

Reference: <https://stackoverflow.com/questions/30198669/how-to-change-public-folder-to-public-html-in-laravel-5>

1. Rename the Amcisa-Laravel/public to Amcisa-Laravel/public\_html
2. This is because the cPanel default root folder (“entry point”) is public\_html.
3. Download and install MAMP. Doc: <https://www.mamp.info/en/>
4. Run MAMP, set your Web Server Document Root to “<path\_of\_your\_project>\Amcisa\Amcisa-Laravel\public\_html”
5. Start Servers, and you can access your Laravel website from browser on http://localhost.

Vue.Js:

1. Install NodeJs. Doc: <https://nodejs.org/en/>
2. Run command line/terminal “npm install –global vue-cli”. This command is to install vue-cli, it is vue command line interface.
3. Run command line/terminal “vue init webpack Amcisa-Vue”. This command creates a new vue project named Amcisa-Vue using “webpack template” ( Reference: <https://github.com/vuejs/vue-cli>). It will ask some questions, just hit enter for all of them, except the project name might not allow capital latter, just enter “amcisa-vue” for project name.
4. Run command line/terminal “cd Amcisa-Vue”, then “npm run dev”. “cd Amcisa-Vue” change your directory into vue project folder. “npm run dev” host your Vue.js application on localhost:8080.
5. Enter URL <http://localhost:8080>, you can access your Vue.js application from browser now.

Deploy Vue.Js to Laravel

\*We use Laravel as backend, Vue.js as frontend. We copy distribution version of Vue.js to Laravel project. The concept is: MAMP server hosting Laravel, when you access the Laravel from browser, it return the vue.js views.

1. Run command line/terminal “npm run build”, this command build distribution files under dist/ folder.
2. Run the Deploy.bat in Amcisa/ folder, this file do the following jobs:
   1. Copy Amcisa-Vue\dist\index.html to Amcisa-Laravel\resources\views\welcome.blade.php
   2. Delete Amcisa-Laravel\public\_html\static\
   3. Copy Amcisa-Vue\dist\static\ to Amcisa-Laravel\public\_html\static
3. You can see your Vue.js project in MAMP server.

Elements in Laravel

Communicate with database

1. Edit DB\_DATABASE, DB\_USERNAME, DB\_PASSWORD in .env file.
2. Run command line/terminal: ”php artisan make:model <model\_name>”. This create a model named <model\_name>, **the <model\_name> must similar or same with the table name**, go here for more details : <https://laravel.com/docs/5.5/eloquent>.
3. The above 2 steps allow you to access table named <model\_name> in database < DB\_DATABASE >.
4. If you don’t want to use time\_stamps (your table don’t have “updated\_at”, “created\_at” columns), add “public $timestamps = false;” to your Model class in /app/<model\_name>

User Registeration

1. Create a table named “users”
2. Add columns “email”, “name”, “password”, “id”(For JWT-auth) and “remember\_token”. Note that must follow this naming, see here for more information: <https://laravel.com/docs/4.2/security>
3. If you don’t want to use time\_stamps (your table don’t have “updated\_at”, “created\_at” columns), add “public $timestamps = false;” to your User model class in /app/User.php
4. Create UserController.php inside /app/Http/Controllers/ folder.
5. Add this code to UserController.php:
6. **<?php  
   namespace** App\Http\Controllers;  
   **use** App\User;  
   **use** Illuminate\Http\Request;  
   **class** UserController **extends** Controller{  
    **public function** register(Request $request)  
    {  
    $this->validate($request, [  
    'name' => 'required',  
    'email' => 'required|email|unique:users',  
    'password' => 'required'  
    ]);  
    $user = **new** User([  
    'name' => $request->input('name'),  
    'email' => $request->input('email'),  
    'password' => bcrypt($request->input('pwassword'))  
    ]);  
    $user->save();  
    **return** response()->json([  
    'message' => 'Successfully register'  
    ], 201);  
    }  
   }
7. Add this code to routes/api.php
8. Route::*post*('/user/register',[  
    'uses' => 'UserController@register'  
   ]);
9. Now the Laravel backend is ready for handle POST method to register user. Note that this two headers: “Content-Type”: ”application/json” and “X-Requested-With”: “XMLHttpRequest”, must be included in your post request. The Content-Type tell the Laravel the request in json format. The X-Requested-With make the Laravel able to do validation. This is the validation: $this->validate($request, [  
    'name' => 'required',  
    'email' => 'required|email|unique:users',  
    'password' => 'required'  
   ]);

Authentication

Install Passport

Doc: <https://laravel.com/docs/5.5/passport>

1. Run command line “composer require laravel/passport”
2. Add the following code to config/app.php providers array.

Laravel\Passport\PassportServiceProvider::*class*;

1. Run command line “php artisan migrate”, this command add these 5 tables into your database:

Migrated: 2016\_06\_01\_000001\_create\_oauth\_auth\_codes\_table

Migrated: 2016\_06\_01\_000002\_create\_oauth\_access\_tokens\_table

Migrated: 2016\_06\_01\_000003\_create\_oauth\_refresh\_tokens\_table

Migrated: 2016\_06\_01\_000004\_create\_oauth\_clients\_table

Migrated: 2016\_06\_01\_000005\_create\_oauth\_personal\_access\_clients\_table

(If you see “migrations” table in your database, this table is the record of migrated tables)

(These 5 table used to store the clients data for OAuth2.0, clients is used to generate access token in OAuth2.0 architecture. Refer to OAuth2.0 architecture for more details)

1. Run command “php artisan passport:install”
2. Add the code(trait) below into App/User.php

<?php

namespace App;

use Laravel\Passport\HasApiTokens;

use Illuminate\Notifications\Notifiable;

use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable

{

use HasApiTokens, Notifiable;

}

1. Add Passport::routes(); into app/Providers/AuthServiceProvider.php boot() function.
2. Finally, in your config/auth.php configuration file, you should set the driver option of the api authentication guard to passport. This will instruct your application to use Passport's TokenGuard when authenticating incoming API requests:

'guards' => [

'web' => [

'driver' => 'session',

'provider' => 'users',

],

'api' => [

'driver' => 'passport',

'provider' => 'users',

],

],

1. Run command line “php artisan passport:keys”
2. Add the following code to app/User.php

//we use email\_school for the passport user name.

**public function** findForPassport($username) {  
 **return** $this->where('email\_school', $username)->first();  
}

passport requires ‘email’, ‘id’, ‘password’ and ‘remember\_token’ column in your users table. The code above allow password to use ‘email\_school’ instead of ‘email’.

Now, I assumed you have basic knowledge about Laravel. The tutorial below will not be step-by-step.

Role

1. Our users got these roles: admin, president, member.
2. Create a ‘roles’ table with ‘user\_id’ and ‘role’ column.
3. Create a model for ‘roles’ table.
4. The ‘user\_id’ colume of roles table it the foreign key of user table, thus, add the following code to User model.

**public function** roles(){  
 **return** $this->hasOne('App\Role');  
}

Doc: https://laravel.com/docs/master/eloquent-relationships#one-to-one

1. Create middleware ‘CheckRole’.
2. Add the following code to app/Http/Middleware/CheckRole.php

**public function** handle($request, Closure $next, $role)  
{  
 **if** ($request->user()->roles->role != $role) {  
 **return** response("Insufficient permission",401);  
 }  
 **return** $next($request);  
}

1. Register the middleware by add the following code to app/Http/Kernal.php $routeMiddleware

'role' => \App\Http\Middleware\CheckRole::*class*

1. Now, you can use your middleware in api.php. For example,

Add 'middleware' => ['auth:api','role:presidentt'] to your route. The ‘president’ after the colon will be the variable pass to (step6) $role in handle() function. Doc: https://laravel.com/docs/master/middleware#middleware-parameters

Store uploaded file

**public function** upload(Request $request)  
{  
 $request->file('file')->move('uploaded\_files', 'aaa.jpg');  
 **return** response()->json("success uploaded",201);  
}

Elements in Vue.js frontend

Sent HTTP Request

1. Doc: <https://github.com/axios/axios>

Send file with HTTP Post Request

1. <template>  
    <div class="title">  
    <input type="file" id="file" ref="file" v-on:change="handleFileUpload()"/>  
    </label>  
    <button v-on:click="submitFile()">Submit</button>  
    </div>  
   </template>  
     
   <script>  
   **import** axios **from** 'axios';  
     
   **export default** {  
    name: 'Home',  
    data () {  
    **return** {  
    file: ''  
    }  
    },  
    methods: {  
    submitFile () {  
    **var** formData = **new** FormData()  
    formData.append('file', **this**.file)

formData.append('extension', 'jpeg')  
 axios.post('/api/upload',  
 formData,  
 {  
 headers: {  
 'Content-Type': 'multipart/form-data'  
 }  
 }  
 ).then(**function** (response) {  
 console.log(response)  
 })  
 .catch(**function** () {  
 console.log(failed to upload')  
 })  
 },  
 handleFileUpload () {  
 **this**.file = **this**.$refs.file.files[0]  
 }  
 }  
}  
</script>

Database

For amcisa.org database, “email\_school” is unique and cannot be NULL.