**Steps Taken for Project**

Installed PHP

Installed Composer – A Dependency Manager for PHP

Installed Laravel base project using Composer’s Create-Project e.g. =

composer create-project --prefer-dist laravel/laravel FinerVision-TechTest

**Use PHP’s local built-in development server =**

php artisan serve

starts a server at <http://localhost:8000>:

**Changing the base pre-set of a project.**

When creating a project, Laravel creates a basic starting point using Bootstrap, React/Vue. Initially downloaded Laravel/ui composer package.

composer require laravel/ui:^2.4

Once installed wanted to replace base Vue framework with React implementation

php artisan ui react

**Install front end dependencies**

Now we need to install the projects front end dependencies using NPM.

npm install

Once the dependencies have been installed using npm install, you can compile your SASS files to plain CSS using [Laravel Mix](https://laravel.com/docs/7.x/mix#working-with-stylesheets). The npm run dev command will process the instructions in your webpack.mix.js file. Typically, your compiled CSS will be placed in the public/css directory:

npm run dev

Now to run the server again we just run

php artisan serve

**Which Page Controls Home Page.**

Shows us the Laravel pre-generated front page. To remove this go in to

resources\views\welcome.blade.php

and remove everything in the body and any unnecessary styling.

Added CSS in the head of the project.

 <link href="/css/app.css" rel="stylesheet">

Added JS in the body of the project at the bottom.

<script src="/js/app.js"> </script>

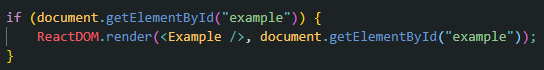
**Displaying a Component page.**

To display another page we need to create a div with the same ID we have specified in our example component.

To find the example component we look in

resources\js\components\Example.js

and we see this code.

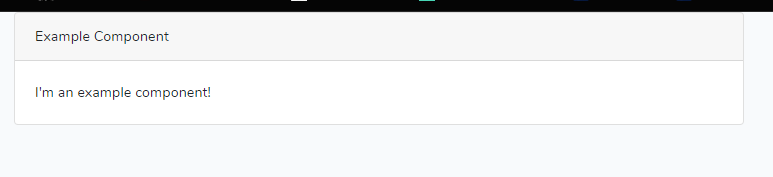


This searches for a React Component of that ID and renders it to the page.

<div id='example'> </div>

This is the code inside the body of the welcome.blade.php

And this is the rendered component inside our local server at 127.0.0.1:8000



Next changed name of file from Example to “UserInputForm”. To get this to display I also needed to change the class and CSS name.

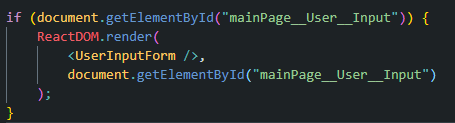
Also changed default export



Then in App.js we also had to change the require to match.

.

Also changed div name and react find name.

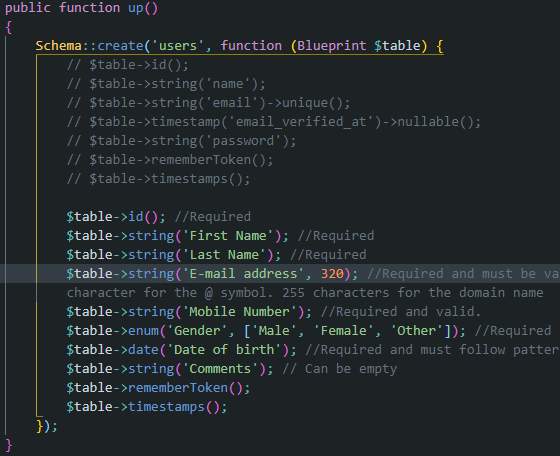


.

**Creating a database table through migration and connecting to database.**

Changed all relevant variables in .env and database.php to be able to migrate to database.

Following basic layout and documentation created a basic table with all the values we need for the table.



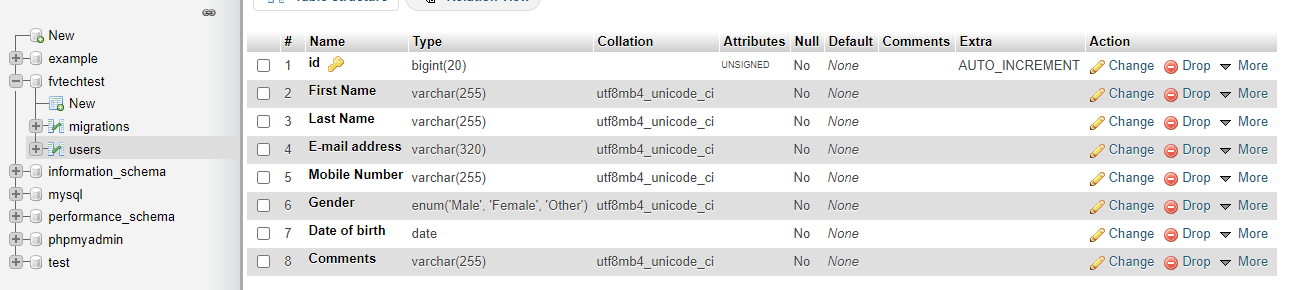
To migrate table called php artisan migrate

Received error(Couldn’t find driver).

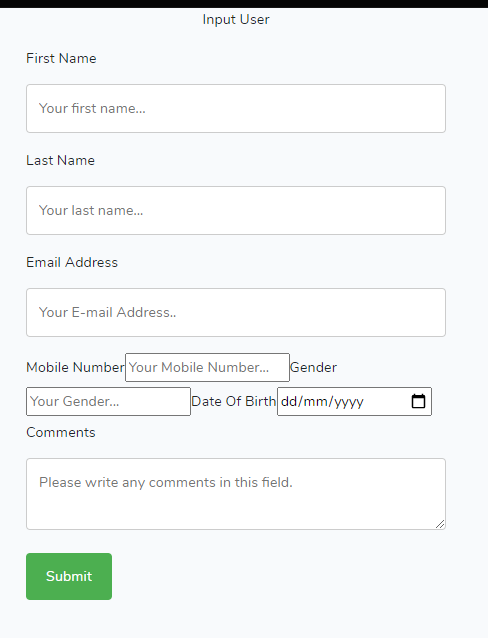
To fix had to uncomment an extension on php.ini file at the PHP base folder.

Then called php artisan migrate to migrate the one we already prefilled. If need to create new tables could call php artisan make:migration MIGRATION NAME.

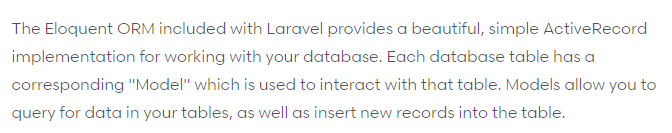
This successfully created a table to our mysql database.



***Next populated react component with basic layout of a user form.***

.

**Next create a Model for our database.**



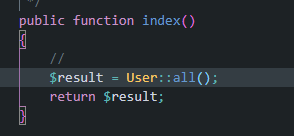
**Next create a Controller for the database. (might not be needed)**

Controller allows us to have specific functions which work on the data. The basic controller file has CRUD operations.

Made using the command php artisan make:controller Api/UserController –resource

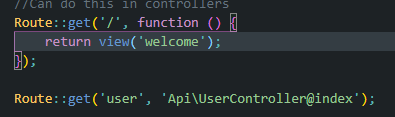


In index we choose to return all user results.



To access this call we need to go to the web.php file and define a route for it as well as tell the controller which model we are using.





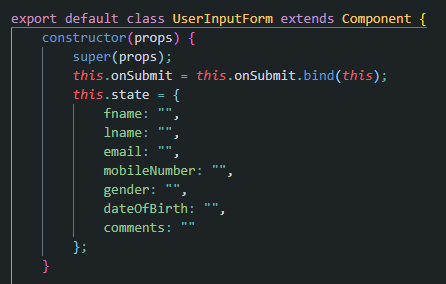
First part of the get function is the url e.g. localhost:8000/user and the second part is the path it takes. In this case it goes to our api/usercontroller and calls the index function. This function then returns all users in the database.



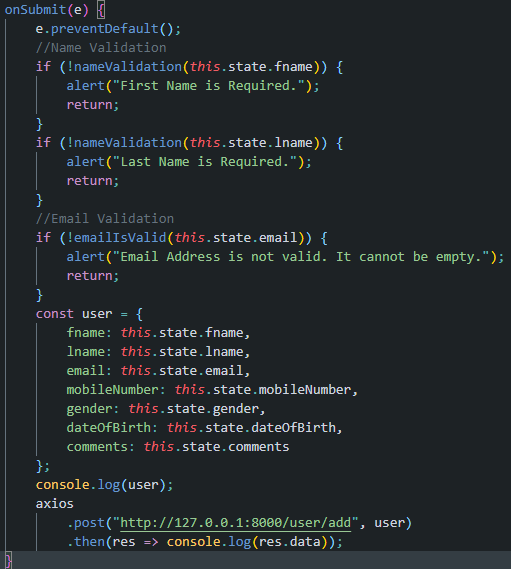
We don’t need this for technical test, but It’s good to understand how it works as we will need to use something similar to POST data(I presume).

**POST data to the database.**

Set a state in our component.



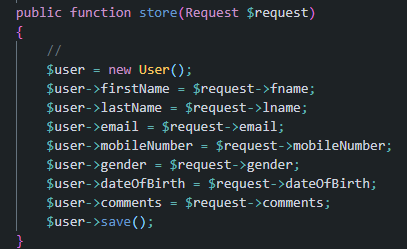
Have a on submit functions which maps state to a user variable and posts to the database url.



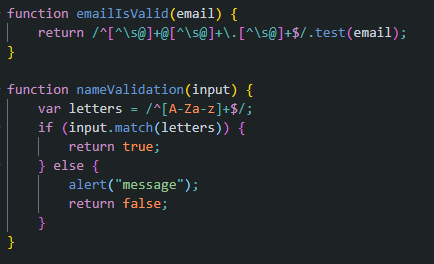
The database url is mapped in web.php. as post as we need to send data.



This api matches the store function inside our controller. In which we create a new object and match the data fields to the inputted datafields.



**Input Validation**



Doing input validation through JS as requested.