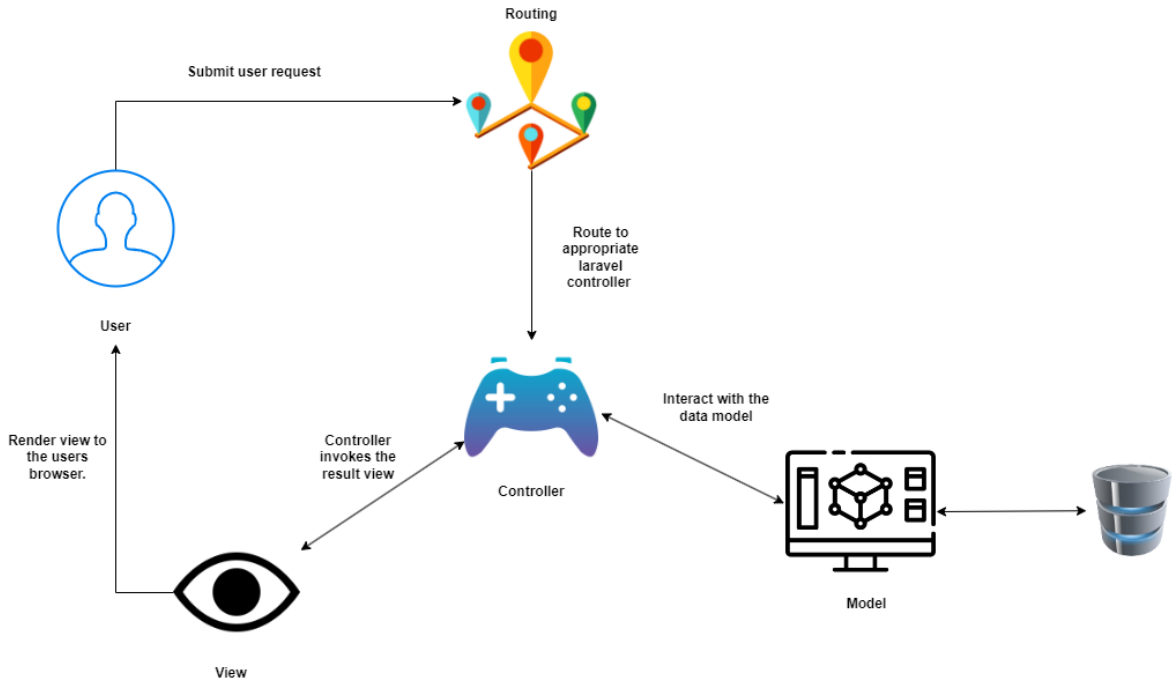
Architecture Diagram:




Framework: Framework comes with rules, ideas and beliefs which are used to solve a particular problem.

Software Framework: Classes, functions, packages and database configurations come in a package. So, we don't need to create or work from scratch.

Laravel Framework:

- PHP web framework
- Open source
- Follows MVC architecture
- Uses Artisan CLI
- OOP
- Built in authentication and authorization
- Eloquent ORM: Class based approach to manage database
- ORM: Object Relational Model
- Template engine: Blade

Laravel is opinionated.

1. Created a project with this command:
   composer create-project laravel/laravel laravel-crud-application
2. To open the application into a web browser:
   php artisan serve

3. Database: phpmyadmin.

MVC:

Contents of controller:

1. Directory: app\Http\Controllers\ApplicantController.php
2. namespace App\Http\Controllers: controller is placed under this namespace which is the standard namespace for laravel application.
3. Use statement: Imports necessary classes and namespaces at the top of the file. Here, model and some classes are imported.
   Model: App\Models\Applicant and classes: Request, Str, Hash, Auth
4. Index method: accepts a Request object as a parameter, which represents the incoming HTTP request.
5. The $search variable will hold the search query provided in the URL as a query parameter q.
6. The next block of code uses the search query to fetch applicants from the database based on the provided query. If a search query is provided ( not empty), the where clause is used to find applicants whose name is like the search query (partial match) or whose ID matches exactly the search query. The results are then paginated with a limit of 5 applicants per page.

```
7.          if ($search != ""){
8.              $applicants = Applicant::where('name', 'LIKE', '%' .
    $search . '%')
9.                                      ->orWhere('id', $search)
10.                                     ->paginate(5);
11.         }
```

12. If no search query is provided, the else block is executed. In this case, all applicants are fetched from the database in descending order (latest first) and paginated with a limit of 5 applicants per page.
13. Finally, the method returns the `applicants.index` view, passing the fetched applicants and the search query to the view using the `compact` function.
14. The `compact` function is used in PHP to create an associative array from variables. It takes a list of variable names as arguments and creates an array where the variable names become the keys, and the variable values become the corresponding values in the array.

In the context of the Laravel framework and the provided code, the `compact` function is used to pass variables from the controller to the view.

```
    'applicants'    $applicants
    'search'    $search
```

This array is then passed to the `view` function, which loads the 'applicants.index' view file and makes the data accessible within the view. In the view, you can directly use the variable names as if they were defined in the view itself.

17. The 'applicants.create' passed to the view function represents the path to the view file. In Laravel's convention, the view files are stored in the resources/views directory. So, 'applicants.create' refers to the view file named create.blade.php located in the resources/views/applicants directory.
18. Store function:
    The $request->validate() method is used to validate the incoming form data before saving it to the database. In this case, the validate() method checks if the required fields (name, mobile_number, previous_institution, and date_of_birth) are present, and if the mobile_number is 11 digits long and unique in the applicants table. If the validation fails, Laravel will automatically redirect back to the form with error messages.
19. The Applicant::create() method is used to create a new record in the applicants table in the database. In this case, the function creates a new applicant record with the provided form data and the generated UUID.

After successfully creating the applicant record, the function redirects the user to the 'applicants.index' route, which will display the list of applicants (index page). The ->with('success', 'Applicant created successfully.')

20. Edit function: The edit method takes an $applicant parameter, which represents the specific applicant record to be edited. Laravel's route model binding automatically fetches the corresponding applicant record from the database based on the URL parameter (usually the id of the applicant).
    a. The function returns a view called 'applicants.edit' and passes the $applicant variable to the view using the compact() function. This allows the edit view to access the data of the specific applicant being edited.
    b. The edit view will be responsible for displaying a form to edit the applicant's information, pre-populated with the existing data.
21. Update:
    a. The update method also takes an $applicant parameter, which represents the specific applicant record to be updated. Like in the edit method, Laravel's route model binding fetches the corresponding applicant record from the database based on the URL parameter.
    b. The function first validates the incoming form data using $request->validate(). It ensures that the required fields (name, mobile_number, previous_institution, and

date_of_birth) are present and meet the specified validation rules (e.g., the mobile number must be a string and 11 digits long).

c. If the form data passes validation, the function proceeds to update the attributes of the specific $applicant instance in the database. The update() method is called on the $applicant instance, passing an associative array with the updated attribute values from the form.
d. After successfully updating the applicant record, the function redirects the user to the list of applicants ('applicants.index' route) with a success message, which can be displayed on the next page.

22. Delete: The method then calls the delete method on the $applicant instance, which deletes the applicant record from the database.
23. Search: The search method is used to perform a search operation on the applicants based on the provided query. It takes a Request object as a parameter, which contains the search query passed through the form submission.
    a. The method retrieves the search query from the request using $request->input('q') and assigns it to the $query variable. It then uses the Eloquent query builder to search for applicants whose name matches the query (using the LIKE operator to find partial matches) or whose id matches the query.
    b. The get() method executes the query and returns a collection of matching applicants

Model:

1. Directory: app\Models\Applicant.php
2. Applicant.php: in the database, there is a table named applicants, it represents that.
3. use HasFactory;: This line of code indicates that the Applicant model uses the Laravel Eloquent "Factory" feature. Factories are used to generate dummy data for testing and seeding the database.
4. protected $fillable: This property specifies the fields that are allowed to be assigned during model creation using the create method or by using the fill method followed by the save method. In this case, the name, mobile_number, previous_institution, and date_of_birth fields can be mass-assigned.

By using the boot method, the validate method is automatically triggered when saving an Applicant model, ensuring that the model data adheres to the defined validation rules before it is persisted in the database.

View:
Contains different view files, mainly for the frontend part.

Database:

1. Directory database/migration: handles the database.
2. Factories and seeds helps to create fake data
3. Route::get('/applicants/search', 'ApplicantController@search')->name('applicants.search');: This route maps the URL path '/applicants/search' to the search method of the ApplicantController. It

also assigns the name 'applicants.search' to this route, which allows us to reference it easily by name in the future. This route handles the search functionality for applicants.

4. Route::resource('applicants', ApplicantController::class);: This line uses the resource method to define multiple routes for the ApplicantController. The resource method automatically generates the standard CRUD routes (index, create, store, show, edit, update, and destroy) for the ApplicantController, making it easier to handle various operations related to applicants.

.env: The .env file is a configuration file in Laravel applications that stores environment-specific settings and configurations. It is located in the root directory of the Laravel project and is not meant to be committed to version control systems like Git.

1. Database name: laravel
2. Table name: applicants