Laravel project

## **Version check**

| Sl no | Package | Syntax | Required version |
|---|---|---|---|
| 1 | Composer | composer | v2 |
| 2 | Node | node -v | V14 above |
| 3 | php | php -v | 7.4 above <=8.0 |
| 4 | mysql | mysql -v | 5.7 > |
| 5 | Laravel | php artisan -v | 6 above<br>(6 php required 7.2 to 8)<br>(9 php required 8 above) |

## **Download Sql**

1. Folder
2. mysql -u root -p
3. mysqldump -u root -p biodiversity_db > biodiversity_db_110923.sql

## **Server hosting**

1. In folder
2. php artisan serve --host=product.cdit.org --port=2005

## **For project creation using Laravel**

For project creation:🧑‍💻

➔ 1.**composer create-project --prefer-dist laravel/laravel projectname "6.*"** (laravel version 6)
➔ **composer create-project laravel/laravel projectname** (laravel version 9)

For auth configuration 🧑‍💻

➔ **composer require laravel/ui**

- *Laravel utilizes Composer to manage its dependencies*
- *Laravel's laravel/ui package provides a quick way to scaffold all of the routes and views you need for authentication using a few simple commands*

➔ **php artisan ui bootstrap --auth** (laravel 9)
**php artisan ui vue --auth** (laravel 6/laravel9)

- *This command should be used on fresh applications and will install a layout view, registration and login views, as well as routes for all authentication end-points.*

https://monovm.com/blog/install-nvm-on-ubuntu/

➔ **npm install && npm run dev**
**npm run build (For laravel 9)**

- *npm run sets the NODE environment variable to the node executable with which npm is executed.*

➔ **.php artisan migrate**

*For migrating all files using auth.*

6.**php artisan serve**

7.**php artisan make: controller auth/LoginController** ( auth folder name)

*If any doubt in comt use **php artisan** command.

8.**create route**
Route::get('/registerhere',[App\Http\Controllers\RegistrationPageController::class,'registerview'])->name('registerhere');

9.**Model controller migration**

| | |
|---|---|
| 1. For creating model with db | :php artisan make:model ModelName -m |
| 2. For creating model | :php artisan make:model Modelname |
| 3.For creating model DB and controller | :php artisan make:model quizresponsesummary -mcr |

10.**For creating middleware** in auth

- Creating auth middleware

```
                1.php artisan make:middleware adminlogin
                2.change adminloginmiddleware
                 *add roleId in users table *use Auth

              public function handle(Request $request, Closure $next)
                  {
                   if(Auth()->check()){
                     if(Auth::user()->roleId==1) {
                                 return $next($request);     }
                         }else{
                            return redirect()->route('home');
                    }
                    }

         3.Change kernel status
              Add routeMiddleware
                     'adminlogin' => \App\Http\Middleware\adminlogin::class,
        4.Add route grouping
Route::group(['middleware'=>['auth','App\Http\Middleware\adminlogin']],function()
{
Route::get('/adminhome',[App\Http\Controllers\adminController::class,'adminhome'])->name('ad
minhome');
});
     5.Change login controller
           // protected $redirectTo = RouteServiceProvider::HOME;
             protected function redirectPath(){
                 return "/adminhome";
             }
```

……………………………………………………………………………………………………………..

- ➔ Php artisan make:middleware XSS

```
         $userInputs=$request->all();
       array_walk_recursive($userInputs, function(&$userInputs, $key){
$userInputs=strip_tags($userInputs,'<a><b><table><tr></td><img><p><u><h1><h2><h3><h4>'
); });
       $request->merge($userInputs);
       return $next($request);
```

**12.To create table in  MySql**

->mysql -u root -p

->Create table

```
php artisan make:migration create_ilmsubcategory_table
```

```
Manually delete

composer dump-autoload
```

**13. Table Migration**

1.Create table: **Php artisan make: migration create_userstypes_table**

2.Take file inside folder migration

3.Edit migration file

```
Example-> Schema::create('users', function (Blueprint $table) {

            $table->id();

            $table->string('name');

            $table->string('email')->unique();

            $table->timestamp('email_verified_at')->nullable();

            $table->string('password');

            $table->rememberToken();

            $table->bigInteger('mobile');

            $table->date('DOB');

            $table->timestamps();

        });
```

4.**set foreign key in migration file**

```
$table->bigInteger('festival_id')->foreign('festival_id')->references
('id')->on('festivals')->onDelete('cascade');
```

**14.Table alteration**

->To add emp_ID to users table

Step1:  php artisan make:migration add_employeeid_to_users
--table="users"

Step2:Take folder

database->migration->(take add table)

```
eg: public function up()

        {

        Schema::table('users', function (Blueprint $table) {

                $table->bigInteger('empID');

        });

        }

        public function down()

        {

        Schema::table('users', function (Blueprint $table) {

                $table->dropColumn('empID');

        });

        }
```

**step3**:php artisan migrate

php artisan migrate
--path=/database/migrations/2022_03_15_044509_create_quizsections_table.php

php artisan make:migration add_description_to_menulinktypes_table --table=menulinktypes

## Method 1: JAVA SCRIPT

Using onclick:
In main page use yield:

```
@yield('customscript')
```

```
<td><button class="btn btn-primary btn-sm"
onClick="status_update()">{{($datas->status==1) ? 'Approve' :  'Not
approved' }}</button></td>
```

Corrs page use @section('')

```
@section('customscript')
<script>

function status_update()
{
   alert("sss");
}
</script>
@endsection
```

## Method 2: Jquery

Include min.js in common function

```
<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
```

```
<td><button class="btn btn-primary btn-sm
status_update">{{($datas->status==1) ? 'Approve' :  'Not approved'
}}</button></td>
```

```
   $(document).ready(function(){
   console.log('hi from jQuery!');
   $('.status_update).on('click',function(){
        alert("dd");
   });
});
```

## Form validation

```
<script>
          function GEEKFORGEEKS() {

 var name =   document.forms.RegForm.Name.value;
             var email =   document.forms.RegForm.EMail.value;
             var phone =   document.forms.RegForm.Telephone.value;
             var what =  document.forms.RegForm.Subject.value;
             var password =  document.forms.RegForm.Password.value;
             var address =   document.forms.RegForm.Address.value;
      var regEmail=/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/g;  //Javascript reGex for Email
Validation.
             var regPhone=/^\d{10}$/;                              // Javascript reGex for Phone Number validation.
             var regName = /\d+$/g;                                 // Javascript reGex for Name validation

             if (name == "" || regName.test(name)) {
                window.alert("Please enter your name properly.");
                name.focus();
                return false;
             }

             if (address == "") {
                window.alert("Please enter your address.");
                address.focus();
                return false;
             }

             if (email == "" || !regEmail.test(email)) {
                window.alert("Please enter a valid e-mail address.");
                email.focus();
                return false;
             }

             if (password == "") {
                alert("Please enter your password");
                password.focus();
                return false;
             }

             if(password.length <6){
                alert("Password should be atleast 6 character long");
                password.focus();
                return false;

            }
            if (phone == "" || !regPhone.test(phone)) {
                alert("Please enter valid phone number.");
                phone.focus();
                return false;
            }

            if (what.selectedIndex == -1) {
                alert("Please enter your course.");
                what.focus();
                return false;
            }

            return true;
        }
    </script>
...................................................................
<body>
    <h1 style="text-align: center;">REGISTRATION FORM</h1>
    <form name="RegForm" onsubmit="return GEEKFORGEEKS()" method="post">

<p>Name: <input type="text"
                size="65" name="Name" /></p>
        <br />
<p>Address: <input type="text"
                size="65" name="Address" />
    </p>

      <br />

<p>E-mail Address: <input type="text"
                size="65" name="EMail" /></p>

      <br />

<p>Password: <input type="text"
              size="65" name="Password" /></p>

      <br />

<p>Telephone: <input type="text"
              size="65" name="Telephone" /></p>

      <br />
```

```
<p>
            SELECT YOUR COURSE
            <select type="text" value="" name="Subject">
               <option>BTECH</option>
               <option>BBA</option>
               <option>BCA</option>
               <option>B.COM</option>
               <option>GEEKFORGEEKS</option>
            </select>
         </p>

         <br />
         <br />

<p>Comments: <textarea cols="55"
                     name="Comment"> </textarea></p>


<p>
            <input type="submit"
               value="send" name="Submit" />
            <input type="reset"
               value="Reset" name="Reset" />
         </p>

      </form>
   </body>
```

……………………………………………………………………………………

Javascript

```
Event function in document.getElementById("savebtn").addEventListener("click",
function(event){
});
```

Null checking java script

```
for (i=0;i<emp.length;i++){
                              if(emp[i]==''){
                                        alert("please enter all details");
                                         event.preventDefault()
                                        return false;
                              }
                              // document.write(emp[i] + "<br/>");
                    }
```

Regex

```
var testname=/^(?!^[\s])(?!.*[\s]$)(?!.*[\s]{2})[A-Za-z\s]+$/;
                    var
testemail=/^(?!^[.\-_])(?!.*[.\-_@]{2})[A-Za-z0-9.\-_]+@[A-Za-z0-9.\-_]+(?:\.[A-Za-z]+)$/;

                    var testDOB=/^[\d]{4}-[\d]{2}-[\d]{2}$/;

                    var testpass=/^(?!^[#\-_.$%])(?!.*[#\-_.$%]{2})[\w#\-_.$%]{8}$/;
```

```
mobile.match(/\d/g).length!=10
if(!testemail.test(email)){
                                alert("not valid email");
                                event.preventDefault();
                     }
```

Install Composer

a. sudo apt update

d. curl -sS https://getcomposer.org/installer -o /tmp/composer-setup.php

e. sudo php /tmp/composer-setup.php --install-dir=/usr/local/bin –filename=composer

Now type composer and you will get the composer information

f. Composer

alias composer='/usr/local/bin/composer.phar'

7. Install NodeJS using NVM

a. curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh | bash

b. source ~/.bashrc

Check Installed NVM Version

c. nvm –version

Product Development Department C-DIT 4

Setting production environment Linux

Check for all versions of NodeJS available for download

d. nvm ls-remote

This would list various versions of NodeJs, check for the latest stable version. (or version of your requirement)

Install the required version

e. nvm install version

Ex: nvm install 16.15.0

**Image Upload**

```
        if(isset($request->photo)){
            $request->validate(['photo'=>['required','mimes:jpg,png,jpeg']]);
            $id=\Crypt::decrypt($request->user_id);
            $phname='user_'.$id.date("y:m:d:h:m:s").$request->photo->extension();
            $path=$request->file('photo')->storeAS('/uploads/userPhoto',$phname,'myfilesab');
            $datarr=array("photo"=>$phname);
            $res=User::where('id',$id)->update($datarr);
            $user_det=User::where('id',$id)->first();
            return view('dashboard',compact('user_det'));
       }else{
         return back()->withInput()->withErrors('Please select photo');
       }
```

**Captcha**

```
composer require mews/captcha
```

config/app.php

```
'providers' => [
        // ...
        Mews\Captcha\CaptchaServiceProvider::class,
    ]
'aliases' => [
        // ...
        'Captcha' => Mews\Captcha\Facades\Captcha::class,
    ]
php artisan vendor:publish
```

config/captcha.php

```
return [
    'default'   => [
        'length'    => 5,
        'width'     => 120,
        'height'    => 36,
        'quality'   => 90,
        'math'      => true,  //Enable Math Captcha
        'expire'    => 60,    //Captcha expiration
    ],
    // ...
];
```

**INTERVIEW QUESTIONS**

Which of the following method of Exception class returns formated string of trace?
Which of the following method of Exception class returns source line?
Which of the following method can be used to parse an XML document using PHP?
Which of the following function is used to get the size of a file?
Which of the following is correct about NULL?

our educational cerificates are not uploaded.our educational cerificates are not uploaded.,
nosql, rdbms, php, laravel, postgresql, array,  constraints, constructor, sql injunction, html, joins, error handling,
framework advantages,

# Laravel interview Questions

**1.What is Database?**

DBMS stands for Database Management System. DBMS is a system software responsible for the creation, retrieval, updation and management of the database. It ensures that our data is consistent, organized and is easily accessible by serving as an interface between the database and its end users or application softwares.

**2.What is RDBMS? How is it different from DBMS?**

RDBMS stands for Relational Database Management System. The key difference here, compared to DBMS, is that RDBMS stores data in the form of a collection of tables and relations can be defined between the common fields of these tables. Most modern database management systems like MySQL, Microsoft SQL Server, Oracle, IBM DB2 and Amazon Redshift are based on RDBMS.

3.**Artisan**

**Artisan** is the name of the command-line interface included with **Laravel**. It provides a number of helpful commands for your use while developing your application. It is driven by the powerful Symfony Console component.

```
 Eg:*php artisan list
    *php artisan help migrate
    *php artisan --version
```

4.**What are Constraints in SQL?**

Constraints are used to specify the rules concerning data in the table. It can be applied for single or multiple fields in an SQL table during creation of table or after creation using the ALTER TABLE command. The constraints are:

- **NOT NULL** - Restricts NULL value from being inserted into a column.
- **CHECK**: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
- **DEFAULT** - Automatically assigns a default value if no value has been specified for the field.
- **UNIQUE** - Ensures unique values to be inserted into the field.
- **INDEX** - Indexes a field providing faster retrieval of records.
- **PRIMARY KEY** - Uniquely identifies each record in a table.
- **FOREIGN KEY** - Ensures referential integrity for a record in another table.

**5.What are the main error types in PHP and how do they differ?**

In PHP there are three main type of errors:

- **Notices** – Simple, non-critical errors that are occurred during the script execution. An example of a Notice would be accessing an undefined variable.

- **Warnings** – more important errors than Notices, however the scripts continue the execution. An example would be include() a file that does not exist.

- **Fatal** – this type of error causes a termination of the script execution when it occurs. An example of a Fatal error would be accessing a property of a non-existent object or require() a non-existent file.

**6.What is the difference between GET and POST?**

- GET displays the submitted data as part of the URL, during POST this information is not shown as it’s encoded in the request.
- GET can handle a maximum of 2048 characters, POST has no such restrictions.
- GET allows only ASCII data, POST has no restrictions,binary data is also allowed.
- Normally GET is used to retrieve data while POST to insert and update.

## 7.What are the 3 scope levels available in PHP and how would you define them?

**Private** – Visible only in its own class

**Public** – Visible to any other code accessing the class

**Protected** – Visible only to classes parent(s) and classes that extend the current class.

## 7.What are the construct() and destruct() methods in a PHP class?

| construct() | destruct() |
|---|---|
| To create and initialize a class object in a single step, PHP provides a special method called as Constructor, which is used to construct the object by assigning the required property values while creating the object. | for destroying the object Destructor method is used. |
| function __construct()<br>{<br>// initialize the object properties<br>} | function __destruct()<br>{<br>// clearing the object reference<br>} |
| PHP doesn't support function overloading hence we cannot have multiple | PHP Destructor method is called just before PHP is about to release any object |

|  |  |
|---|---|
| implementations for constructor in a class. | from its memory. Generally, you can close files, clean up resources etc in the destructor method. |

## 8) Use of limit and offset in db

| LIMIT | OFF SET |
|---|---|
| The limit keyword is used to limit the number of rows returned in a query result. | The OFF SET value allows us to specify which row to start from retrieving data |
| It can be used in conjunction with the SELECT, UPDATE OR DELETE commands LIMIT keyword syntax | It can be used in conjunction with the SELECT, UPDATE OR DELETE commands LIMIT keyword syntax |
| SELECT * FROM members LIMIT 10; | SELECT * FROM Orders LIMIT 10 OFFSET 15; |

## 9) substr() and strstr() in PHP

| **substr()** | **strstr()** |
|---|---|
| The substr() is a built-in function in PHP that is used to extract a part of a string. | It searches for the first occurrence of a string inside another string and displays the portion of the latter starting from the first occurrence of the former in the latter (before if specified). This function is case-sensitive. |
| **Syntax:** substr(string_name, start_position, string_length_to_cut) | **Syntax:** strstr( $string, $search, $before ) |

**10) Difference bw MySQL and Postgresql**

| **MySQL** | **Postgresql** |
|---|---|
| It is the most **popular** Database. | It is the most **advanced** Database. |
| MySQL is a community driven DBMS system | PostgreSQL is an Object Relational Database Management System (ORDBMS) |
| MySQL only supports JSON (**JSON** is often used when data is sent from a server to a web page) | PostgreSQL support modern applications feature like JSON, XML etc -Extensible Markup Language (**XML**) |
| SQL only supports **Standard data types**. | It supports **Advanced data types** such as arrays, hstore and user defined types. |
| SQL provides **limited MVCC support** ( in InnoDB) | **Full MVCC** support. |
| MySQL boasts some notable users: Facebook,Google,Flickr,GitHub,NASA,Netflix,Spotify,Tesla,Twitter,Uber,Wikipedia,YouTube. | PostgreSQL users include: Apple,Cisco,Debian,,Facebook,Fujitsu,IMDB,Instagram,Macworld,Red Hat,Skype,Spotify,Sun Microsystem,Yahoo |

| **Parameter** | **MYSQL** | **PostgreSQL** |
|---|---|---|
| Open Source | The MySQL project has made its source code available under the terms of the GNU General Public License. | PostgreSQL is released under the PostgreSQL license which is a free Open Source license. This is similar to the BSD & MIT licenses. |
| Acid compliance | MySQL is ACID compliant only when it is used with InnoDB and NDB Cluster Storage engines. | PostgreSQL is completely ACID compliant. |
| SQL compliant | MySQL is partially SQL compliant. For example, it | PostgreSQL is largely SQL compliant. |

| | | |
|---|---|---|
| | does not support check constraints. | |
| Community Support | It has a large community of contributors who Focus mainly on maintaining existing features with new features emerging occasionally. | Active community constantly improves its existing features while its innovative community strives to ensure it remains the most advanced database. New cutting-edge features and security enhancements regularly released. |
| Performance | It is mostly used for web-based projects that need a database for straightforward data transactions. | It is highly used in large systems where to read and write speeds are important |
| Best suited | MySQL performs well in OLAP & OLTP systems when only read speeds are needed. | PostgreSQL performance well when executing complex queries. |
| Support for JSON | MySQL has JSON data type support but does not support any other NoSQL feature. | Support JSON and other NoSQL features like native XML support. It also allows indexing JSON data for faster access. |
| Support for materialized views | Supports materialized views and temporary tables. | Supports temporary tables but does not offer materialized views. |
| Ecosystem | MySQL has a dynamic ecosystem with variants like MariaDB, Percona, Galera, etc. | Postgres has had limited high-end options. However, it is changing with new features introduced in the latest version. |
| Default values | The default values can be overwritten at the session level and the statement level | The default values can be changed at the system level only |
| B-tree Indexes | Two or more B-tree | B-tree indexes merged at |

|  |  |  |
|---|---|---|
|  | indexes can be used when it is appropriate. | runtime to evaluate are dynamically converted predicates. |
| Object statistics | Fairly good object statistics | Very good object statistics |
| Join capabilities | Limit join capabilities | Good join capabilities |
| GitHub Stars | 3.34k | 5.6k |
| Forks | 1.6k | 2.4k |
| Prominent Companies using the product | Airbnb, Uber, Twitter | Netflix, Instagram,Groupon |

## <u>**11) CSRF protection**</u>

Cross-Site Request Forgery Prevention Cheat Sheet

Is a type of attack that occurs when a malicious web site, email, blog, instant message, or program causes a user's web browser to perform an unwanted action on a trusted site when the user is authenticated.

## **12)<u>Difference between RDBMS and ORDBMS</u>**

| **BASIS** | **RDBMS** | **OODBMS** |
|---|---|---|
| Long Form | Stands for Relational Database Management System. | Stands for Object Oriented Database Management System. |
| Way of storing data | Stores data in Entities, defined as tables hold specific information. | Stores data as Objects. |
| Data Complexity | Handles comparatively simpler data. | Handles larger and complex data than RDBMS. |

| | | |
|---|---|---|
| Grouping | Entity type refers to the collection of entities that share a common definition. | Class describes a group of objects that have common relationships, behaviors, and also have similar properties. |
| Data Handling | RDBMS stores only data. | Stores data as well as methods to use it. |
| Main Objective | Data Independence from application program. | Data Encapsulation. |
| Key | A Primary key distinctively identifies an object in a table.. | An object identifier (OID) is an unambiguous, long-term name for any type of object or entity. |

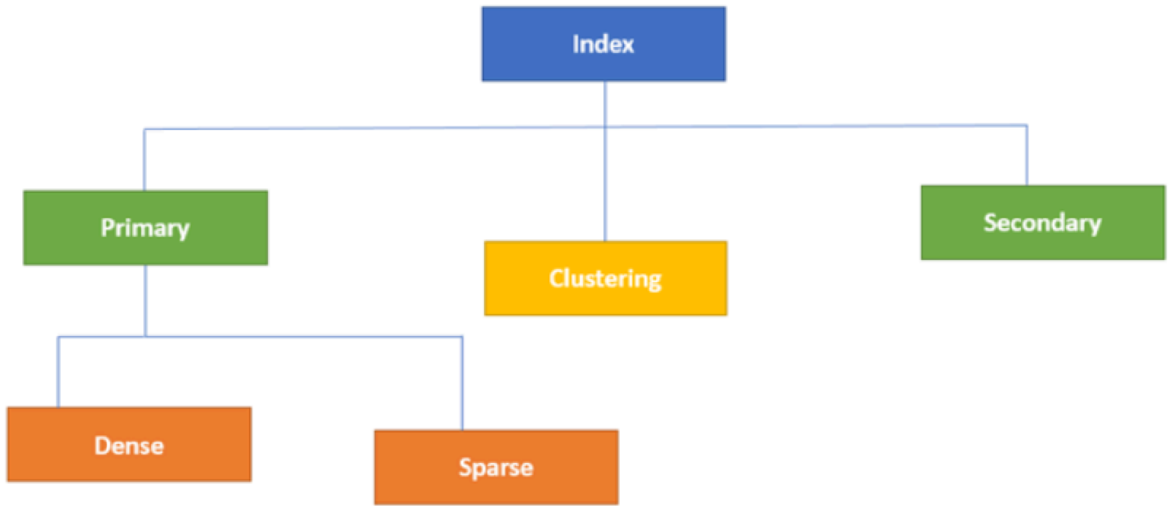
13) **Use of middleware**

Middleware is software which lies between an operating system and the applications running on it. Essentially functioning as a hidden translation layer, middleware enables **communication** and data management for distributed applications.

**14) indexing in db**

Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database.

## Types of Indexing

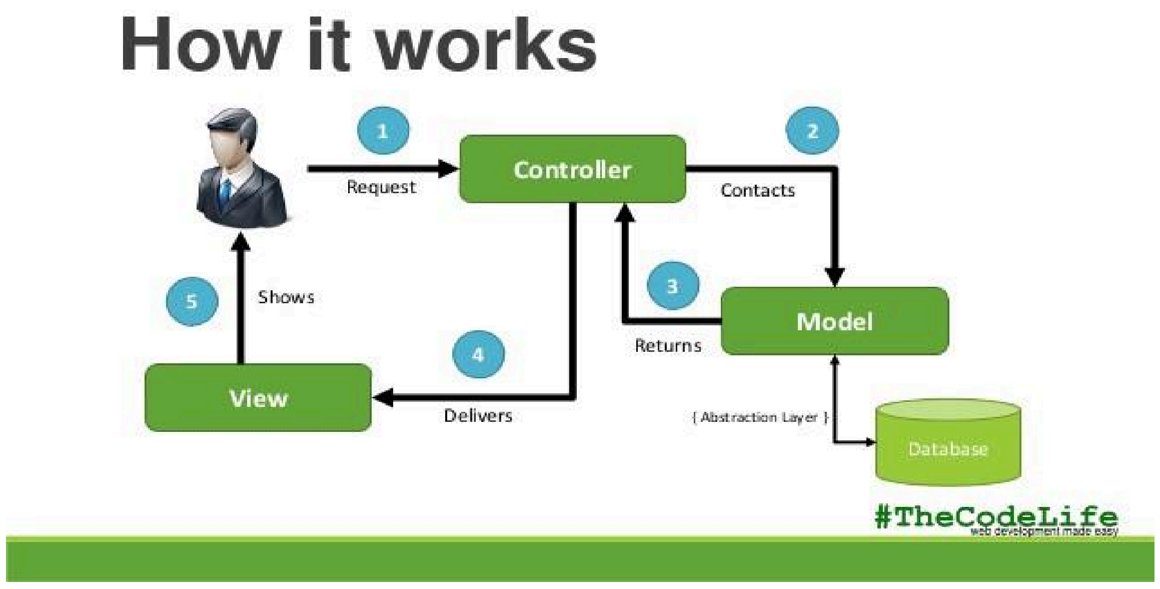


Type of Indexes in Database

## 14.**CORS**

CORS (cross-origin resource sharing) is a method that permits restricted web page resources to be requested from a domain other than the one that supplied the first resource.

15) Explain MVC



16)Important points

- `Console`
  - Contains all your Artisan commands
- `Http`
  - Contains all your controllers, middleware, requests, and routes file
- `Providers`
  - Contains all your application service providers. You can read more about Service Providers here
- `Events`
  - Contains all your event classes
- `Exceptions`

    - Contains your application exception handler and custom exception classes
- `Jobs`
    - Contains all the jobs queued by your application
- `Listeners`
    - Contains all the handler classes for your events
- `Policies`
    - Contains the authorization policy classes for your application. Policies are used to determine if a user can perform a given action against a resource.

The other directories namely:

- `boostrap` contains your framework autoloading files and generated cache files
- `config` contains your app's configuration files
- `database` contains your database migrations and seeds
- `public` contains your assets (images, JavaScript, css, etc.)
- `resources` contains your views and localization files
- `storage` contains all your compiled Blade templates, file caches, and logs
- `tests` contains all your tests
- `vendor` contains your app dependencies

**PHP PROGRAMMING QUESTIONS**

1.Palindrome