Web Application Architecture [20 minutes]

1. **Server – Hardware**
   A server is a computer that serves your web application. This could be physical or virtual machine, depending on the needs of your web application.

2. **Server – Software**
   Server software is responsible for receiving, handling and responding to client requests.

3. **Database**
   Database is server software that is used to store data for your web application.

4. **Database Abstraction**
   Database abstraction is used to separate the code you use to request for data from database-specific ways of retrieving that data from the database server.

5. **Framework**
   Framework is a tool that contains common code base available to be extended for your web application. There are frontend and backend frameworks.

   Frontend frameworks:
   a. Bootstrap
   b. AngularJS
   c. jQuery

   Backend frameworks:
   a. Laravel
   b. Rails
   c. Django
   d. Zend

6. **Backend**
   Backend is responsible for server-side operations of your web application. For backend you will usually use a programming or scripting language, like PHP, Python, Ruby, JavaScript or Java.

7. **Frontend**
   Frontend is responsible for client-side operations of your web application. It is usually written using HTML, CSS and JavaScript.

Laravel Request Lifecycle [20 minutes]

1. **Entry point**
   The starting point of Laravel web application is index.php, located in public directory. This file loads the rest of the framework using Composer generated autoload definition.

2. **HTTP / Console Kernels**
   Laravel uses different kernels to handle HTTP and console requests. Console kernel handles console requests while HTTP kernel is responsible to handle HTTP requests.

   HTTP kernel takes HTTP request as an input, and returns HTTP response for an output.

Hypertext Transfer Protocol – application layer protocol
This protocol dictates the format to request data from a server and the format the data is returned to user.

HTTP Request Methods

GET	The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
HEAD	Same as GET, but it transfers the status line and the header section only.
POST	A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
PUT	Replaces all the current representations of the target resource with the uploaded content.
DELETE	Removes all the current representations of the target resource given by URI.
CONNECT	Establishes a tunnel to the server identified by a given URI.
OPTIONS	Describe the communication options for the target resource.
TRACE	Performs a message loop back test along with the path to the target resource.

Example of HTTP Request:

```
GET /page.html HTTP/1.1
HOST: nyu.edu
...
```

Example of HTTP Response for page found:

```
HTTP/1.1 200 OK
Content-type: text/html
...
```

Example of HTTP Response for page not found:

```
HTTP/1.1 404 Not Found
Content-type: text/html
...
```

HTTP Response Codes

Class	Code	Text	Comments
Success	200	OK	All is well. Valid request, valid response.
Redirection	301	Moved Permanently	Page is now at a new location, automatic redirects are built in to most browsers.
	302	Found	Page is now at a new location *temporarily*.

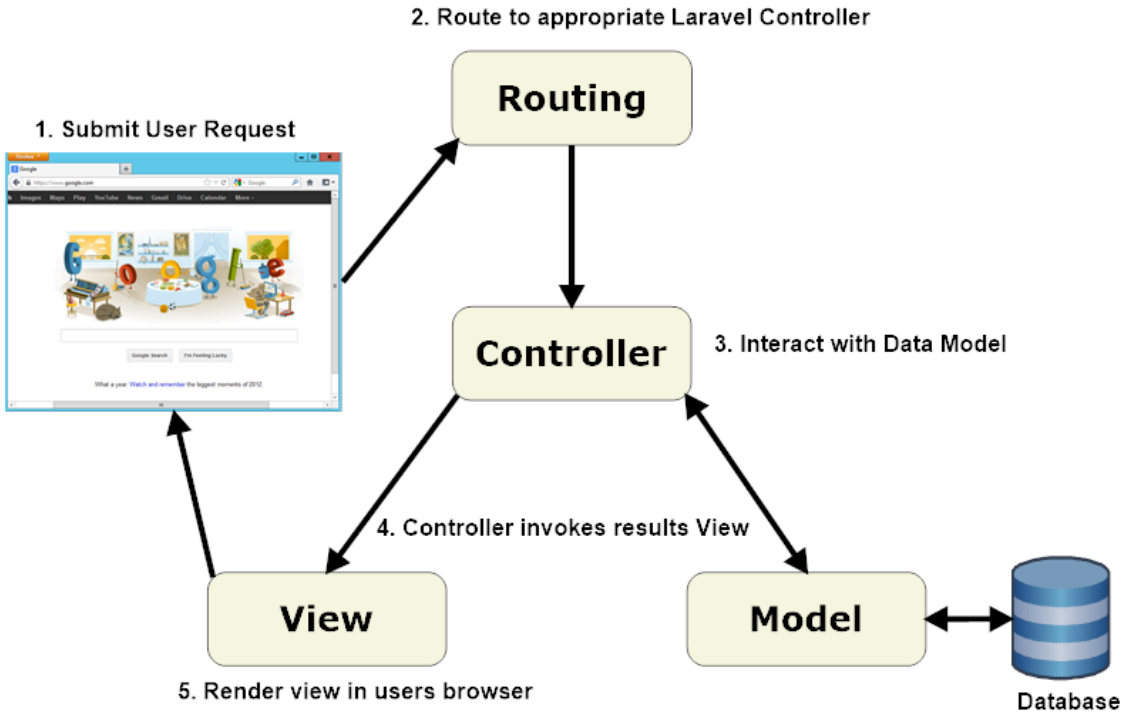
Client Error	401	Unauthorized	Page typically requires login requests.
	403	Forbidden	Server will not allow this request.
	404	Not Found	Server cannot find what you asked for.
Server Error	500	Internal Server Error	Generic server failure in responding to the otherwise valid request.

Laravel [30 minutes]

Model-View-Controller

Model-View-Controller architectural pattern provides separation of the web application components. Each of MVC components has very specific, well-defined roles and interactions with other components.

Models are responsible for retrieving the data.
Views are typically HTML pages rendered by controllers.
Controllers are the core logic of your web applications.



Source: http://laravelbook.com/laravel-architecture/

Routing [15 minutes]

Laravel routing component is responsible for comparing HTTP URI request against routes.php file. If the route exists within that file, Laravel will use the directions to handle the request. There are 2 main types of directions you will be using: 1) closures or 2) controller methods.

Basic route for HTTP GET request using closure:

```
Route::get('hello', function () {
    return 'Hello World';
});
```

Basic route for HTTP GET request with controller method callback:

```
Route::get('hello', 'SampleController@hello');
```

Following methods are available for Laravel routing:

```
Route::get($uri, $callback);
Route::post($uri, $callback);
Route::put($uri, $callback);
Route::patch($uri, $callback);
Route::delete($uri, $callback);
Route::options($uri, $callback);
```

In case you need to match multiple HTTP request methods, you can do so with the following way:

Match HTTP request methods GET and POST:

```
Route::match(['get', 'post'], '/', function () {
   //
});
```

Match HTTP request any methods:

```
Route::any('foo', function () {
    //
});
```

Route parameters

Route with required parameters

```
Route::get('user/{id}', function ($id) {
    return 'User '.$id;
});
```

Route with optional parameters

```
Route::get('user/{name?}', function ($name = null) {
    return $name;
});
```

Route middleware

Route with 'auth' middleware

```
Route::group(['middleware' => 'auth'], function () {
    Route::get('/', function ()    {
        // Uses Auth Middleware
    });
});
```

Route groups

Route with 'admin' prefix

```
Route::group(['prefix' => 'admin'], function () {
    Route::get('users', function ()    {
        // Matches The "/admin/users" URL
    });
});
```

More about routing: https://laravel.com/docs/master/routing

Controllers [5 minutes]

Controllers are responsible for coordinating between models and views. They are the next logical step after the route URI. Controllers handle inputs, interact with the models, and render the output through the views.

More about controllers: https://laravel.com/docs/master/controllers

Views [10 minutes]

Views are the visual representation of the model. They are typically presented as HTML pages, styled with CSS. Additionally, JavaScript is used to control user interactions and various user interface elements.

More about views: https://laravel.com/docs/master/views

Sources:

1. https://laravel.com/docs/master
2. http://www.tutorialspoint.com/http/http_requests.htm
3. https://cs50.harvard.edu/