UNVEILING LARAVEL: EXPLORING THE MVC PARADIGM AND CORE FEATURES

- *Teacher: Sir Arsalan Shah*
- *Typist: Yousuf Naveed*

Welcome to our exploration of Laravel – a versatile framework that simplifies web development through organized structures and pre-defined features. In this lecture, we'll break down Laravel's core elements, including security measures, API integration, authentication processes, and more. Our focus will be on the MVC (Model-View-Controller) architecture, a key concept that enhances web application development. By the end, you'll grasp the basics of Laravel and understand how it can streamline your web development journey. Let's dive in!

Framework: A Collection of Classes.

- **Security:** Routes, HTTP Protocol.
- **API:** Application Program Interface (In-built),
  Types: Post, Get, Update & Put…
- **Authentication:** Email & Password, etc.
- **Database:** ORM (Object Relational Map).
- **CLI** (Command-Line Interface).

Why use a framework:

- **Organization:** Frameworks provide a structured environment, facilitating systematic code arrangement for better readability and maintenance.
- **Pre-defined:** Frameworks come with built-in functionalities and conventions, reducing the need for manual coding and speeding up development by offering ready-to-use solutions.

A web application means establishing a structure for a website. Laravel follows the MVC (Model-View-Controller) concept, a common paradigm used in various frameworks such as Django, Flask, ASP.net, ASP.net Core, Mern, Next.js, React.js, Flutter, etc.

Php framesworks using MVC:

- Laravel
- Symfony
- Yii
- CodeIgniter
- Cake PHP

Model-View-Controller (MVC):

Model: Database

View: HTML...

Controller: Logic

Example of Controller:

Suppose a user sends a request (hits a URL), where the request can be any action or move made by the user on our view page. This action is considered a request. The controller examines the request; if it involves a database interaction, the controller promptly forwards the request to the model (database-related action). Upon receiving a response, the controller forwards it to the view, which is then displayed to the user. If there is no database involvement, the response generated by the controller is directly sent to the view for display.