What is CodeIgniter 3?

CodeIgniter 3 is a powerful PHP framework with a small footprint, designed for developers who need a simple and elegant toolkit to build full-featured web applications. It uses the Model-View-Controller (MVC) pattern, which helps separate logic, design, and database operations.

Features include:
- Lightweight and fast performance
- MVC Architecture
- Built-in libraries and helpers
- Excellent documentation
- Good security (CSRF, XSS filtering)
- No need for Composer or CLI

Use CodeIgniter 3 when:
- You want quick development with minimal configuration
- Your hosting doesn't support advanced setups like Laravel
- You are learning or building small-to-medium applications

Directory Structure

application/
config/ => Configuration files (config.php, database.php, autoload.php)
controllers/ => Application logic
models/ => Database interaction logic
views/ => HTML/PHP templates (frontend)
libraries/ => Custom or extended libraries
helpers/ => Reusable procedural functions

system/
Core CodeIgniter framework files

index.php => Entry point of the application

Configuration Files

config.php – base URL, session, encryption settings
database.php – DB hostname, username, password, dbname
autoload.php – autoload libraries, helpers, models

Understanding MVC in CI3

Model - Handles database operations.
View - Displays the data as HTML.
Controller - Acts as a bridge between model and view.

Example:
- Controller: Loads model, fetches data, passes to view.
- Model: Uses Query Builder to get data from DB.
- View: Displays that data using PHP.

Helpers vs Libraries

Helpers: Collection of procedural functions (e.g., url_helper, form_helper)
Libraries: Class-based utilities (e.g., email, session)

Autoload in autoload.php to avoid manual loading.

Routing & URI Segments

Routing defines which controller handles a request.
$route['default_controller'] = 'Home';

URI segments allow dynamic data:
URL: site.com/user/edit/4
$this->uri->segment(3) => 4

Database Usage & CRUD

Use Query Builder for CRUD:

$this->db->insert(), ->get(), ->update(), ->delete()

Join, group_by, order_by, transactions are supported.
Stored procedures can be called via $this->db->query("CALL proc_name()");

Forms & Input

$this->input->post('field')
$this->form_validation->set_rules()
Enable CSRF protection in config.php
File upload via $this->upload->do_upload('field')

Sessions & Flashdata

$this->session->set_userdata(), ->userdata(), ->unset_userdata()
$this->session->set_flashdata(), ->flashdata()
set_cookie(), get_cookie() for cookies

Security

XSS Filtering: $this->security->xss_clean($data)
Password Hashing: password_hash(), password_verify()
Sanitize inputs and validate file uploads carefully

File Handling & Export

Download files using force_download()
Export CSV using fputcsv()
Export Excel/PDF using third-party libraries like PhpSpreadsheet or TCPDF

Multiple Views & Bootstrap

Load multiple views like header, body, footer.
$this->load->view('header'); etc.

Use Bootstrap by linking its CDN in your view templates.

AJAX, Login & Authentication

Use jQuery to send AJAX requests to CI controller methods.
Validate login using form_validation and check hashed passwords.
Use sessions to manage login state.

Error Handling & Logging

$route['404_override'] = 'Errors/show404';
Use try-catch for exception handling.
log_message('error', 'Error message here');

Emails

$this->email->from(), ->to(), ->subject(), ->message(), ->send()
Can use SMTP settings for external email providers

Custom Helpers, Libraries & Hooks

Create helpers in application/helpers/
Create libraries in application/libraries/
Enable and define hooks in config/hooks.php

Hooks can run before/after controllers or globally

HMVC in CI3

Use Modular Extensions (HMVC) for modular development.
Each module has its own controllers, models, views.
Useful for large applications with multiple independent components.