CodeIgniter Coding Standards

Naming Conventions

- Use PascalCase for class names (e.g., UserModel, OrderLibrary).
- Use camelCase for variables and functions.
- Table names should be lowercase and plural (e.g., users, products).
- Controller file names must match the class name exactly.
- Libraries and Helpers use snake_case file names but PascalCase class names.
- Avoid abbreviations in naming for clarity.

Security Practices

- Enable CSRF protection in config.php.
- Use the input class to fetch user input ($this->input->post).
- Always use Query Builder or parameter binding to avoid SQL injection.
- Use XSS filtering ($this->security->xss_clean).
- Restrict file upload types and validate extensions.
- Never store passwords in plain text.
- Use encryption library for sensitive data.

Syntax Structure

- Follow PSR-12 coding standards.
- Class names must start with an uppercase letter.
- Methods should be grouped logically.
- Use constructor for dependency injection.
- Use short array syntax [].

Error Handling

- Use CodeIgniter's logging library for errors.
- Display errors only in development mode.
- Use try-catch for critical operations like database transactions.

- Use set_status_header for HTTP errors.

Statements & Control Flow

- Always use braces for control structures.
- Use early return to avoid deep nesting.
- Validate input before processing.
- Use strict comparison where applicable.

File Organization

- Controllers go in application/controllers.
- Models go in application/models.
- Views go in application/views.
- Libraries go in application/libraries.
- Helpers go in application/helpers.
- Routes are configured in application/config/routes.php.

Comments & Documentation

- Use PHPDoc for controllers, models, and libraries.
- Document methods with @param and @return.
- Comment complex logic and queries.
- Keep comments synchronized with code changes.

Indentation & Formatting

- Use 4 spaces for indentation.
- Max line length of 120 characters.
- Blank line between methods.
- Consistent spacing for operators and control structures.
- Avoid trailing whitespace.

Best Practices

- Use models for database operations, not controllers.
- Separate business logic from presentation logic.
- Use libraries or services for reusable components.
- Enable hooks for pre- and post-processing.
- Keep configuration in config files, not in controllers.
- Always sanitize and validate user input.