🏗️ MVC ARCHITECTURE EXPLAINED (for your app)

Your folder structure:

```
bike-app/
│
├── index.php
├── controller/
├── model/
├── view/
├── helpers/
└── config/
```

I'll explain **each part**.

1️⃣ `index.php` — The Front Controller

What it does:

- It is the **entry point of your application**.
- Every request (click, URL, form submission) goes **through this file** first.
- It decides **which controller and action to call** based on the URL.

Example:

`index.php?page=login`

- `index.php` sees `page=login`
- Loads `AuthController.php`
- Calls `login()` method → loads login view

Why it's important:

- Keeps routing in **one central place**
- Makes the app organized and scalable

2️⃣ `controller/` — The Controllers

Controllers are the "brain" that handles logic.

- Example: `ProductController.php`, `AuthController.php`
- Their job:
    1. Receive the request from `index.php`
    2. Talk to the **model** to get or save data
    3. Decide which **view** to show

Example:

```
class ProductController {
    public function index() {
        $products = Product::all(); // call model
        require 'views/front/product_list.php'; // show view
    }
}
```

Why it's important:

- Separates **business logic** from **HTML/UI**
- Makes it easier to **add features or change logic** later

3️⃣ `model/` — The Models

Models are the "data layer" — they interact with the database.

- Example: `Product.php`, `User.php`
- Their job:
    1. Retrieve data from database
    2. Save/update/delete data
    3. Provide clean methods for the controller

Example:

```
class Product {
    public static function all() {
        return self::db()->query("SELECT * FROM
products")->fetchAll();
    }
}
```

Why it's important:

- Keeps database queries **centralized**
- Controller doesn't need to know SQL details
- Makes app **cleaner and easier to maintain**

4️⃣ `view/` — The Views

Views are the "presentation layer" — what the user sees.

- Example: `home.php`, `product_list.php`, `login.php`
- Their job:
    1. Display HTML/CSS/JS
    2. Show dynamic data provided by the controller

Example:

```
<h2>Products</h2>
<?php foreach ($products as $p): ?>
    <p><?= $p['name'] ?></p>
<?php endforeach; ?>
```

Why it's important:

- Keeps HTML **separate** from logic
- Designers can work on the UI without touching PHP logic

5️⃣ `helpers/` — The Helpers

Helpers are utility classes or functions used across the app.

- Example: `Auth.php`
- Their job:
    - Check if a user is logged in
    - Check roles (`client`, `owner`, `admin`)
    - Protect pages
    - Provide common reusable code

Example:

```
class Auth {
    public static function check() {
        if (!isset($_SESSION['user'])) {
            header('Location: index.php?page=login');
            exit;
        }
    }
}
```

Why we use helpers:

- Avoid repeating code like **“is user logged in?”** in every controller
- Makes controllers **cleaner**

6️⃣ `config/` — Configuration Files

- Example: `database.php`
- Stores settings like:
    - Database host, username, password
    - Charset
- Provides a **database connection** to models

```
class Database {
    public static function connect() {
        return new
PDO("mysql:host=localhost;dbname=bike_rental_app", "root", "");
    }
}
```

Why important:

- Centralized configuration
- Easy to change DB credentials without touching controllers or models

---

7️⃣ `session` — What it is and why we use it

What is a session?

- A session stores data on the **server** about a user while they browse your site
- Commonly used for **login**

- PHP session example:

```
session_start();
$_SESSION['user'] = $user; // store logged-in user info
```

Why we use it in AuthController & helpers:

1. **AuthController:**
    - When a user logs in, we save their info in `$_SESSION['user']`
    - This allows us to **remember the user across pages**
2. **Auth Helper:**
    - Provides reusable functions:
        - `Auth::check()` → block access if not logged in
        - `Auth::isAdmin()` → check role
    - Keeps **security centralized**

Example workflow:

1. User logs in → AuthController sets session
2. User visits `/products` → Auth::check() ensures user is logged in
3. Page loads with **dynamic data** visible only to logged-in users

---

8️⃣ How it all works together

1. **User clicks a URL** → goes to `index.php`
2. **Router** reads URL → chooses controller/action
3. **Controller** calls **Model** → retrieves or updates DB
4. **Controller** loads **View** → HTML output

5. **Helpers** & **Sessions** ensure security and role-based access

```
User → index.php → Controller → Model → DB
                              ↘ View
```

🔹 Key Takeaways

- **Controller** → “brain” / logic
- **Model** → “data” / database
- **View** → “presentation” / HTML
- **Helpers** → utility / common tasks
- **Index.php** → central router
- **Session** → store user login data, access control