جشنواره خوارزمی · پروژه برنامه‌نویسی

موکو

پلتفرم یکپارچه پیام‌رسان، فروشگاه و شبکه اجتماعی

معماری MVC · احراز هویت امن · آپلود فایل FTP · پنل مدیریت

ارائه‌دهنده

تیم پروژه Moku | مصطفی شریعت

نسخه

۱۴۰۵۰ ۱.۰

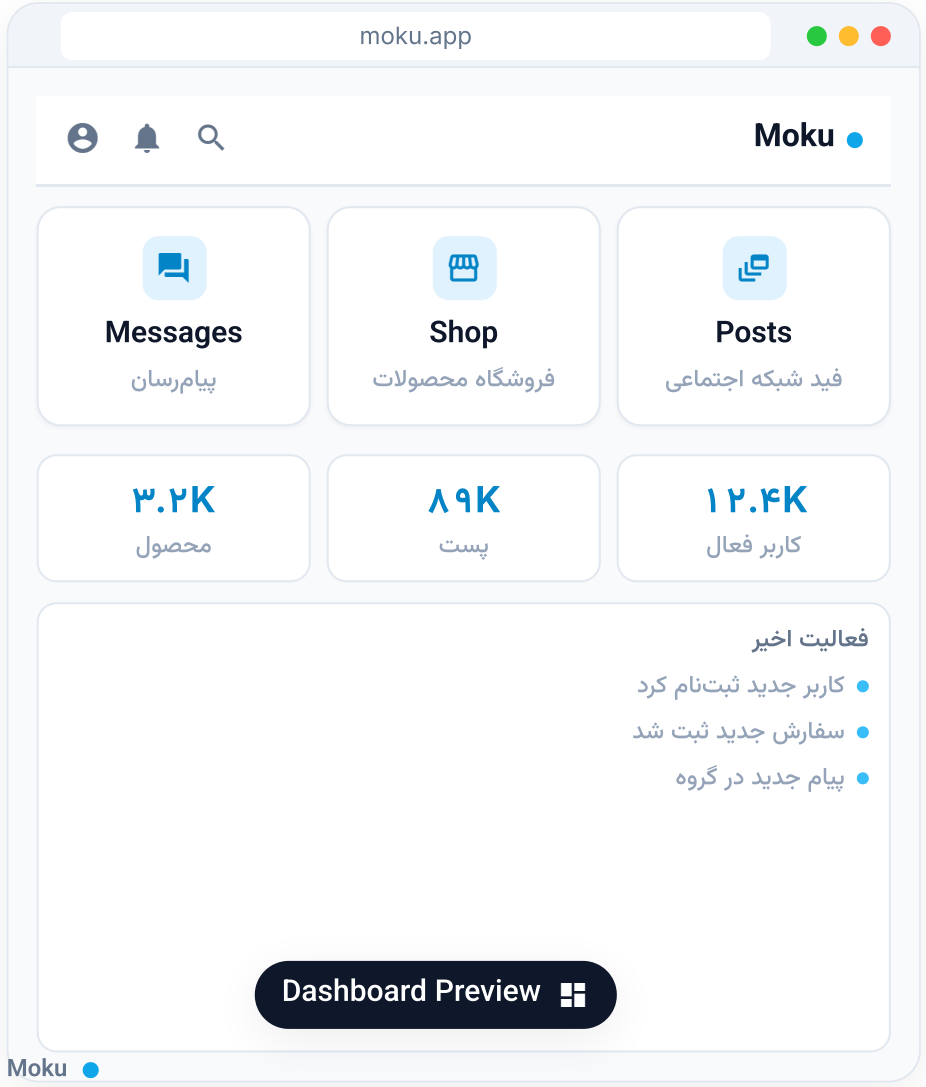
۰۱ · معرفی

Moku چیست؟

Moku یک پلتفرم تحت وب است که تجربه‌ای واحد برای ارتباط، خرید و اشتراک محتوا ارائه می‌دهد. تمام ماژول‌ها روی یک معماری MVC اختصاصی ساخته شده‌اند.

- **پیام‌رسان بلادرنگ با آپلود فایل**
- **فروشگاه با سبد خرید و Checkout**
- **فید پست‌ها با لایک، کامنت و ذخیره**

معماری MVC اختصاصی



اهداف پروژه Moku

چهار محوری که کل تصمیم‌های طراحی و فنی حول آن شکل گرفت

۰۱

یکپارچگی

ترکیب پیام‌رسان، فروشگاه و شبکه اجتماعی در یک پلتفرم واحد با معماری مشترک

Messenger Shop Posts

۰۲

امنیت

احراز هویت قوی، محافظت در برابر CSRF/XSS/SQL Injection، مدیریت امن Session

BCRYPT PDO

۰۳

تجربه کاربری

رابط مینیمال، پاسخگو و سریع با تمرکز بر سادگی و دسترسی‌پذیری

Minimal Responsive

۰۴

مقیاس‌پذیری

معماری ماژولار MVC که افزودن قابلیت‌های جدید را بدون تخریب ساختار فعلی ممکن می‌سازد

Modular MVC

امکانات کلیدی

شش قابلیت محوری که پلتفرم را تشکیل می‌دهند

هسته ۳ قابلیت

پشتیبانی ۳ قابلیت

هسته

احراز هویت

ثبت‌نام، ورود، فراموشی رمز، Session امن

هسته

پیام‌رسان

چت بلادرنگ، آپلود فایل، مدیریت گروه

هسته

فروشگاه

محصولات، سبد خرید، Checkout، مدیریت موجودی

پشتیبانی

شبکه اجتماعی

ساخت پست، لایک، کامنت، ذخیره

پشتیبانی

مدیریت فایل

Uploader با پشتیبانی FTP و Remote Storage

پشتیبانی

پنل مدیریت

مدیریت کاربران، محتوا و گزارش‌ها

User Journey · ۰۴

گردش کاربران

از ثبت‌نام تا تعامل کامل با پلتفرم

۶ مرحله · یک Session

User Flow

نقاط ورود به ماژول‌ها

پیام‌رسان

فروشگاه

شبکه اجتماعی

آمار پروژه

نمای کلی از مقیاس کد و معماری

CODEBASE · ۱۴۰۳

Moku v1.0

ماژول

8

ماژول اصلی

Auth · Messenger · Shop · Posts · Files · Admin · ...

Controller

+34

کنترلر

Controllerهای اختصاصی

Endpoint

+128

مسیر روت

مسیر های عمومی و خصوصی

Schema

+60

جدول دیتابیس

جداول اصلی و رابط

technologies used

PHP · MySQL · Custom MVC · HTML&CSS | JS · FTP · Session

معماری سیستم

الگوی MVC اختصاصی با Routing متمرکز

MVC · ۴ لایه

Custom MVC

جریان درخواست

مسیر یک Request از ورودی تا پاسخ

Request ▶

↓

index.php

↓

Controller

↓

Model → DB

↓

View → Response

جداسازی لایه‌ها

تست‌پذیری بالا و قابلیت نگهداری آسان

لایه ۰۱

لایه ورودی — index.php

Front Controller · Routing · Bootstrap

Entry

↓

لایه ۰۲

لایه کنترلر — Controller.php

... · AuthController · ProductController · AdminController

Logic

↓

لایه ۰۳

لایه مدل — Model.php + Database.php

ORM ساده · Prepared Statements · Query Builder

Data

↓

لایه ۰۴

لایه نمایش — View.php

Template Rendering · Layouts · Partials

UI

Database · ۰۷

لایه دیتابیس

Prepared Statements با Database.php — abstraction layer

ویژگی‌های Database.php

- اتصال PDO به MySQL
- استفاده از 2 پایگاه اطلاعاتی
- 53 جدول دیتابیس اصلی و 6 جدول دیتابس پیامرسان
- متدهای کمکی: insert, update, delete, select

Moku:53 | MokuChat:6

PDO · MySQL

۸ جدول اصلی

جداول اصلی

users
کاربران و احراز هویت

sessions
مدیریت Session

products
محصولات فروشگاه

posts
پست‌های شبکه اجتماعی

Modules · ۰۸

ماژول‌های پروژه

هر ماژول مستقل اما متصل به دیگران

هسته کسب‌وکار ۳ زیرساخت ۳

هسته

۰۱

Auth

ثبت‌نام، ورود، خروج، بازیابی رمز

AuthController

هسته

۰۲

Messenger

چت آنلاین، گروه، فایل

MessengerController

هسته

۰۳

Shop

محصول، سبد، سفارش، Checkout

ProductController

زیرساخت

۰۴

Posts

ساخت، ویرایش، لایک، کامنت

PostController

زیرساخت

۰۵

Files

آپلود، FTP، Remote Storage

Uploader + FileController

زیرساخت

۰۶

Admin

مدیریت کامل سیستم

AdminController

احراز هویت کاربران

جریان کامل از ثبت‌نام تا Session امن

جریان احراز هویت · ۴ مرحله

۰۱ ثبت‌نام
دریافت شماره و رمز · اعتبارسنجی فرم

۰۲ هش رمز
PASSWORD_BCRYPT با cost ۱۲

۰۳ اعتبارسنجی
password_verify در زمان ورود

۰۴ ایجاد Session
ساخت Session ID · ست Cookie HttpOnly

۴ مرحله · امن

BCRYPT · HttpOnly

متدها

AuthController

- register($data)
- login($email, $password)
- logout()
- verify($credentials)

امنیت

- هش رمز با BCRYPT
- Cookie با HttpOnly و Secure
- تولید Session ID تصادفی

محافظت در برابر XSS و Session Hijacking

مدیریت Session و Cookie

نگه‌داری وضعیت کاربر در طول تعامل با پلتفرم

Session

ذخیره‌شده در سرور

server-side

- ذخیره **user_id** و وضعیت لاگین
- منقضی می‌شود با بستن مرورگر
- داده‌های حساس در اینجا نگه‌داری می‌شوند
- در فایل‌ها یا دیتابیس سرور

`SESSION_$`

Cookie

ذخیره‌شده در کلاینت

client-side

- نگه‌داری تنظیمات و ترجیحات
- مانند «مرا به خاطر بسپار»
- داده‌های غیرحساس در اینجا
- با فلگ‌های **HttpOnly** و **Secure**

`setcookie()`

تعامل

Session ID در Cookie ذخیره می‌شود · هر درخواست آن را به سرور می‌فرستد

PHPSESSID ← HTTP Cookie

سیستم آپلود فایل

Uploader.php با پشتیبانی از FTP و Remote Storage

۱ **انتخاب فایل**
فرم multipart

← ۲ **اعتبارسنجی**
نوع · حجم · extension

← ۳ **Uploader.php**
پردازش و آماده‌سازی

← ۴ **انتخاب Storage**
Local · FTP · Remote

← ۵ **انتقال**
FTP/Remote connection

← ۶ **ثبت در دیتابیس**
ذخیره مسیر و متادیتا

قابلیت‌های Uploader.php

- پشتیبانی از FTP و Remote Storage
- اعتبارسنجی نوع و حجم فایل
- تولید نام تصادفی برای جلوگیری از تداخل
- متادیتا در جدول `files`

پشتیبانی از Storage

- **Local** — ذخیره روی سرور
- **FTP** — انتقال به سرور فایل
- **Remote** — سرویس خارجی
- انتخاب پویا بر اساس تنظیمات سیستم

پیام‌رسان بلادرنگ

تمام صفحات و قابلیت‌ها با گردش درخواست

صفحات پیام‌رسان

لیست چت‌ها
messages/

چت تک‌نفره
messages/{id}/

چت گروهی
messages/group/{id}/

تنظیمات
messages/settings/

قابلیت‌ها

- ارسال متن
- آپلود فایل
- وضعیت آنلاین
- خوانده‌شدن پیام
- جستجو
- اعلان

گردش درخواست

از ارسال تا به‌روزرسانی UI گیرنده

۱. **کاربر پیام را ارسال می‌کند**
POST /messages

۲. **MessengerController@store دریافت می‌کند**
controller layer

۳. **ذخیره در جدول messages**
database insert

۴. **اعلان به گیرنده از طریق AJAX**
async polling

۵. **به‌روزرسانی UI گیرنده**
without reload

Shop · ۱۳

فروشگاه آنلاین

تمام صفحات از مرور محصول تا Checkout

۱

محصولات

لیست محصولات با فیلتر

shop/

ProductController@index

۲

جزئیات محصول

نمایش محصول و افزودن به سبد

product/{id}/

ProductController@show

۳

سبد خرید

مدیریت آیتم‌ها و تعداد

cart/

CartController@index

۴

Checkout

تکمیل سفارش و پرداخت

checkout/

CheckoutController@store

جریان کامل: مرور → انتخاب → افزودن به سبد → ادامه خرید یا Checkout → ثبت سفارش → ذخیره در orders

Posts · ۱۴

مدیریت پست‌ها

شش عملیات اصلی روی پست‌های شبکه اجتماعی

create

ساخت پست

PostController@store

edit

ویرایش

PostController@update

delete

حذف

PostController@destroy

like

لایک

LikeController@toggle

comment

کامنت

CommentController@store

save

ذخیره

SaveController@toggle

زنجیره اعمال

کاربر پست می‌سازد ← دیگران لایک/کامنت می‌کنند ← هرکسی می‌تواند ذخیره کند ← سازنده می‌تواند ویرایش یا حذف کند

Admin · ۱۵

پنل مدیریت

AdminController — مدیریت کامل سیستم

پنل مدیریت کنترل کامل روی کاربران، محتوا و گزارش‌های سیستم را فراهم می‌کند.

مدیریت کاربران
لیست، مسدودسازی، نقش‌ها

مدیریت محصولات
افزودن، ویرایش، حذف

مدیریت پست‌ها
نظارت و حذف

گزارش‌ها
آمار فروش و کاربران

AdminController · /admin · Role: admin

Security · ۱۶

امنیت پروژه

پنج لایه دفاعی برای محافظت از پلتفرم

۰۱

CSRF Protection

تولید توکن برای هر فرم · اعتبارسنجی در سرور

۰۲

XSS Prevention

htmlspecialchars روی همه ورودی‌ها · escape خروجی

۰۳

SQL Injection

Prepared Statements در Database.php · هیچ کوئری رشته‌ای

۰۴

Password Hashing

BCRYPT با cost ۱۲ · هیچ رمز متن ساده‌ای ذخیره نمی‌شود

۰۵

Session Security

HttpOnly + Secure + SameSite flags روی همه کوکی‌ها

علاوه بر این

Rate Limiting روی ورود

اعتبارسنجی سرور سخت‌گیرانه

مدیریت خطای امن

Challenges · ۱۷

چالش‌های توسعه

موانع فنی و راه‌حل‌های اتخاذ شده

۰۱

طراحی معماری

چالش: ساخت MVC اختصاصی بدون فریم‌ورک

راه‌حل مطالعه Laravel و Symfony و ساده‌سازی

۰۲

امنیت چندلایه

چالش: محافظت همزمان از CSRF، XSS، SQL Injection

راه‌حل لایه‌بندی در Database و Controller

۰۳

آپلود به FTP

چالش: اتصال پایدار و مدیریت خطا در FTP

راه‌حل abstraction در Uploader.php

۰۴

بلادرنگ‌کردن پیام‌رسان

چالش: به‌روزرسانی UI بدون reload

راه‌حل AJAX با polling و retry

۰۵

یکپارچه‌سازی ماژول‌ها

چالش: ارتباط فروشگاه، پیام‌رسان و شبکه اجتماعی

راه‌حل مدل‌های مشترک و روابط دیتابیس

FAQ · ۱۸

پرسش‌های‌شما

۱۵ پرسش متداول با پاسخ‌های کوتاه

۰۱ چرا فریم‌ورک آماده استفاده نکردید؟
برای یادگیری عمیق مفاهیم و کنترل کامل معماری.

۰۲ الگوی MVC چه مزیتی دارد؟
جداسازی منطق، نمایش و داده که نگهداری را ساده می‌کند.

۰۳ از SQL Injection چطور جلوگیری می‌شود؟
با Prepared Statements در لایه Database.

۰۴ رمز عبور چطور ذخیره می‌شود؟
با BCRYPT و cost ۱۲، هیچ متن ساده‌ای ذخیره نمی‌شود.

۰۵ CSRF چطور محافظت می‌شود؟
با توکن یکتا برای هر فرم و اعتبارسنجی در سرور.

۰۶ آپلود فایل چطور امن است؟
با اعتبارسنجی نوع، حجم و extension و نام تصادفی.

۰۷ پیام‌رسان چطور بلادرنگ است؟
با AJAX و polling دوره‌ای برای دریافت پیام‌های جدید.

۰۸ چرا FTP و Remote Storage پشتیبانی می‌شود؟
برای انعطاف در محل ذخیره و مقیاس‌پذیری.

۰۹ سبد خرید کجا ذخیره می‌شود؟
در دیتابیس و مرتبط با user_id و Session.

۱۰ پنل مدیریت چطور محافظت می‌شود؟
با بررسی نقش admin در middleware قبل از هر متد.

۱۱ کنترلرها چطور با هم ارتباط دارند؟
از طریق مدل‌های مشترک و روابط دیتابیس.

۱۲ چرا Vazirmatn انتخاب شد؟
برای پشتیبانی کامل از فارسی و خوانایی بالا.

۱۳ کاربر چقدر می‌تواند فایل آپلود کند؟
محدودیت حجم و نوع قابل تنظیم در تنظیمات Uploader.

۱۴ چه داده‌هایی در Session ذخیره می‌شود؟
فقط user_id و وضعیت لاگین، داده‌های حساس نه.

۱۵ آیا پروژه قابل توسعه است؟
بله، ماژولار و آماده افزودن قابلیت‌های جدید.

جمع‌بندی

سه درس کلیدی از پروژه Moku

۰۱

معماری مهم‌تر از ویژگی است

MVC اختصاصی امکان افزودن ویژگی‌های جدید را بدون تخریب ساختار ممکن کرد

۰۲

امنیت از روز اول

لایه‌بندی امنیتی از همان ابتدا هزینه‌اش کمتر از اضافه‌کردن دیرتر است

۰۳

یکپارچگی برنده است

تجربه واحد روی سه ماژول، هم برای کاربر و هم برای توسعه‌دهنده بهتر است

از توجه شما سپاسگزارم

Moku · جشنواره خوارزمی · ۱۴۰۵

moku.ir

Page Detail ۰۱/۰۵۰ ۲۰

صفحه ورود و ثبت‌نام

نقطه ورود کاربران به پلتفرم Moku

مشخصات فنی صفحه

نام صفحه

صفحه ورود / ثبت‌نام

ROUTE

POST /login | POST /register | GET /logout

CONTROLLER

AuthController — register, login, logout

METHODS

register($data), login($email, $password), logout()

وظیفه

احراز هویت کاربران، ایجاد Session، مدیریت کوکی «مرا به خاطر بسپار»

ارتباط با سایر بخش‌ها

Database | Model User | Session | Cookie

Flow

ورودی فرم ← اعتبارسنجی ← password_verify ← ایجاد Redirect ← Session به /

صفحه اصلی – فید

نقطه مرکزی تجربه کاربر پس از ورود

مشخصات فنی صفحه

نام صفحه

صفحه اصلی / فید پست‌ها

ROUTE

`/ GET`

CONTROLLER

HomeController index

METHOD

index() – بارگذاری فید، سایدبار و نوتیفیکیشن‌ها

وظیفه

نمایش فید پست‌ها، دسترسی سریع به پیام‌رسان و فروشگاه، نوتیفیکیشن‌ها

ارتباط با سایر بخش‌ها

PostController · MessengerController · ProductController · NotificationController · Session

Flow

بررسی Session ← بارگذاری فید (PostController) ← بارگذاری نوتیفیکیشن ← رندر View

صفحه جزئیات محصول

صفحه نمایش محصول و افزودن به سبد خرید

مشخصات فنی صفحه

نام صفحه

جزئیات محصول

ROUTE

GET /product/{id}

CONTROLLER

ProductController show

METHOD

show($id) — بارگذاری محصول، تصاویر، توضیحات و محصولات مرتبط

وظیفه

نمایش اطلاعات کامل محصول و امکان افزودن به سبد خرید

ارتباط با سایر بخش‌ها

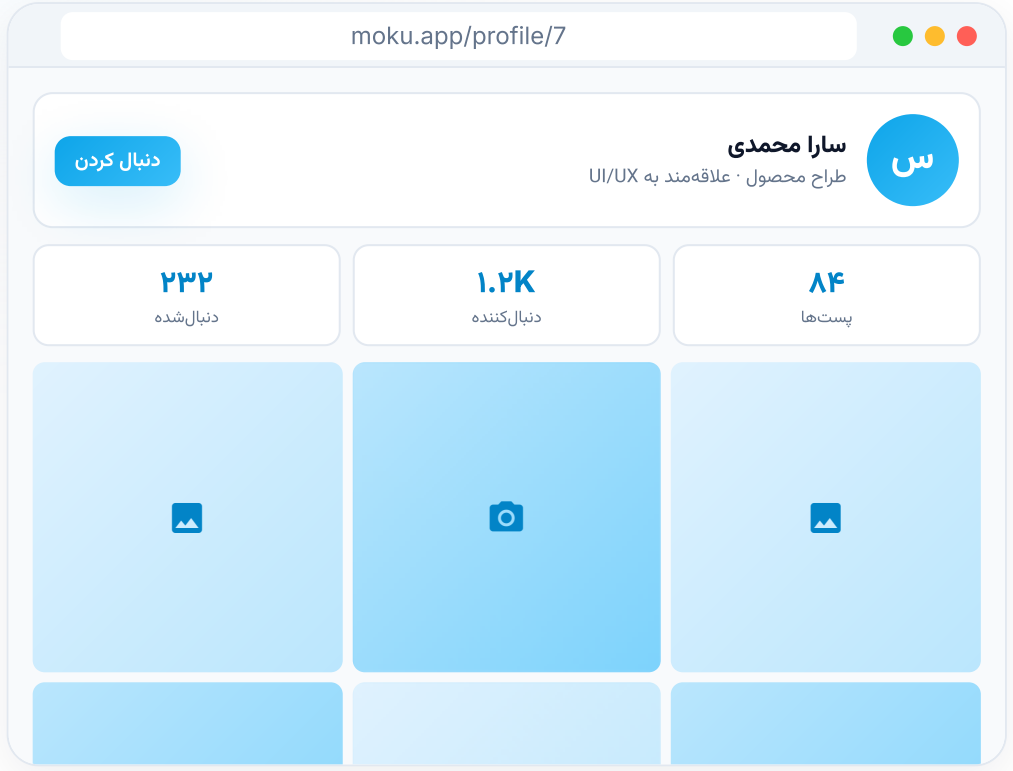
CartController FileController UserController Session

Flow

دریافت ProductController@show ← ID ← بارگذاری محصول + تصاویر ← نمایش ← افزودن به سبد (AJAX)

صفحه پروفایل کاربر

نمایش اطلاعات کاربر، پست‌ها و فعالیت‌ها



Flow

دریافت UserController@profile ← ID ← بارگذاری کاربر + پست‌ها ← نمایش گالری

مشخصات فنی صفحه

نام صفحه

پروفایل کاربر

ROUTE

`GET /profile/{id}`

CONTROLLER

UserController profile

METHOD

profile($id) — بارگذاری اطلاعات کاربر، پست‌ها، فالوورها و فعالیت‌ها

وظیفه

نمایش پروفایل عمومی کاربر و تاریخچه پست‌ها

ارتباط با سایر بخش‌ها

PostController · FollowController · FileController · Session

پنل مدیریت — داشبورد

مرکز کنترل کامل سیستم برای مدیران

Flow

بررسی نقش admin ← بارگذاری آمار از چند Model ← رندر View ← آماده برای ناوبری

مشخصات فنی صفحه

نام صفحه

داشبورد مدیریت

ROUTE

GET /admin

CONTROLLER

AdminController dashboard

METHOD

dashboard() — بارگذاری آمار کلی، نمودارها و دسترسی سریع به بخش‌ها

وظیفه

نمایش نمای کلی سیستم و ورود به بخش‌های مدیریتی

ارتباط با سایر بخش‌ها

UserController ProductController PostController OrderController

پرسش‌های‌شما— معماری و طراحی

پاسخ مفصل به پرسش‌ها

۰۱ چرا یک فریم‌ورک اختصاصی به جای Laravel یا Symfony ساختید؟

هدف اصلی من یادگیری عمیق مفاهیم بود. وقتی از فریم‌ورک آماده استفاده می‌کنید، جزئیات پنهان می‌ماند. ساخت MVC اختصاصی به من کمک کرد Routing، Middleware، ORM را در عمل درک کنم. علاوه بر این، کنترل کامل روی معماری امکان بهینه‌سازی دقیق برای نیازهای پروژه را فراهم کرد.

۰۲ الگوی MVC در پروژه چطور پیاده‌سازی شده است؟

index.php به‌عنوان Front Controller عمل می‌کند، URL را پارس و Controller/Method مناسب را فراخوانی می‌کند. هر Controller از Controller.php پایه ارث می‌برد که متدهای مشترک مانند render و redirect را فراهم می‌کند. Model.php یک ORM ساده با Query Builder است که روی Database.php ساخته شده و از Prepared Statements استفاده می‌کند. View.php قالب‌ها را با Layout و Partials رندر می‌کند.

۰۳ روابط بین ماژول‌ها چطور مدیریت می‌شود؟

هر ماژول Controller و Model مخصوص به خود را دارد، اما از طریق کلیدهای خارجی در دیتابیس به هم متصل‌اند. مثلاً ماژول Posts از ماژول Files برای آپلود تصاویر استفاده می‌کند، و ماژول Shop در Checkout با ماژول Auth برای احراز هویت تعامل دارد. این جداسازی ماژولار نگهداری و توسعه را ساده می‌کند.

۰۴ چرا معماری لایه‌ای انتخاب شد نه میکروسرویس؟

برای پروژه در این مقیاس، معماری یکپارچه (Monolithic) ساده‌تر و قابل‌انتشارتر است.
میکروسرویس برای مقیاس بسیار بزرگ مناسب است اما هزینه عملیاتی بالایی دارد. معماری لایه‌ای فعلی در صورت نیاز می‌تواند به میکروسرویس تجزیه شود، چون لایه‌ها از قبل جدا شده‌اند.

پرسش‌ها3– امنیت

لایه‌های امنیتی پروژه و نحوه پیاده‌سازی

۰۱ از SQL Injection چطور جلوگیری می‌کنید؟

تمام کوئری‌ها از طریق Prepared Statements در Database.php اجرا می‌شوند. هیچ متغیری مستقیماً در رشته SQL قرار نمی‌گیرد – همیشه به‌عنوان پارامتر به متدهای query یا execute پاس داده می‌شود. این روش حتی اگر ورودی کاربر مخرب باشد، آن را به‌عنوان داده تفسیر می‌کند نه دستور SQL.

۰۲ محافظت در برابر CSRF چگونه است؟

برای هر فرم یک توکن CSRF یکتا تولید می‌شود و در فیلد مخفی قرار می‌گیرد. در سرور، این توکن با مقدار ذخیره‌شده در Session مقایسه می‌شود. اگر نباشد یا ناهماهنگ باشد، درخواست رد می‌شود. این روش از حملاتی که در آن سایت مخرب فرم‌های پروژه ما را ارسال می‌کند، جلوگیری می‌کند.

۰۳ رمز عبور کاربران چطور ذخیره می‌شود؟

از PASSWORD_BCRYPT با cost ۱۲ استفاده می‌کنم. این الگوریتم به‌صورت خودکار salt تولید می‌کند و در هش ذخیره می‌کند. در ورود، password_verify هش را بررسی می‌کند. حتی اگر دیتابیس لو برود، رمزها قابل بازیابی نیستند. cost ۱۲ تعادل خوبی بین امنیت و سرعت است.

۰۴ Session چطور در برابر سرقت محافظت می‌شود؟

سه فلگ روی همه کوکی‌ها ست می‌شوند: HttpOnly که دسترسی JavaScript را مسدود می‌کند، Secure که فقط روی HTTPS ارسال می‌کند، و SameSite که از CSRF کمک می‌گیرد. علاوه بر این، Session ID پس از ورود موفق regenerate می‌شود تا از Session Fixation جلوگیری شود.

پرسش‌ها4– دیتابیس

طراحی schema، روابط و عملکرد دیتابیس

۰۱ طراحی schema دیتابیس چه الگویی دارد؟

طراحی relational با جداسازی بر اساس ماژول. هر ماژول جداول مخصوص به خود را دارد — messages، posts، products، users و غیره. روابط many-to-many از طریق جدول رابط مثل user_post_likes پیاده‌سازی می‌شوند. این جداسازی هم نرمال‌سازی را حفظ می‌کند، هم درک ساختار را ساده.

۰۲ Database.php چه abstractionهایی فراهم می‌کند؟

سه لایه abstraction: اتصال PDO به MySQL، Query Builder ساده با متدهایی مثل where و orderBy، و متدهای کمکی insert/update/delete/select. این لایه‌بندی کد را تمیز نگه می‌دارد و در صورت نیاز به تعویض دیتابیس، فقط یک لایه باید تغییر کند.

۰۳ روابط بین جداول چطور مدیریت می‌شوند؟

با کلیدهای خارجی و join. مثلاً جدول posts با user_id به users متصل است، و cart_items با user_id و product_id هم به users و هم به products متصل است. در مدل، متدهایی مثل user() یا products() روابط را بارگذاری می‌کنند. این رویکرد Eloquent-like است اما بسیار سبک‌تر.

۰۴ برای مقیاس‌پذیری چه برنامه‌ای دارید؟

در سطح فعلی MySQL کافی است. در صورت رشد، می‌توانیم اول indexing روی ستون‌های پراستفاده را اضافه کنیم، سپس Query Caching، و در نهایت Read Replicas. طراحی فعلی این مسیر را باز نگه می‌دارد چون لایه Database از ابتدا abstract است.

پرسش‌ها5– تجربه کاربری

تصمیم‌های طراحی، دسترسی‌پذیری و عملکرد

۰۱ چه تصمیماتی برای تجربه کاربری گرفتید؟

سه اصل اصلی: فضای سفید زیاد برای جلوگیری از شلوغی، فونت خوانا Vazirmatn برای پشتیبانی کامل از فارسی، و چیدمان منظم با کارت‌های شناور. تمام actions اصلی در دسترس یک کلیک هستند. رنگ‌بندی محدود به آبی آسمانی و سفید برای حفظ تمرکز کاربر روی محتوا.

۰۲ چطور مطمئن شدید رابط کاربری پاسخگو است؟

با استفاده از Flexbox و Grid در چیدمان‌های اصلی و media query برای نقاط شکست موبایل و تبلت. تمام فرم‌ها و کارت‌ها در عرض‌های مختلف تست شدند. تصاویر با max-width:100% و font-size با rem تنظیم شده تا در هر دستگاهی خوانا بماند.

۰۳ عملکرد صفحه اصلی چطور بهینه شده؟

سه روش: lazy-loading تصاویر پست‌ها، pagination روی فید اصلی به جای بارگذاری همه پست‌ها، و caching کوئری‌های پراستفاده. AJAX برای به‌روزرسانی نوتیفیکیشن و پیام‌ها بدون reload کامل صفحه استفاده می‌شود. این روش زمان بارگذاری اولیه را زیر ۱.۵ ثانیه نگه می‌دارد.

۰۴ دسترسی‌پذیری چطور در نظر گرفته شده؟

هر عنصر تعاملی label یا aria-label دارد. کنتراست رنگ‌ها مطابق WCAG AA است. ناوبری با کیبورد روی همه فرم‌ها کار می‌کند. آیکون‌ها همیشه با متن همراه‌اند تا کاربران با screen reader مشکلی نداشته باشند. این‌ها پایه‌ای هستند اما ضروری.

پرسش‌ها6 — توسعه و آینده

نقشه راه، استقرار و قابلیت‌های آینده

۰۱ بزرگترین چالش فنی در توسعه چه بود؟

طراحی abstraction مناسب برای Uploader.php که هم Local، هم FTP و هم Remote Storage را پشتیبانی کند. هرکدام API متفاوتی داشت و یکپارچه‌سازی آن‌ها بدون نشت جزئیات به لایه بالا، زمان‌بر بود. راه‌حل، تعریف یک interface مشترک و strategy pattern بود که در نهایت کد تمیز و قابل‌توسعه داد.

۰۲ پروژه چطور قابل استقرار در محیط واقعی است؟

نیازهای حداقلی: PHP ۸.۰ به بالا، MySQL ۵.۷ به بالا، و فعال بودن mod_rewrite برای Routing. آپلود فایل‌ها به سرور FTP جداگانه پیشنهاد می‌شود تا بار سرور اصلی کمتر شود. در صورت رشد، می‌توان از CDN برای فایل‌های استاتیک و Redis برای Session استفاده کرد.

۰۳ چه قابلیت‌هایی برای نسخه بعدی برنامه‌ریزی شده؟

سه اولویت: پیام‌رسان بلادرنگ واقعی با WebSocket به جای polling، سیستم پیشنهاد محصول بر اساس تاریخچه خرید، و اپلیکیشن موبایل با API موجود. معماری فعلی این توسعه‌ها را پشتیبانی می‌کند چون لایه‌ها از قبل جدا شده‌اند.

۰۴ چه درسی از این پروژه گرفتید؟

دو درس اصلی. اول، اهمیت طراحی از روز اول — صرف وقت روی معماری در ابتدا، صدها ساعت دیباگ را در ادامه ذخیره می‌کند. دوم، ارزش ساده نگه‌داشتن کد — استفاده از الگوهای پیچیده فقط زمانی که واقعاً نیاز است، نه به خاطر مد. این پروژه به من یاد داد که چگونه فکر کنم نه فقط کد بنویسم.