FashionVoid E-Commerce Platform

Interview Preparation Summary

Project: Full-Stack E-Commerce Platform
Live Site: https://www.fashionvoid.net
Status: Production-ready, tested with 15+ users

🎯 Project Overview (30 seconds)

Built a complete e-commerce platform from scratch using Next.js 14, TypeScript, and MySQL. Features include product management, secure authentication, Stripe/PayPal payments, comprehensive admin dashboard, and order processing. Successfully deployed to production and tested with real users.

💻 Technology Stack

Languages

- **TypeScript** (Primary - entire codebase)
- **JavaScript** (Runtime)
- **SQL** (Database queries)
- **CSS** (Tailwind CSS)

Frontend

- Next.js 14 (App Router, SSR)
- React 18
- TypeScript
- Tailwind CSS
- Framer Motion (animations)
- Zustand (state management)

Backend

- Next.js API Routes
- Prisma ORM
- NextAuth.js (authentication)
- bcryptjs (password hashing)

Services

- MySQL (database)
- Cloudinary (images)
- Stripe (payments)
- PayPal (payments)
- Railway (hosting)

✨ Key Features

E-Commerce Core

- ✅ Product catalog with variants (sizes, colors)

- ✅ Shopping cart (database-persisted)
- ✅ Multi-step checkout
- ✅ Order management & tracking
- ✅ Wishlist functionality
- ✅ Featured products (up to 4)

Admin Dashboard

- ✅ Product CRUD operations
- ✅ Order management (7 statuses)
- ✅ User management
- ✅ Discount code system
- ✅ Review approval workflow
- ✅ Inventory management
- ✅ Analytics dashboard
- ✅ Database migrations UI

User Features

- ✅ Registration & authentication
- ✅ User profiles with avatars
- ✅ Purchase history
- ✅ Address management
- ✅ Public profile pages
- ✅ Verification badges

Payment Integration

- ✅ Stripe checkout
- ✅ PayPal integration
- ✅ Webhook handling
- ✅ Order confirmations

Review System

- ✅ Multi-payment-method reviews
- ✅ Image uploads
- ✅ Admin approval workflow
- ✅ Order linking

🏗️ Architecture

```
Client (React/Next.js)
    ↓ HTTPS
Next.js App (Railway)
    ├── Frontend (SSR/Components)
    ├── API Routes (RESTful)
    └── Middleware (Auth/HTTPS)
    ↓
MySQL Database (Prisma ORM)
```

Design Patterns:

- Server-Side Rendering (SSR)

- RESTful API
- Component-based architecture
- ORM pattern (Prisma)
- JWT authentication

📊 Database Schema

10 Tables:

- Users (with verification)
- Products (with variants)
- Orders (7 statuses)
- OrderItems
- Addresses
- Wishlist
- Cart
- Reviews (with approval)
- DiscountCodes

Key Relationships:

- User → Orders (1:Many)
- Order → OrderItems (1:Many)
- Product → Variants (1:Many)
- User → Reviews (1:Many)

🔐 Security Features

- ✅ Password hashing (bcryptjs)
- ✅ HTTPS/SSL enforcement
- ✅ JWT sessions
- ✅ Role-based access control
- ✅ SQL injection prevention (Prisma)
- ✅ Webhook signature verification
- ✅ CSRF protection

🚀 Deployment

Platform: Railway
Domains: www.fashionvoid.net, fashionvoid.net
SSL: Automatic certificates
Build: Standalone Next.js output
Database: MySQL on Railway

Environment:

- Node.js 22.22.0
- Automatic deployments
- Health checks
- Logging & monitoring

💡 Technical Challenges Solved

1. Cart Persistence

Problem: Cart lost on refresh/logout
Solution: Database-backed cart that syncs across devices
Tech: Prisma, Zustand state management

2. Payment Webhooks

Problem: Reliable order creation from payments
Solution: Webhook signature verification + idempotency
Tech: Stripe/PayPal webhooks, event logging

3. Image Optimization

Problem: Large image uploads slow
Solution: Client-side compression before Cloudinary
Tech: browser-image-compression, Cloudinary API

4. Database Migrations on Railway

Problem: Can't run CLI migrations on Railway
Solution: Admin-accessible migration endpoint
Tech: Prisma $executeRawUnsafe, column checking

5. HTTPS Enforcement

Problem: Users accessing HTTP version
Solution: Middleware redirects HTTP → HTTPS
Tech: Next.js middleware, header checking

6. Review Approval Workflow

Problem: Different rules for different review types
Solution: Conditional approval based on payment method
Tech: Database flags, admin UI

📈 Project Metrics

- **API Endpoints:** 40+ RESTful routes
- **Database Tables:** 10 with relationships
- **Admin Dashboard:** 8 comprehensive tabs
- **Codebase:** 3000+ lines (admin dashboard alone)
- **Users Tested:** 15+ successful signups
- **Payment Methods:** 2 (Stripe + PayPal)
- **Order Statuses:** 7 different states
- **Featured Products:** Up to 4

🎓 What I Learned

- Full-stack development with Next.js

- Payment gateway integration (Stripe & PayPal)
- Database design and optimization
- Authentication & authorization patterns
- Production deployment (Railway)
- API design and security best practices
- State management (Zustand)
- Image handling and optimization
- Webhook processing
- Error handling and logging

💬 Interview Talking Points

"Tell me about a challenging feature you built"

Review System with Conditional Approval:

- Built review system supporting 4 payment methods
- Website reviews require completed orders (auto-approved)
- External reviews (PayPal/Revolut) allow anyone (needs approval)
- Implemented admin approval workflow
- Added image uploads and order linking
- Created verification badges for trusted users

"How did you handle security?"

- Password hashing with bcryptjs (10 rounds)
- HTTPS enforcement via middleware
- JWT sessions with HTTP-only cookies
- Role-based access control (admin/customer)
- SQL injection prevention via Prisma ORM
- Webhook signature verification
- Input validation on all endpoints

"Describe your database design"

- 10 tables with proper relationships
- Users table with roles and verification
- Products with variants (sizes/colors)
- Orders with 7 status states
- Reviews linked to orders and users
- Proper indexing for performance
- Foreign key constraints for data integrity

"How did you handle payments?"

- Integrated both Stripe and PayPal
- Created checkout sessions for Stripe
- PayPal Orders API for PayPal payments
- Webhook handlers for payment events
- Idempotency to prevent duplicate orders
- Email confirmations on successful payment
- Error handling and retry logic

"What was your deployment process?"

- Deployed to Railway platform
- Configured custom domains with SSL
- Set up environment variables
- Created admin migration system (can't access CLI)
- Implemented health checks
- Set up automatic deployments on git push
- Tested with 15+ users successfully

🔮 Future Enhancements (If Asked)

- Advanced product search with filters
- Email notification system
- Order tracking integration (shipping APIs)
- Enhanced analytics dashboard
- Multi-language support
- Mobile app (React Native)
- Product recommendations
- Customer support chat

📝 Code Examples (If Asked)

Authentication Check

```
// lib/auth-helpers.ts
export async function requireAuthApi() {
  const session = await getServerSession(authOptions)
  if (!session) {
    throw { status: 401, message: 'Unauthorized' }
  }
  return session
}
```

Database Query

```
// Using Prisma ORM
const products = await prisma.product.findMany({
  where: { inStock: true },
  include: { variants: true }
})
```

API Route

```
// app/api/products/route.ts
export async function GET() {
  const products = await prisma.product.findMany()
  return NextResponse.json(products)
}
```

✅ Quick Facts

- **Built:** From scratch (no templates)
- **Time:** Several months of development
- **Status:** Production-ready, live site
- **Testing:** 15+ users successfully signed up
- **Payments:** Real Stripe/PayPal integration
- **Deployment:** Railway with custom domains
- **Security:** HTTPS, password hashing, RBAC
- **Performance:** SSR, image optimization

🎯 Key Achievements

1. ✅ Built complete e-commerce platform solo
2. ✅ Integrated multiple payment gateways
3. ✅ Created comprehensive admin dashboard
4. ✅ Implemented secure authentication system
5. ✅ Deployed to production successfully
6. ✅ Tested with real users
7. ✅ Handled complex business logic (reviews, orders)
8. ✅ Optimized for performance and security

Remember: This is YOUR project. Speak confidently about what you built, the challenges you faced, and how you solved them. Be ready to dive deep into any area if asked!