

Project Initialization

- Install Laravel via Composer
- Choose and install starter kit with React
- Configure `.env` (DB, mail, queue)
- Run initial migrations
- Set up Tailwind CSS
- Initialize Git repository
- Create GitHub repo and push initial commit

Authentication Setup

- Verify Laravel auth routes are working
- Confirm user registration, login, logout
- Protect routes using auth middleware
- Create dummy admin user (email for notifications)

Database Design & Migrations

- Create `products` table migration (name, price, stock_quantity)
- Create `carts` table migration (user_id)
- Create `cart_items` table migration (cart_id, product_id, quantity)
- Create `orders` / `order_items` tables (for sales tracking)
- Run migrations
- Add database indexes where needed

Model Creation & Relationships

- Create `Product` model
- Create `Cart` model
- Create `CartItem` model
- Create `Order` and `OrderItem` models
- Define Eloquent relationships:
 - User → hasOne Cart
 - Cart → hasMany CartItems
 - CartItem → belongsTo Product
 - Order → belongsTo User
 - Order → hasMany OrderItems

Product Seeding

- Create product seeder
- Add sample products with varying stock levels
- Run seeder
- Verify products appear in database

Product Browsing (Frontend + Backend)

- Create product listing route
- Create product controller / Livewire component
- Fetch products from database
- Display product name, price, and stock quantity

- Disable add-to-cart if stock is zero

Cart Creation Logic

- Auto-create cart for authenticated user (on login or first add)
- Ensure one cart per user
- Fetch cart based on authenticated user
- Prevent cart access for unauthenticated users

Add to Cart Flow

- Create add-to-cart endpoint/action
- Validate product exists
- Validate stock availability
- If product exists in cart, increment quantity
- If not, create new cart item
- Persist cart item in database
- Return updated cart state to UI

Update Cart Item Quantity

- Create update-quantity endpoint/action
- Validate quantity is positive integer
- Validate stock availability
- Update cart item quantity
- Persist changes
- Refresh cart totals on UI

Remove Cart Item

- Create remove-item endpoint/action
- Delete cart item from database
- Update cart totals
- Refresh UI state

Checkout / Purchase Logic (Minimal)

- Create checkout action
- Loop through cart items
- Deduct stock from products
- Create order record
- Create order items
- Clear cart after successful checkout

Low Stock Detection

- Define low-stock threshold (e.g., `LOW_STOCK_THRESHOLD` \leq units)
- Check stock after each purchase
- Identify products that cross threshold

Low Stock Email Job

- Create Laravel Job (LowStockNotificationJob)
- Pass product data to job

- Configure mail template
- Send email to dummy admin
- Dispatch job when threshold is reached

Queue Configuration

- Configure queue driver (database)
- Create jobs table
- Run queue worker
- Verify async email dispatch

Daily Sales Tracking

- Store product sales per order
- Record quantity sold per product
- Timestamp sales entries

Scheduled Daily Report Job

- Create scheduled job/command
- Query products sold today
- Aggregate quantities per product
- Generate email report content
- Send email to dummy admin

Scheduler Setup

- Register scheduled job
- Set cron to run daily in the evening
- Test scheduled job manually

Frontend Cart UI

- Create cart page/component
- Display cart items with quantity controls
- Show subtotal / total
- Handle empty cart state
- Style using Tailwind CSS

Validation & Edge Cases

- Prevent adding more than available stock
- Handle concurrent stock updates
- Handle deleted products
- Graceful error messages

Testing (Light but Practical)

- Test auth-protected routes
- Test add/update/remove cart flow
- Test stock deduction

- Test low-stock email dispatch
- Test daily report job

Cleanup & Best Practices

- Use form requests for validation
- Use services/actions for business logic
- Avoid fat controllers
- Follow Laravel naming conventions

Documentation

- Write README:
 - Setup instructions
 - Queue & scheduler setup
 - Dummy admin email

Final Review

- Run project from scratch
- Verify no hardcoded data
- Push final changes to GitHub
- Ensure repo is clean and readable