🛒 PROJECT NAME: MarketNest – A Modern Multi-Vendor Marketplace

Users can buy products, become sellers, and sellers can manage inventory + orders.
Admin controls everything.
Payment method: **Cash on Delivery (COD)** only.
Tech Stack: **Next.js 15, Clerk, Neon PostgreSQL, Drizzle ORM, ShadCN, Tailwind, Cloudinary.**

📌 1. EXECUTIVE SUMMARY

MarketNest is a multi-vendor eCommerce marketplace where users can sign up, browse products, purchase using Cash on Delivery, and optionally become sellers to manage their own products and orders. Admins maintain full platform control using a dedicated admin panel.

The MVP focuses on:

- Fast seller onboarding
- Simple COD checkout
- Clean UI
- Strong role-based access
- Powerful seller & admin dashboards

📌 2. SCOPE OF WORK (MVP)

Users

- Create account (Clerk)
- Browse products
- Search + filter
- Add to cart
- Checkout (COD)

- View their orders

Sellers

- Apply to become seller
- Create/edit/delete products
- Manage inventory
- View orders received on their products
- Update order status (Pending → Confirmed → Shipped → Delivered)

Admin

- Dashboard with platform statistics
- Approve sellers
- Manage users
- Manage products
- Manage categories
- View all orders
- Toggle seller/user activity status

📌 3. SYSTEM ARCHITECTURE

Frontend

- Next.js 15 App Router
- ShadCN UI + Tailwind CSS
- React Server Components
- Protected Layouts (middleware)

Backend

- Next.js server actions / API routes
- Drizzle ORM
- Clerk Webhooks for user creation

Backend

- Next.js server actions / API routes
- Drizzle models: User, Seller, Product, Order, OrderItem, Category, Role, Reviews, Categories

Image Upload

- Cloudinary or Uploadcare

Security

- RLS (Row-level access) via server middleware
- JWT (handled by Clerk)
- Role-based API authorization

📌 4. DATABASE SCHEMA (Drizzle MODELS)

```
// ------------------ ENUMS ------------------

export const ROLE_ENUM = pgEnum("role", ["USER", "ADMIN",
"SELLER"]);

export const PRODUCT_STATUS_ENUM = pgEnum("product_status",
[
  "ACTIVE",
  "INACTIVE",
  "OUT_OF_STOCK",
  "DRAFT",
]);

export const RATING_ENUM = pgEnum("rating", ["1", "2", "3",
"4", "5"]);


// ------------------ USERS ------------------

export const users = pgTable("users", {
  id: uuid("id").defaultRandom().primaryKey(),
  clerkId: varchar("clerk_id", { length: 255
}).notNull().unique(),
  fullName: varchar("full_name", { length: 255
}).notNull(),
  email: text("email").notNull().unique(),
  imageUrl: text("image_url"),
  phone: varchar("phone", { length: 20 }),
  emailVerified: boolean("email_verified").default(false),
  onboardingCompleted:
boolean("onboarding_completed").default(false),

  role: ROLE_ENUM("role").default("USER"),
  isActive: boolean("is_active").default(true),

  lastLoginAt: timestamp("last_login_at", { withTimezone:
true }),
  createdAt: timestamp("created_at", { withTimezone: true
}).defaultNow(),
  updatedAt: timestamp("updated_at", { withTimezone: true
}).defaultNow(),
});


// ------------------ USER ADDRESSES ------------------
// (Users can save multiple addresses for delivery)
```

```
export const addresses = pgTable("addresses", {
  id: uuid("id").defaultRandom().primaryKey(),
  userId: uuid("user_id").notNull().references(() =>
users.id, {
    onDelete: "cascade",
  }),

  fullName: varchar("full_name", { length: 255
}).notNull(),
  phone: varchar("phone", { length: 20 }).notNull(),
  country: varchar("country", { length: 150 }).notNull(),
  city: varchar("city", { length: 150 }).notNull(),
  street: text("street").notNull(),
  postalCode: varchar("postal_code", { length: 20 }),

  isDefault: boolean("is_default").default(false),

  createdAt: timestamp("created_at", { withTimezone: true
}).defaultNow(),
});


// ------------------ SELLERS ------------------

export const sellers = pgTable("sellers", {
  id: uuid("id").defaultRandom().primaryKey(),
  userId: uuid("user_id").notNull().references(() =>
users.id, {
    onDelete: "cascade",
  }),

  username: varchar("username", { length: 255
}).notNull().unique(),
  shopName: varchar("shop_name", { length: 255
}).notNull(),
  shopSlug: varchar("shop_slug", { length: 255 }).unique(),

  logoUrl: text("logo_url"),
  bannerUrl: text("banner_url"),

  bio: text("bio").notNull(),
  phone: varchar("phone", { length: 20 }).notNull(),
  address: text("address").notNull(),
  contactEmail: text("contact_email").notNull().unique(),

  rating: numeric("rating").default("0"),
  totalProducts: integer("total_products").default(0),
  totalSales: integer("total_sales").default(0),

  isVerified: boolean("is_verified").default(false),
  createdAt: timestamp("created_at", { withTimezone: true
}).defaultNow(),
});


// ------------------ CATEGORIES ------------------

export const categories = pgTable("categories", {
  id: uuid("id").defaultRandom().primaryKey(),
  name: varchar("name", { length: 255
}).notNull().unique(),
  slug: varchar("slug", { length: 255
}).notNull().unique(),

  parentCategoryId:
uuid("parent_category_id").references(() => categories.id,
{
    onDelete: "set null",
  }),

  description: text("description"),
  imageUrl: text("image_url"),
  createdAt: timestamp("created_at", { withTimezone: true
}).defaultNow(),
});


// ------------------ PRODUCTS ------------------

export const products = pgTable("products", {
  id: uuid("id").defaultRandom().primaryKey(),

  name: varchar("name", { length: 255 }).notNull(),
  slug: varchar("slug", { length: 255
}).notNull().unique(),
  brand: varchar("brand", { length: 255 }),

  description: text("description").notNull(),
  tags: jsonb("tags").$type<string[]>().default([]),

  price: integer("price").notNull(),
  discountPrice: integer("discount_price"),

  images: jsonb("images").$type<string[]>().notNull(), //
up to 4 URLs

  stock: integer("stock").default(0),
  status: PRODUCT_STATUS_ENUM("status").default("ACTIVE"),
  views: integer("views").default(0),
  salesCount: integer("sales_count").default(0),

  averageRating: numeric("average_rating").default("0"),
  totalReviews: integer("total_reviews").default(0),

  weight: numeric("weight"),
  dimensions: jsonb("dimensions").$type<{ length: number;
width: number; height: number }>(),

  returnPolicy: text("return_policy"),
  warranty: text("warranty"),

  // Relations
  sellerId: uuid("seller_id").references(() => sellers.id,
{
    onDelete: "set null",
  }),

  createdByAdmin:
boolean("created_by_admin").default(false),

  categoryId: uuid("category_id").references(() =>
categories.id, {
    onDelete: "set null",
  }),

  createdAt: timestamp("created_at", { withTimezone: true
}).defaultNow(),
  updatedAt: timestamp("updated_at", { withTimezone: true
}).defaultNow(),
});
```

```
// ------------------ REVIEWS ------------------

export const reviews = pgTable("reviews", {
  id: uuid("id").defaultRandom().primaryKey(),

  userId: uuid("user_id").notNull().references(() =>
users.id, {
    onDelete: "cascade",
  }),

  productId: uuid("product_id").notNull().references(() =>
products.id, {
    onDelete: "cascade",
  }),

  rating: RATING_ENUM("rating").notNull(),
  comment: text("comment"),

  helpful: integer("helpful").default(0),

  createdAt: timestamp("created_at", { withTimezone: true
}).defaultNow(),
}, (table) => ({
  uniqueUserProductReview: unique().on(table.userId,
table.productId),
}));
```

📌 5. USER FLOW DIAGRAMS (TEXT FORMAT)

Buyer Flow

1. Visit site
2. Browse products
3. Search / filter
4. Add to cart
5. Checkout (COD only)
6. Order placed
7. View order history

Seller Flow

1. User → “Become Seller”
2. Admin approves
3. Seller Dashboard opens
4. Add product
5. Orders come in
6. Update order status

Admin Flow

1. Login
2. Admin dashboard
3. View analytics
4. Approve sellers
5. Manage categories
6. Manage products
7. See orders across platform

📌 8. 10-DAY DEVELOPMENT ROADMAP (SIMPLE + REALISTIC)

DAY 1 — Project Setup

- Initialize Next.js 15

- Install Tailwind + ShadCN
- Connect Clerk
- Setup Neon PostgreSQL
- Init Drizzle ORM

DAY 2 — Auth + Middleware

- Create Clerk auth flows
- Add role-based middleware
- Setup protected routes
- Create layout for admin/seller

DAY 3 — Database Models

- Build Drizzle ORM schema
- Migrate DB
- Seed categories
- Test relational links

DAY 4 — User Interface (Public)

- Home page
- Product listing
- Categories filter
- Product details page

DAY 5 — Cart + Checkout (COD)

- Cart context
- Add to cart UI

- COD order placement
- Save order & order items

DAY 6 — Seller Dashboard

- Seller onboarding request page
- Product CRUD pages
- Orders table
- Order status update

DAY 7 — Admin Panel

- Admin dashboard
- Approve sellers
- Manage users
- Manage categories
- Manage products

DAY 8 — Analytics

- Seller revenue graph
- Admin platform metrics
- Best-selling products report

DAY 9 — Polish

- Error handling
- Form validation
- Skeleton loaders

- Mobile responsive

DAY 10 — Deployment

- Deploy to Vercel
- Set all environment variables
- Test full system
- Fix last-minute bugs