# Day 2 Activities: Transitioning to Technical Planning

---

## Technical Requirements for My Electronics Marketplace

## 1. Define Technical Requirements

To transition into technical planning for your e-commerce marketplace, here are the key technical requirements:

**FRONTEND REQUIREMENTS:**

- **Framework:** Next.js 14 with the app directory for server-side rendering (SSR) and static site generation (SSG).
- **Styling:** Tailwind CSS for responsive and customizable design.
- **Typescript:** Type-safe codebase to reduce runtime errors.
- **User Interface:** Dynamic product listings, filtering options, search functionality, and a user-friendly shopping cart system.

**BACKEND REQUIREMENTS:**

- Sanity CMS: Manage dynamic content for products, categories, and promotional banners.
- API Layer: APIs to fetch product data, handle user authentication, and manage orders.

## DATABASE REQUIREMENTS:

- Sanity CMS will handle structured data for products, categories, and user-generated content.

**AUTHENTICATION:**

- Use providers like NextAuth.js or a custom JWT-based system for secure login/signup.

**PAYMENT GATEWAY:**

- Integrate with Stripe or PayPal for secure and seamless transactions.

**PERFORMANCE OPTIMIZATION:**

- Leverage Next.js features like ISR (Incremental Static Regeneration) and edge caching for performance.
- Optimize images with Next.js Image component.

## Security:

- HTTPS for secure communication.
- Secure API endpoints and implement rate limiting.

---

# 2. Design System Architecture

A high-level system architecture for your marketplace:

1. ***Frontend:***

- Framework: Next.js 14 App Directory.
- Routing: File-based routing for products, categories, and user pages.
- State Management: React Context or Zustand for local state, and SWR or React Query for data fetching.

2. ***Backend:***

- Content Management: Sanity CMS to store and manage product data.
- API Layer: Build RESTful APIs using Next.js API routes or a separate serverless framework like AWS Lambda.

3. ***Database:***

- Primary Database: Sanity CMS.
- Search Engine: Optionally integrate Algolia or Elasticsearch for fast and relevant product searches.

4. ***Third-Party Integrations:***

- Payment Gateway: Stripe for secure payment processing.
- Email Service: SendGrid or AWS SES for transactional emails.

5. ***Hosting & Deployment:***

   - Vercel Host the Next.js app for optimized performance and scalability.

# 3. Plan API Requirements

APIs for the e-commerce marketplace can be broadly divided into these categories:

1. ***User APIs:***

- POST /api/auth/register: Register a new user.
- POST /api/auth/login: User login.
- GET /api/auth/logout: User logout.
- GET /api/users/profile: Fetch user profile data.

2. ***Product APIs:***

- GET /api/products: List all products.
- GET /api/products/:id: Get product details.

- GET /api/products/search: Search products with filters.

- POST /api/products: Add a new product (admin only).

- PATCH /api/products/:id: Update product details (admin only).

### 3. Category APIs:

- GET /api/categories: Fetch all categories.

- POST /api/categories: Add a new category (admin only).

### 4. Order APIs:

- POST /api/orders: Create a new order.

- GET /api/orders/:id: Fetch order details.

- GET /api/orders/user: Fetch orders for the logged-in user.

### 5. Payment APIs:

- POST /api/payments/checkout: Initiate payment with Stripe/PayPal.

- GET /api/payments/verify: Verify payment status.

6. ***Miscellaneous APIs:***

- GET /api/banners: Fetch homepage banners and promotions.

# 4. Write Technical Documentation

### Create detailed technical documentation to guide the development team:

1. ***Project Overview:***

- Brief introduction to the marketplace's purpose.

- Technologies used: Next.js, Tailwind CSS, Sanity CMS, TypeScript.

2. **System Design:**

- Diagram of system architecture (frontend, backend, CMS, database, external APIs).

- Explanation of how components interact.

**3. API Documentation:**

- List all endpoints with descriptions, request/response formats, and status codes.

**4. Development Workflow:**

- Branching strategy (e.g., GitFlow).

- Code review process.

- CI/CD pipeline configuration (e.g., Vercel or GitHub Actions).

**5. Security Policies:**

- Best practices for authentication and data security.

# 5. Collaborate and Refine

1. ***Team Collaboration:***

- Set up regular standups and meetings to discuss progress and blockers.
- Use tools like Jira or Trello for task management.

2. ***Feedback Loop:***

- Conduct code reviews and architecture walkthroughs.
- Test APIs and features with tools like Postman.

3. **Iteration:**

- Refine the architecture and documentation based on team feedback.
- Perform performance tests to identify bottlenecks and improve scalability.