High Level Design (HLD)

“E-Commerce Application Clone”

Document Version Control

Version	Date	Description	Author
1.0	20-Oct-2024	Abstract, Introduction, General Description	Rajan Kumar
1.1	21-Oct-2024	Design Detail, KPI, Deployment	Rajan Kumar
1.2	22-Oct-2024	Final Revision	Rajan Kumar

Contents

Abstract

This document provides a high-level design for an E-Commerce Application Clone that allows users to view a collection of products, see product details, add items to their cart, and complete a purchase through a checkout process. The primary focus of this design is to create an intuitive and optimized user experience while maintaining functionality and scalability.

---

1. Introduction

1.1 **Why this High-Level Design Document?**

This document serves as a blueprint for developing the E-Commerce Application Clone. It outlines the structural and functional elements of the application, ensuring clear communication between stakeholders and the development team. The document details how various components interact, ensuring a smooth flow of operations from product display to checkout.

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions before coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

• Present all of the design aspects and define them in detail

• Describe the user interface being implemented

• Describe the hardware and software interfaces

• Describe the performance requirements

• Include design features and the architecture of the project

-Security

-Reliability

-Maintainability

-Portability

-Reusability

-Application compatibility

-Resource utilization

-Serviceability

1.2 **Scope**

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly technical terms which should be understandable to the administrators of the system.

This design document will cover the following aspects:

- Homepage to display available products
- Collection/category page for product types
- Cart management page to review added products
- Checkout page for finalizing orders
- Functional architecture and optimization strategies
- Key performance indicators (KPIs) for measuring success

2. General Description

2.1 **Product Perspective & Problem Statement**

The E-Commerce Application Clone will emulate the core features of a standard e-commerce platform, enabling users to:

- Browse through a collection of products on the homepage
- View products based on categories (electronics, fashion, etc.)
- Add products to their shopping cart
- Review and edit their shopping cart before proceeding to checkout
- Complete an order through the checkout page

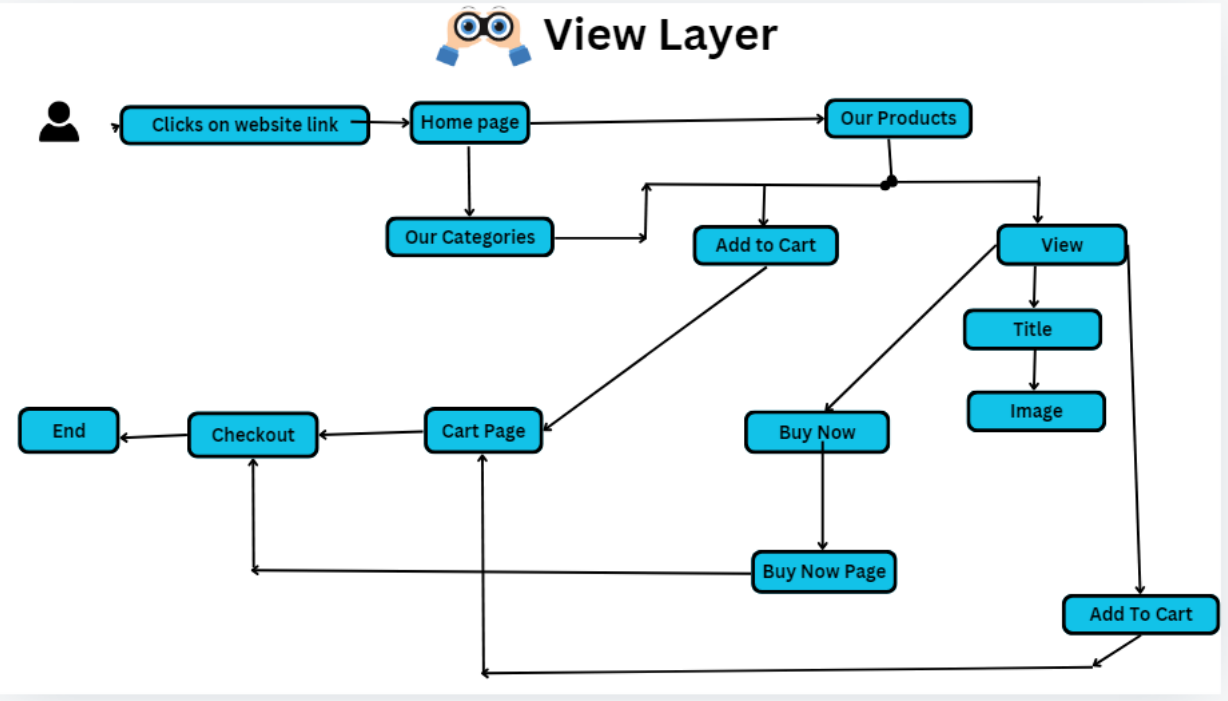
This system will also support basic user flows such as product search, filtering by category, and handling cart interactions.

2.2 **Tools Used**

The following tools and technologies will be used to implement the project:

- **Frontend:** HTML, CSS, JavaScript, React.js (for UI components and dynamic interactions)
- **Additional Tools:** Git for version control, deployment through services like Vercel or Heroku.

3. Design Detail



3.1 **Functional Architecture**

- **Homepage:**
  - Displays all available products.
  - Includes search functionality and filtering by categories.
  - Each product card will contain the product image, name, price, and "Add to Cart" button.

- **Category/Collection Page:**
  - Displays products filtered by category.
  - The user can browse through a specific collection (e.g., electronics, clothing).
  - Products displayed similarly to the homepage.
- **Product Page:**
  - On clicking a product, the user is taken to a detailed view of the product.
  - Displays product image, description, price, and an "Add to Cart" button.
- **Cart Page:**
  - Lists all products added to the cart.
  - Allows users to modify the quantity or remove products.
  - Displays total cost and a "Proceed to Checkout" button.

- **Checkout Page:**
    - Allows users to enter shipping details and payment information.
    - Displays a final summary of the order before placing it.
- **Navigation:**
    - Navbar for easy access to Home, Collections, Cart, and Checkout pages.
    - Responsive design for mobile, tablet, and desktop views.

3.2 **Optimization**

- **Lazy Loading:** Products and categories will be lazy-loaded to improve performance, ensuring only visible products are loaded at any time.
- **Caching:** Utilize browser caching and CDN for static content to improve load times.
- **Database Indexing:** Ensure indexes are set on frequently queried fields such as product names, categories, and user data.
- **Pagination:** Implement pagination or infinite scroll for product listings to optimize performance with large product catalogs.

4. Key Performance Indicators (KPIs)

4.1 **KPIs (Key Performance Indicators)**

- **Page Load Speed:** Target load times of under 3 seconds for homepage and product pages.
- **Cart Abandonment Rate:** Track and aim to reduce the percentage of users who add items to the cart but don't complete checkout.
- **Conversion Rate:** Measure the percentage of users who make a purchase after adding items to the cart.
- **Error Rate:** Monitor and aim to minimize the occurrence of errors (500 or 404 responses) during user interactions.

5. Deployment

The deployment process will involve the following steps:

- **Continuous Integration (CI):** Code changes are pushed to the repository, tested using automated test suites.
- **Continuous Deployment (CD):** After successful tests, the application is deployed to a staging environment for review.
- **Production Deployment:** The final version of the application is deployed to production servers (e.g., AWS, Vercel, Heroku).
- **Monitoring:** Tools like Google Analytics or New Relic will be used to monitor app performance, user behavior, and errors in real-time.