Architecture Design

E-Commerce Application Clone

Aman Negi

Contents

1. Abstract:

This project entails the design and development of a frontend-only E-Commerce Application Clone aimed at providing users with a seamless online shopping experience. The application allows users to browse through a collection of products,

explore individual items, and conveniently add desired products to their virtual shopping cart. Key functionalities include a dynamic homepage showcasing available products, category pages for organized browsing, and a user-friendly cart page for managing selected items. Leveraging modern frontend technologies such as React.js and React Router, the application ensures smooth navigation and responsive design across various devices. Data mocking techniques simulate backend API calls, enabling the presentation of sample product data, while local storage facilitates temporary cart item storage. Emphasizing user experience, the design integrates accessibility features, performance optimizations. Development tools such as ESLint, Prettier, and version control systems ensure code quality and collaboration efficiency. Testing strategies encompass unit testing, end-to-end testing, and accessibility testing to guarantee functionality and compliance with standards. Finally, deployment involves hosting the frontend application on static site hosting platforms with continuous integration and deployment pipelines for automated testing and deployment processes. This abstract outlines the comprehensive approach to building a robust frontend solution for an E-Commerce Application Clone, catering to modern development standards and user expectations.

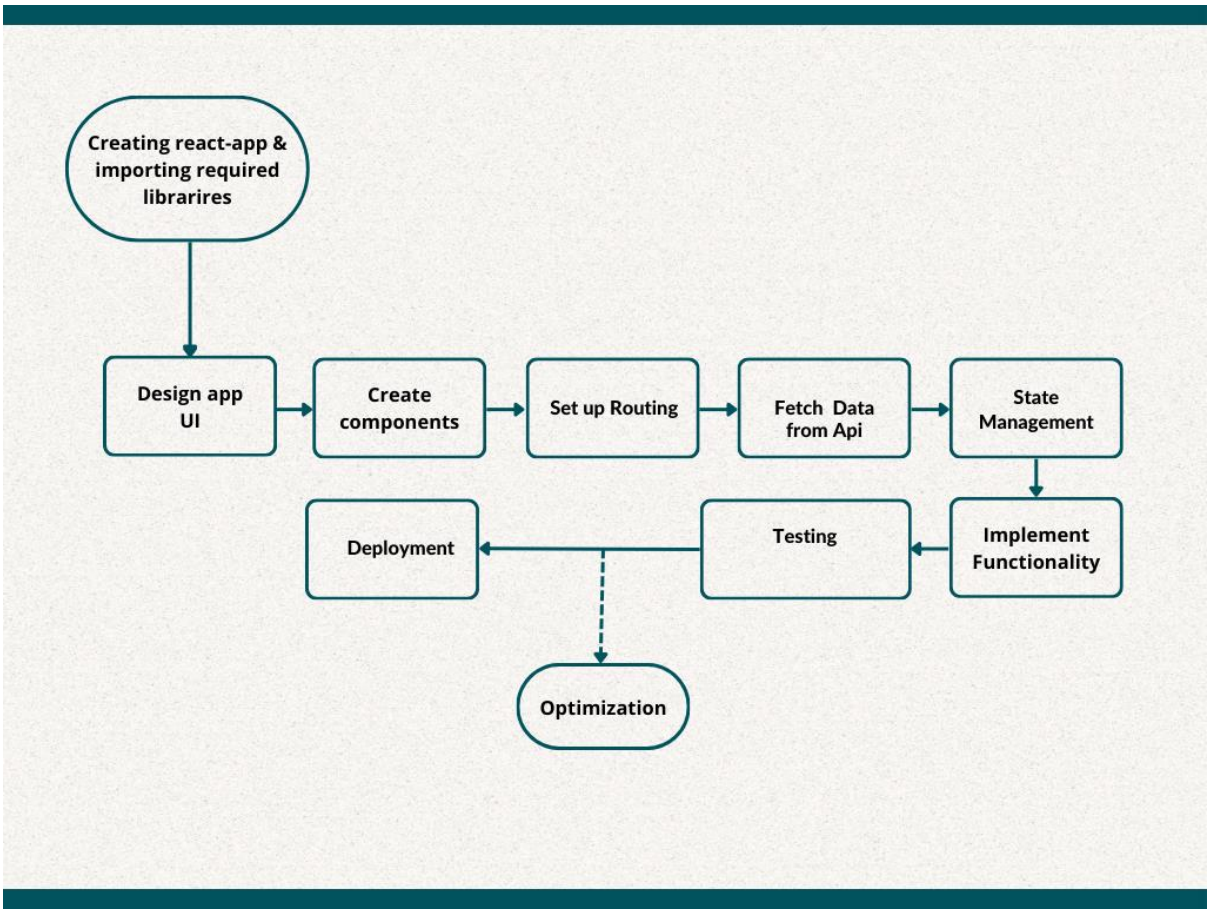
1. Introduction

1.1 Why this Architecture Design Document ?

The main objective of the Architecture design documentation is to provide the internal logic understanding of the Rental Bike share demand prediction code. The

Architecture design documentation is designed in such a way that the programmer can directly code after reading each module description in the documentation

2. Architecture



3. Architecture Design

3.1 Data Collection

The dataset was taken from API

https://api.pujakaitem.com/api/products

3.2 Data Description

After making api call you will get a JSON array containing information about various products available in an ecommerce store. Here's a description of the data:

Product Information: Each object in the array represents a product available for sale.

Product details include:

- id: Unique identifier for the product.
- name: Name of the product (e.g., iPhone X, Samsung S20).
- company: Brand or manufacturer of the product (e.g., Apple, Samsung).
- price: Price of the product in the specified currency.
- colors: Array of color options available for the product.
- image: URL of the product image.
- description: Detailed description of the product, including features, specifications, and highlights.
- category: Category of the product (e.g., mobile, laptop, accessories).
- shipping: Boolean indicating whether shipping is available for the product.
- featured: Boolean indicating whether the product is featured or highlighted.

3.3. Components

- The frontend is structured using a component-based architecture, with reusable UI components for various elements such as headers, product listings, cart, and checkout.
- Components are organized into a hierarchy, with higher-level components composing lower-level ones to create complex UI layouts.

3.4. State Management

- State management is handled using React Context API for managing global application state.
- Context providers are used to encapsulate state related to user authentication, cart items, and other application data.

3.5. Routing

- React Router is used for client-side routing, enabling navigation between different pages and components within the application.
- Routes are defined to map URLs to corresponding components, allowing for a single-page application (SPA) experience.

3.6. Styling

- CSS modules or styled-components are used for styling components, providing scoped styles and better maintainability.
- Responsive design principles are applied to ensure the application is accessible and usable across different devices and screen sizes.