ECOMMERCE WEBSITE

DAISYDEALS

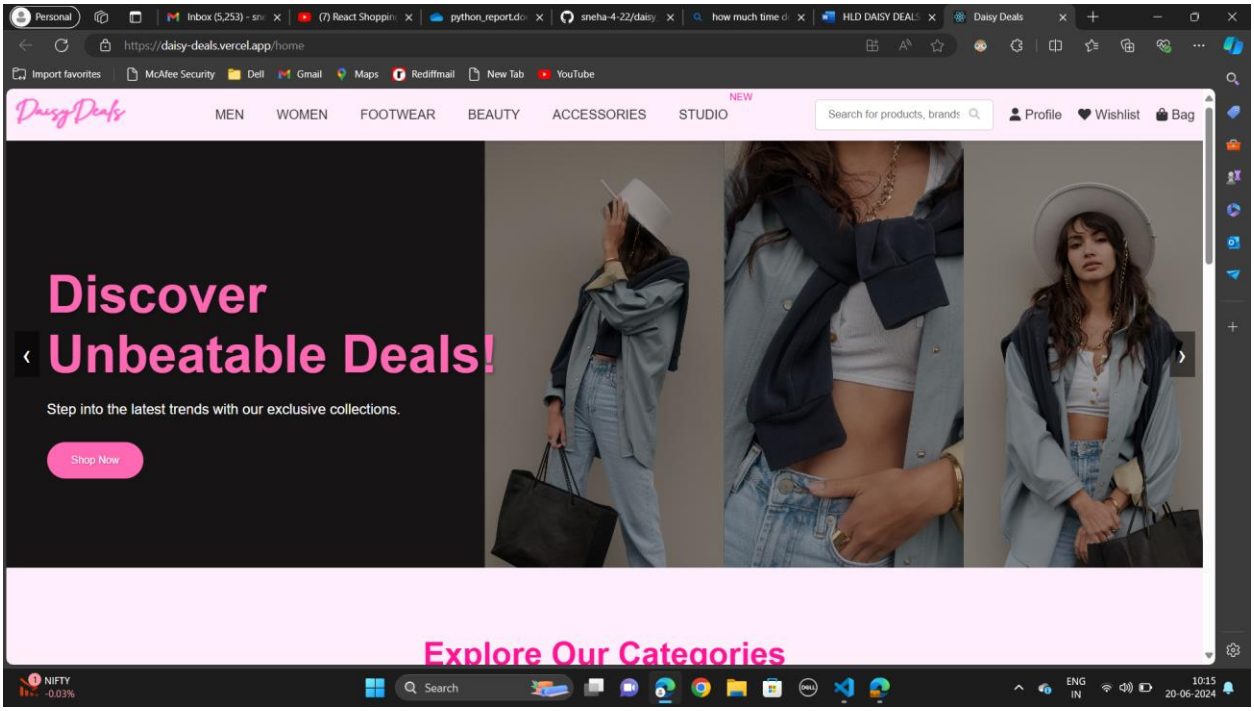
High Level Design (HLD) 

Date Issued	Version	Description	Author
15/06/24	1	Abtract and intro	sneha
16/0624	2	design	sneha
19/06/24	3	medthodolgy	sneha
20/06/24	4	conclusion	sneha
20/06/24	5	Final	sneha

Contents

Abstract

Daisy Deals is an e-commerce application designed to showcase products, allow users to add products to their cart, and complete purchases. This document outlines the high-level design (HLD) for creating a clone of this application using React for the frontend, Node.js for the backend, and various supporting tools and technologies.



Introduction

1.1 Why this High-Level Design Document?

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 prior to 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.
- List and describe the non-functional attributes like:
    - 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.

1.3 Definitions

Term	Description
UI	User Interface
IDE	Integrated Development Environment
API	Application Programming Interface
DBMS	Database Management System
AWS	Amazon Web Services
MVC	Model View Controller architecture

General Description

2.1 Product Perspective

Daisy Deals is an e-commerce application designed to provide users with a platform to browse, search, and purchase products. It includes functionalities such as a homepage, product detail pages, a cart, and a checkout process.

2.2 Problem Statement

Design an e-commerce application where users can:

- Browse a collection of products.
- View individual product details.
- Add products to a cart.
- Place orders.

2.3 Proposed Solution

The proposed solution is to build an e-commerce application with the following functionalities:

- A homepage showcasing available products.
- Category/Collection pages displaying specific types of products.
- A cart page where users can view and manage items added to their cart.
- A checkout page for placing orders.

2.4 Further Improvements

Future enhancements can include:

- User authentication and profiles.
- Recommendation systems.
- Advanced search and filter options.
- Integration with various payment gateways.

2.5 Technical Requirements

- Web framework: React (Frontend)
- Backend: Node.js, Express.js
- Database: MongoDB
- Hosting: AWS
- Version control: GitHub

2.6 Data Requirements

- Product data including images, descriptions, prices, and categories.
- User data for accounts and order history.
- Transaction data for orders and payments.

2.7 Tools Used

- IDE: Visual Studio Code
- Frontend Libraries: React, Redux
- Backend Libraries: Node.js, Express.js
- Database: MongoDB
- Deployment: Vercel

- Version control: GitHub
- Authentication: JWT (JSON Web Tokens)
- Styling: CSS, Sass, Bootstrap
- State Management: Redux

2.8 Constraints

The system must be user-friendly, highly responsive, and able to handle concurrent users and transactions efficiently.

2.9 Assumptions

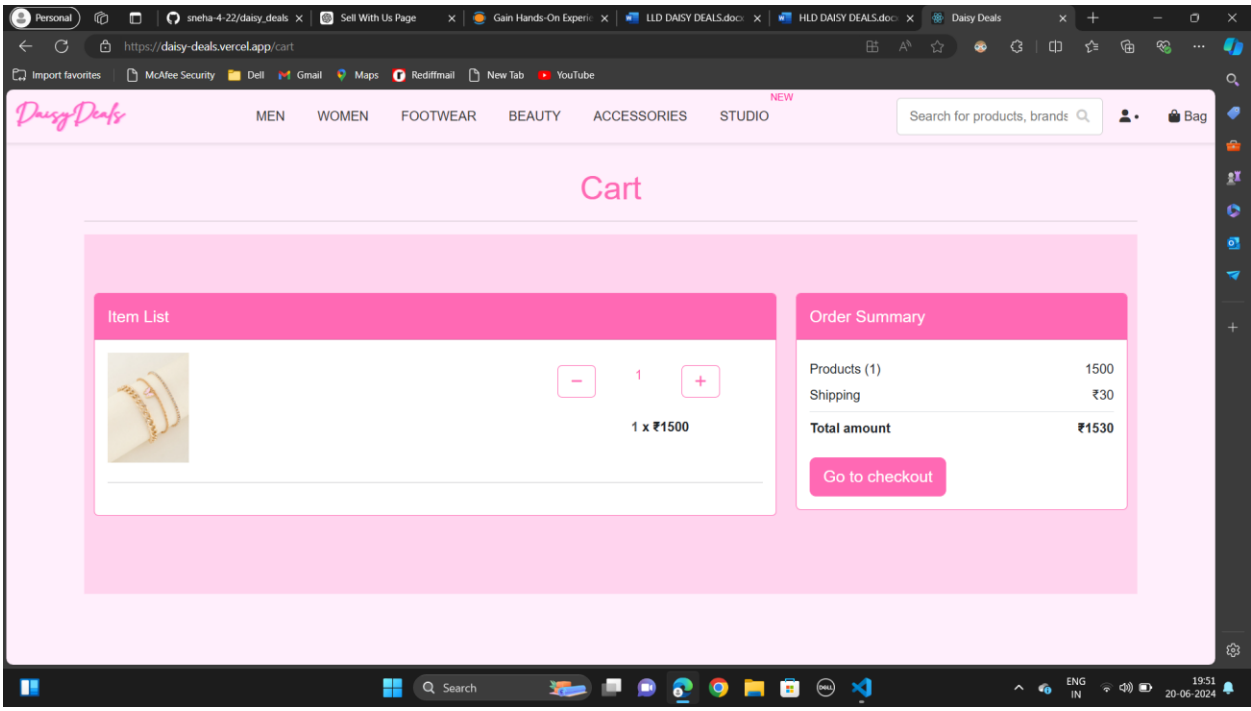
- The project assumes that all components will work seamlessly together.
- Users will have basic knowledge of navigating e-commerce websites.

Design Details

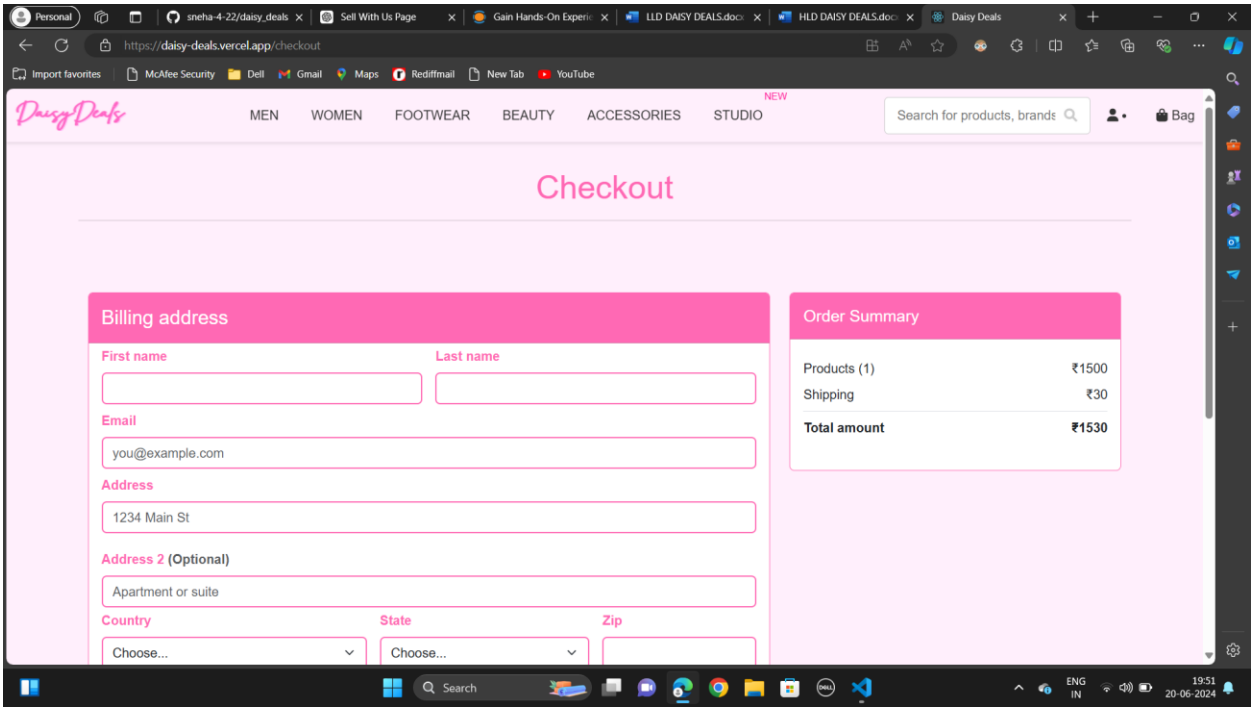
3.1 Process Flow

3.1.1 Frontend Design and Implementation

- **Homepage**:
  - Display featured products.
  - Implement a search bar and category navigation.
- **Product Detail Page**:
  - Show detailed information about the product.
  - Provide options to add the product to the cart.
- **Cart Page**:
  - Display products added to the cart.
  - Allow users to update quantities and remove items.



- **Checkout Page**:
  - Collect shipping information and payment details.
  - Provide order summary and place the order.



3.1.2 Backend Design and Implementation

- **API Development**:
  - Develop RESTful APIs using Express.js.
  - Implement endpoints for products, cart, orders, and user authentication.
- **Database Schema**:
  - Design MongoDB schemas for products, users, and orders.
  - Ensure data validation and integrity.

7. **Authentication**:
   - Use JWT for secure authentication.
   - Implement middleware for protected routes.

3.1.3 Deployment Process

- **Local Development**:
  - Set up development environment with Node.js and React.
  - Test all functionalities locally.
- **Staging Environment**:
  - Deploy to a staging environment on AWS.
  - Conduct user acceptance testing (UAT).
- **Production Deployment**:
  - Deploy to the production environment.
  - Monitor performance and handle scaling.

3.2 Event Log

Log events for tracking user actions, system errors, and performance metrics:

- Identify key events to log (e.g., user login, product view, order placed).
- Implement logging using Winston or Morgan for Node.js.
- Store logs in a centralized database for analysis.

3.3 Error Handling

- Display user-friendly error messages for client-side errors.
- Log server-side errors and notify the development team.
- Implement retries and fallbacks for critical operations.

3.4 Performance

- Optimize database queries.
- Use caching strategies to reduce load times.
- Conduct load testing to ensure the application can handle high traffic.

3.5 Reusability

- Write modular and reusable code.
- Use component-based architecture in React.

3.6 Application Compatibility

- Ensure the application works on major browsers (Chrome, Firefox, Safari, Edge).
- Ensure responsiveness on different devices (desktops, tablets, smartphones).

3.7 Resource Utilization

- Monitor and optimize CPU and memory usage.
- Scale resources on AWS based on traffic patterns.

3.8 Deployment

- Use AWS EC2 for hosting.
- Use RDS for the database.
- Implement CI/CD pipelines for automated deployments.

Dashboards

4.1 KPIs (Key Performance Indicators)

- Key indicators displaying a summary of the anomaly detection in the society/area.
- Time and workload reduction using the UGV based surveillance.
- To detect mob (illegal) activities and inform police.
- On time alert to nearest hospital on medical emergency (accident).
- Taking adequate evidence of mob.
- Send disaster details to concerned authorities.
- Display of battery life and percentage of UGV.
- Distance traveled by UGV.
- Get the exact location of UGV.

Conclusion

The Daisy Deals e-commerce application will be developed using React for the frontend and Node.js for the backend, ensuring a modern, scalable, and responsive solution for online shopping. The application will provide users with a seamless experience from browsing products to completing their purchases, while the technical architecture ensures robustness and ease of maintenance.