LAB EXP 01 POSTLAB

Q1)

Importance of a well-defined Software Requirement Specification (SRS) in the software development lifecycle and its impact on project success:

A Software Requirement Specification (SRS) is a crucial document that outlines the detailed description of what a software system should do. It acts as a foundation for the entire software development lifecycle and plays a significant role in project success. Here are some key reasons why a well-defined SRS is essential:

Clear communication: The SRS serves as a communication bridge between stakeholders, including clients, developers, testers, and project managers. It ensures that everyone has a common understanding of the software's requirements and functionalities.

Scope definition: A well-written SRS helps in defining the scope of the project accurately. It outlines the boundaries of the software, what features it will include, and what functionalities it will not support, preventing scope creep during development.

Minimizing ambiguities and inconsistencies: A clear and well-structured SRS reduces the likelihood of misunderstandings, ambiguities, and inconsistencies in the software requirements. This, in turn, reduces the risk of costly rework and delays during development.

Basis for design and development: The SRS provides developers with a comprehensive set of requirements that they can use as a reference while designing and coding the software. It acts as a blueprint for development activities.

Validation and verification: The SRS is a basis for validating the final product against the initial requirements. It allows stakeholders to compare the delivered software with the specified requirements to ensure that the project meets the desired goals.

Risk management: A well-defined SRS helps in identifying potential risks and challenges early in the project. This enables the project team to plan risk mitigation strategies and make informed decisions.

Q2)

Without access to a specific SRS document, it is not possible to perform a detailed analysis. However, I can provide some general guidelines on identifying ambiguities and inconsistencies and proposing improvements to enhance clarity and completeness:

Ambiguities: Look for any unclear statements or terms that could have multiple interpretations. Ambiguities can lead to misunderstandings and errors during development. Clarify the language and use precise terminology to avoid ambiguity.

Inconsistencies: Check for conflicting requirements within the document. Ensure that all requirements align with each other and do not create contradictions. Resolve any discrepancies or conflicts to maintain a coherent SRS.

Completeness: Ensure that all necessary requirements are included and nothing critical is

omitted. Verify that the document covers all functional and non-functional requirements, as well as any constraints or assumptions.

Use cases and scenarios: Review the use cases presented in the SRS. Ensure they are comprehensive and cover various scenarios, including both normal and exceptional cases. Add or modify use cases as needed to cover all possible interactions.

Traceability: Establish clear traceability between the requirements and their sources, such as stakeholder inputs or business processes. This ensures that each requirement has a clear rationale and origin.

Testability: Evaluate whether each requirement can be tested effectively. If any requirement lacks testability, consider rewriting it to make it more measurable and verifiable.

3)

Comparison of different requirement elicitation techniques:
Interviews: Interviews involve direct interaction with stakeholders, such as users, clients, and subject matter experts. They provide an opportunity to gather in-depth information and insights from individuals. Interviews are effective for understanding complex and subjective requirements but can be time-consuming and may be biased based on the interviewees.

Surveys: Surveys allow collecting information from a larger audience efficiently. They are useful for getting a broad understanding of user needs and preferences. However, surveys may not delve into complex requirements and might lack the depth provided by interviews.

Use case modeling: Use case modeling is a technique where user interactions and system responses are represented in scenarios. It helps in understanding how the software will be used and what functionalities are required. Use case modeling is effective for capturing functional requirements and their interactions but may not cover all non-functional aspects.

Effectiveness in gathering user needs:
Interviews: Highly effective for capturing detailed and specific requirements from individual stakeholders but can be time and resource-intensive.

Surveys: Efficient for collecting a wide range of opinions and preferences from a larger audience but may lack the depth of understanding provided by interviews.

Use case modeling: Effective for capturing functional requirements and user interactions, providing insights into system behavior.

Lab Experiment 01

Experiment Name: Software Requirement Specification (SRS) as per IEEE Format.

1) Abstract

E-commerce thrift shopping website aims to provide a unique and personalized shopping experience for users. The website provides a platform for buyers and sellers to engage in transactions for used or pre-owned clothing items. The website features a user-friendly interface that enables users to easily navigate through the site and find products of interest. The website incorporates various features such as a search function, filter options, and user profiles that allow buyers and sellers to interact with each other. The website also offers secure payment and shipping options, as well as a rating and review system to ensure the quality of the products and the reliability of the sellers. Overall, the e-commerce thrift shopping website seeks to promote sustainability and affordability while providing a convenient and enjoyable shopping experience.

2) Introduction

2.1) Purpose

The aim of our e-commerce thrift shopping website is to provide an alternative to traditional retail by offering customers affordable and sustainable options.

2.2) Scope

The scope of the project involves designing, developing, and launching an e-commerce website that enables customers to donate their products and purchase items to support charitable organizations. The project team will set up a process for customers to donate their products, including guidelines for acceptable items, shipping instructions, and quality control procedures. The website will also include information about the charitable organizations that will receive the donations. Team will launch the website and deploy it to a production environment, making it available to customers.

2.3) Definitions, Acronyms, Abbreviations

Not applicable.

2.4) References

[1] March 2014, Computer Science & Engineering School Of Engineering Cochin University Of Science & Technology Kochi-682022 (Shibin Chittil, Nidheesh Chittil , Rishikese M R) Already existing e-commerce thrift shopping website:

1. https://kiabza.com/

2. https://www.amalfiindia.com/

2.5) Developer’s Responsibilities

• To create a user-friendly and intuitive website that allows customers to easily browse, search, and purchase second-hand or used items.

• To promote sustainable and affordable shopping by offering a platform for customers to buy and sell second-hand goods.

• To differentiate the website from traditional retailers and other e-commerce websites by offering unique value propositions such as sustainability, affordability, and community.

3) General Description

3.1) Product Functions Overview

FRONTEND:

REACT: With its component-based architecture and powerful state management, React makes it easy to create a dynamic and responsive user interface that can handle complex data and user interactions. First, we designed the components in a way that allows for easy reuse and modularity, so that we could efficiently build out different pages and features of for site. Second, we made sure that the components are optimized for performance, since the website handles large amounts of data and user traffic. Finally, we have to integrate our React frontend with a backend system that can handle payments, orders, and other e-commerce-specific functionality.

BACKEND:

1) NODE.JS: Node.js provides a lightweight, event-driven architecture that makes it well-suited for building scalable, high-performance web applications. With Node.js, we built a RESTful API to handle ecommerce specific functionality like processing payments, managing orders, and updating inventory. We used various packages and libraries of Node.js ncluding popular frameworks like Express.js, that make it easy to build robust and secure backend systems.

2) EXPRESS: Express is a popular web framework for Node.js that can be used to build the backend of an ecommerce shopping website. Express provides a robust set of features and middleware that can make it easier to handle common web development tasks, such as routing, error handling, and parsing request bodies.

3) REST API: REST (Representational State Transfer) is an architectural style for building web services that uses HTTP requests to GET, POST, PUT, and DELETE data. By using a RESTful API, we created a standard set of endpoints that will allow frontend client to communicate with your backend server to retrieve and manipulate data related to ecommerce functionality such as products, orders, carts, and user accounts.

4) POSTMAN: Postman is a powerful tool that can help streamline the development and testing of an e-commerce shopping website, allowing us to quickly and easily test your API endpoints and ensure that our backend server is functioning correctly. This is especially useful when working with a RESTful API, as you can quickly test GET, POST, PUT, and DELETE requests to retrieve and manipulate data related to ecommerce functionality like products, orders, and user accounts.

5) MONGODB: MongoDB is a popular NoSQL database that can be used to store and manage data for an ecommerce shopping website. As a document-oriented database, MongoDB stores data in JSON-like documents, making it a flexible and scalable choice for handling the complex and dynamic data structures commonly found in ecommerce applications.

6) STRIPE: Stripe is a popular payment processing platform that can be used to handle online transactions for an ecommerce shopping website. Stripe provides a simple and secure way to accept payments from customers, allowing you to easily integrate payment functionality into your website and start selling products

3.2) User Characteristics

The main users of this system will be the costumers.

3.3) General Constraints

The system should run on servers and hosting environments.

3.4) General Assumptions and Dependencies

Not applicable.

4) Specific Requirements

4.1) Inputs and Outputs

Methodology:

1. Planned the structure of the ecommerce shopping website by defining the target audience, product offerings, and business goals.

2. Build the frontend and designed different pages and components like register, login, homepage, navbar, newsletter, footer, products, productlist, and shopping cart of the website using React. Used two libraries:

1) Styled Modules is a CSS-in-JS that allowed to write CSS styles as individual modules, where each module contains scoped CSS that only applies to the component it's associated with. This approach helped avoid style clashes and makes it easier to maintain and refactor the codebase.
2) Material-UI component library that provides pre-built, customizable React components that implement the Material Design system is used for the various icons used, ex. the cart icon in the navbar.

3. Set up a MongoDB database and configured it to store the data for the website.

4. Created a Node.js server using Express and configure it to interact with the MongoDB database.

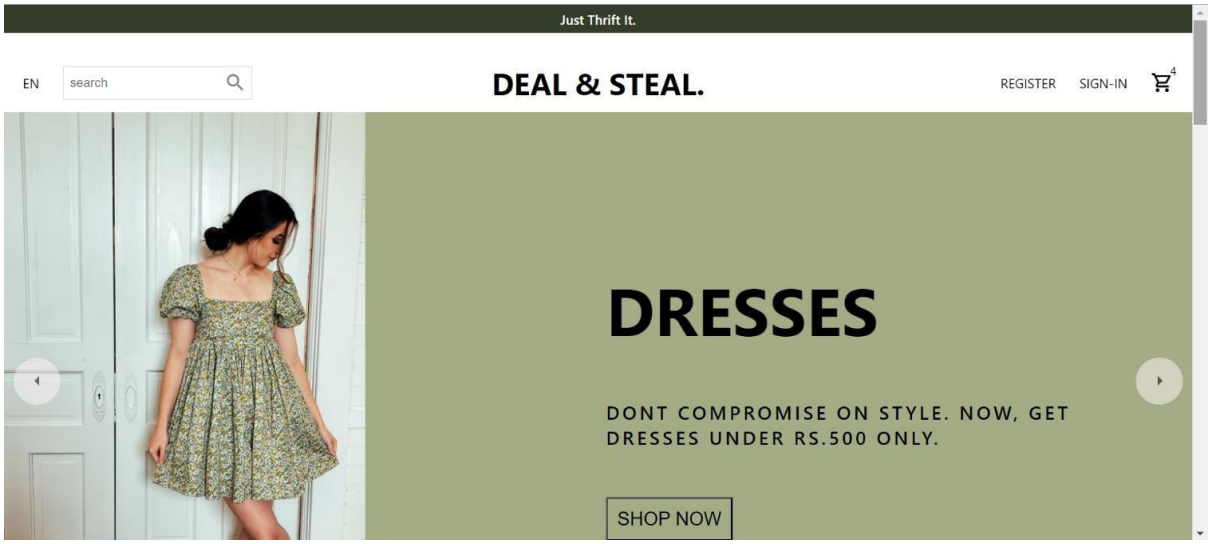
5. Created a REST API using Node.js and Express to allow communication between the client-side and server-side of the application.

6. Used Postman to test the REST API and ensure that it is working correctly.

7. Integrated Stripe to your server-side to allow users to make secure online payments.

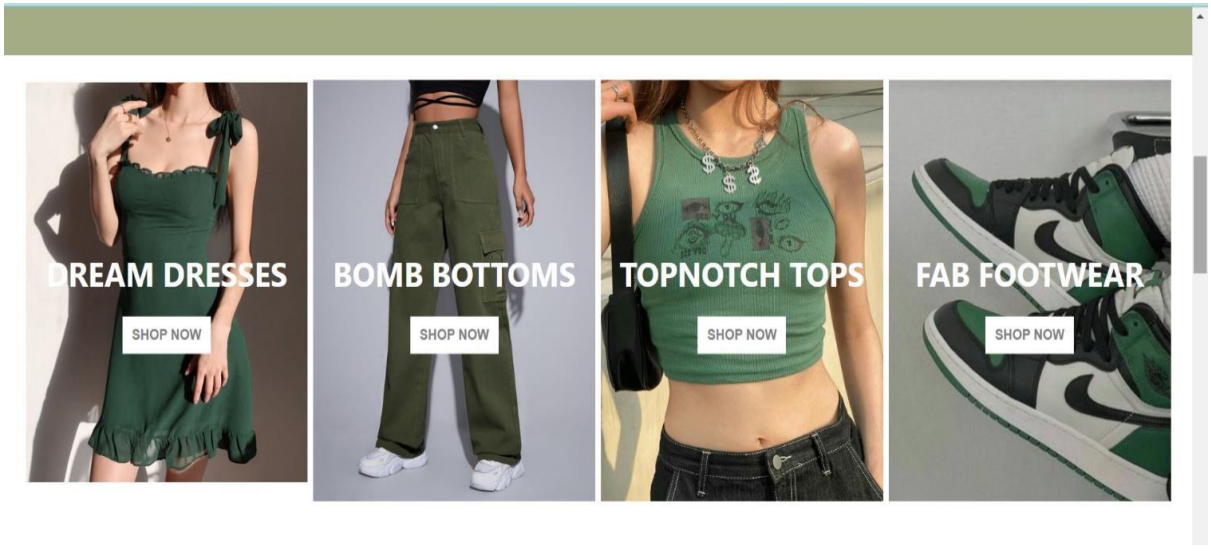
8. Tested and debugged the website to ensure that it is functioning correctly

Implementation :



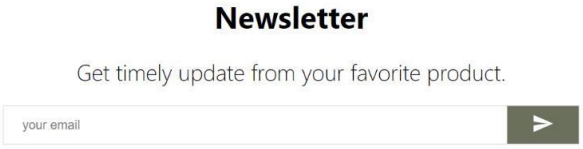
4.31 Nav and Slider on Home Page

The homepage of our website is the main or initial page where the users land on when they visit the website. It serves as a window into the site's content, purpose, and personality by presenting a highlevel overview of the website's features and services.



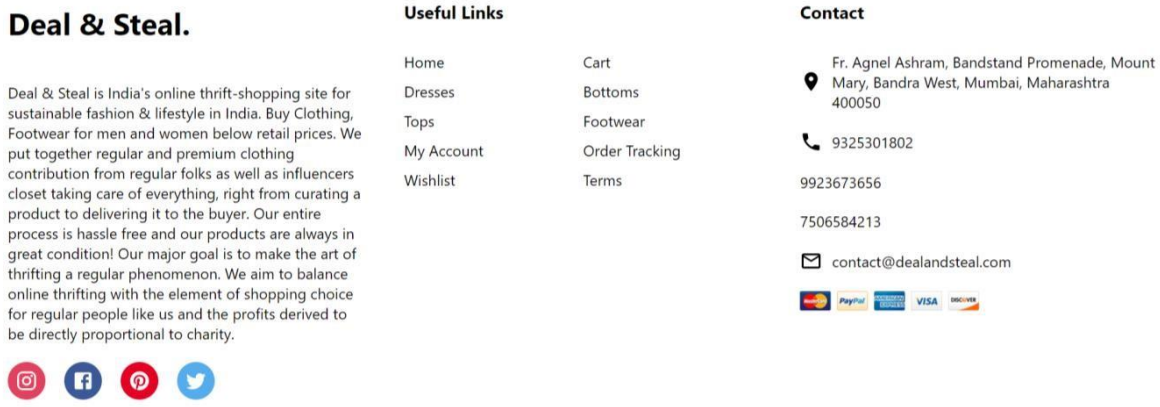
4.32 Categories on home page

The category section on our home page shows varieties of clothing available on our website. Categories such as dresses, bottoms, tops, and footwear.



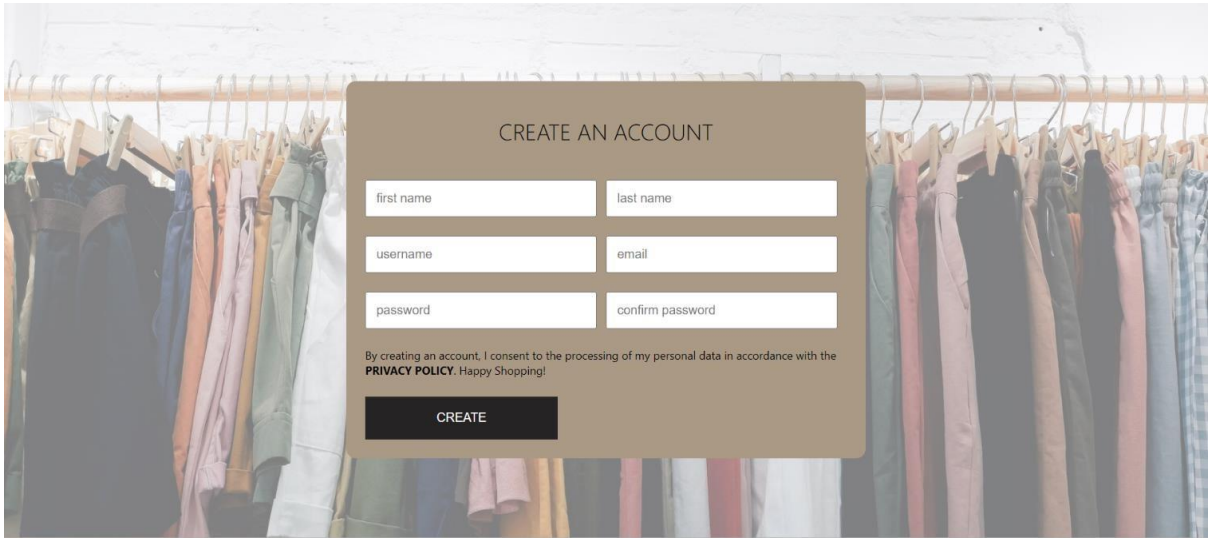
4.33 Newsletter

The newsletter is to keep subscribers informed about updates, news, special offers, and other relevant information about the website.



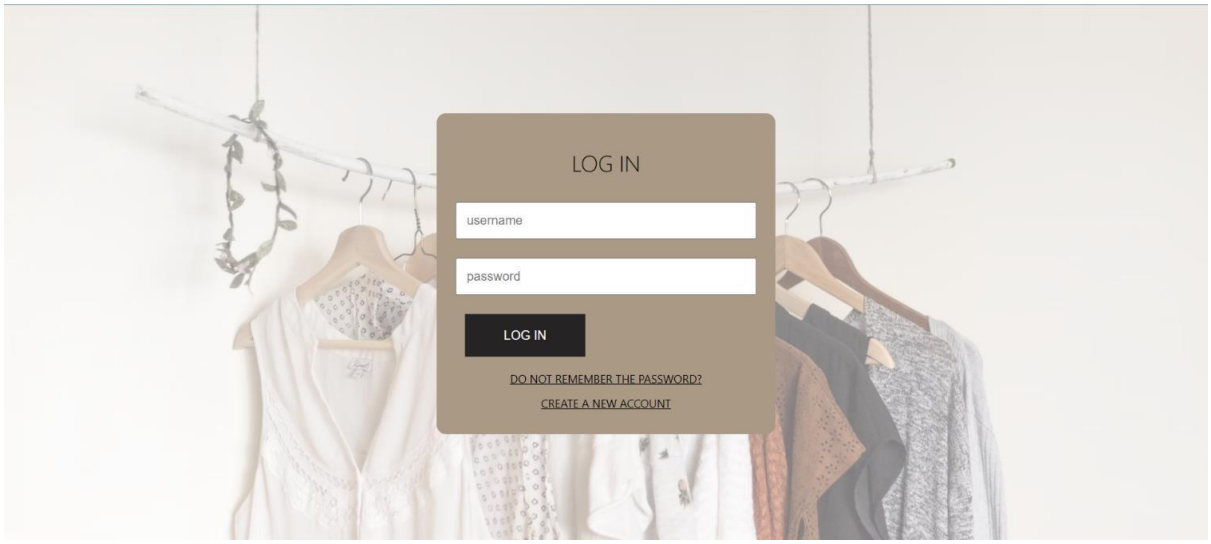
4.34 About and contact

The About and Contact page is located at the bottom of the page and contains important links and information about the website and its features.



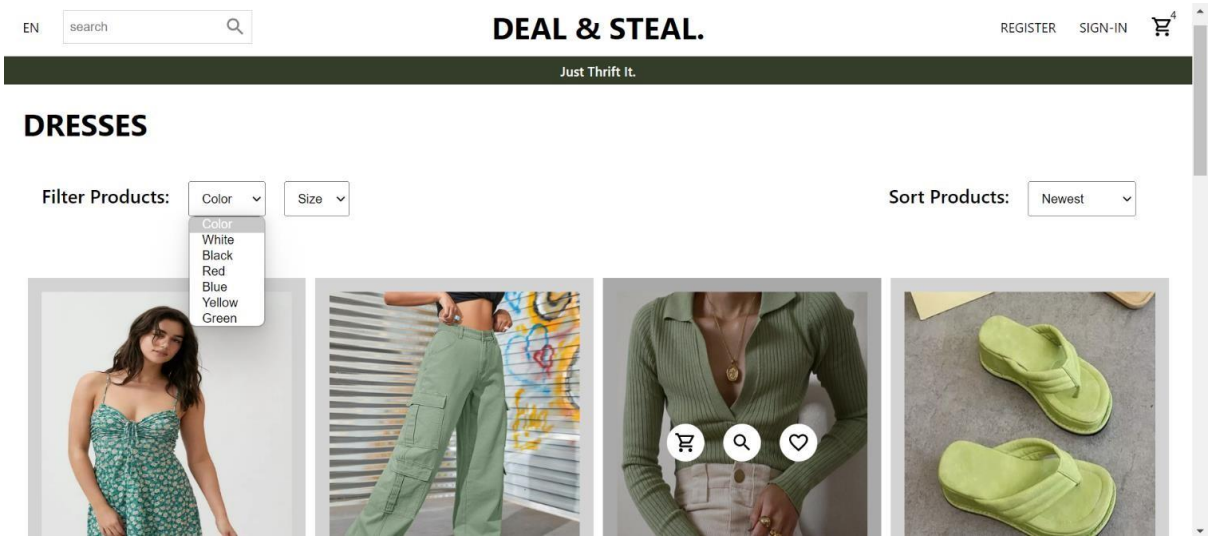
4.35 Register page

The register page of the website is where new users can create an account.



4.36 Login page

The login page of the website is the designated location for registered users to enter their credentials and access their accounts.

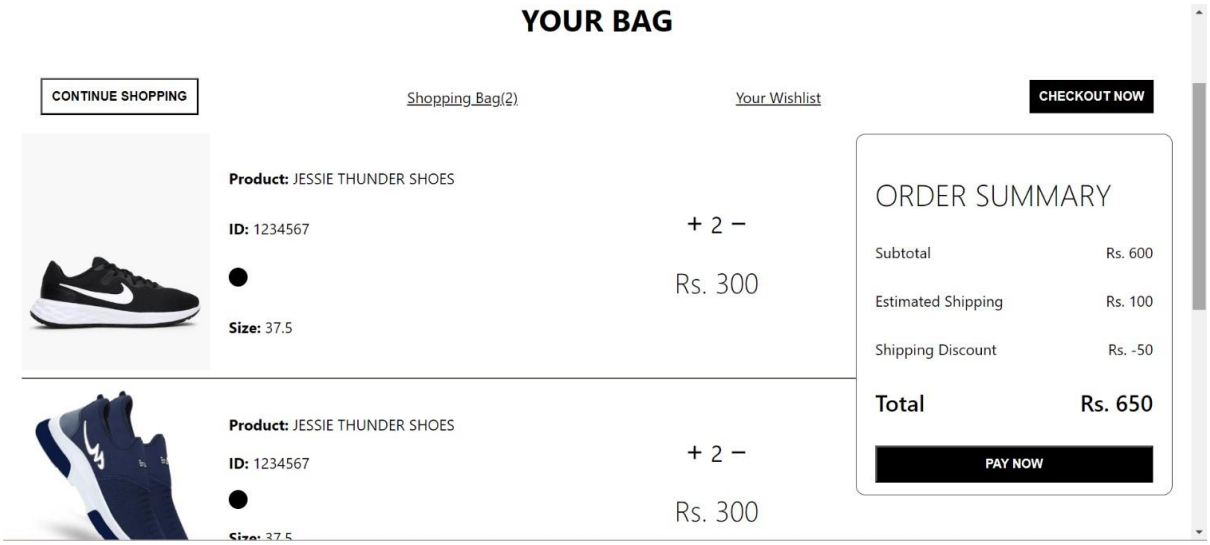


4.37 Product list

This page displays a list of products available for purchase. It can provide customers with a quick and easy way to discover new items and filter them by color, size and sort products.

4.38 Product

The product page opens the product selected by the customer and tells its description, price, availability of color, sizes and quantity.



4.39 Cart

On the cart page, customers can see the item's information, including the name, price, id, color, quantity, and size along with the order summary .

4.2) Functional Requirements

Hardware requirements

There was no hardware used.

Software requirements

1) React

2) Node.js

3) NPM

4) MongoDB

5) Postman

6) Stripe

4.3) External Interface Requirements

Not applicable.

4.4 Performance Constraints

Not applicable.

4.5 Design Constraints

Not applicable

5) Summary & Conclusion

The E-commerce thrift shopping website project offers a viable solution to the problem of unsustainable and expensive shopping practices. Our mission is to inspire people and make them aware of how important it is to shop consciously. In this project, we have used React for Frontend part and Node .js, Express, Rest Api, Postman, MongoDB, and Stripe for the backend. Overall, the e-commerce thrift shopping website project can make a positive impact on the environment and the economy, while meeting the needs of customers and sellers alike.