



CAUVERY DEGREE COLLEGE, GONIKOPPAL

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATION



PROJECT REPORT ON

“FABRICLORE”

Submitted in partial fulfilment of the requirement of the award of
the degree of

BACHELOR OF COMPUTER APPLICATION

Submitted By:

SHISHIK DEVAIAH SP

Reg no. **U05CU21S0035**

Under the guidance of

Internal Guide:

Mr. Pemmaiah U.T

Department of Computer Science
Cauvery College, Gonikoppal.

External Guide:

Mrs. Sunitha

SP Technologies
Bangalore.

CAUVERY COLLEGE, GONIKOPPAL

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATION



This is to certify that Mr Shishik Devaiah SP With register no U05CU21S0035 of III BCA VI semester has satisfactorily completed the project work entitled "LOGISTICS MANAGEMENT SYSTEM" in partial fulfilment for the award of Bachelor of Computer Application of Mangalore University during the year 2023-2024.

.....
INTERNAL GUIDE

.....
Dr. M.B. Kaveriappa
PRINCIPAL

.....
HOD

Submitted to the university Examination on at Cauvery College, Gonikoppal, Kodagu.

.....
INTERNAL

.....
EXTERNAL

Refno - SPTCH04/17-24

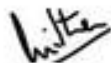
Date: 19-06-2024

Certificate of Internship Completion

This is to certify that **Mr. SHISHIK DEVAIAH SP.** (Reg_No: **U05CU21S0035**),
III BCA of Cauvery College, Gonikoppal, Karnataka, India, has successfully completed an
internship as a Junior Software Engineer for the role of Front End Developer at CodeNish
Technology Pvt. Ltd., Ponnampet, India, from **1/02/2024 to 15/06/2024**. During this internship,
he exhibited exemplary dedication, enthusiasm, and professionalism in software development.

Throughout the internship period, he demonstrated a strong aptitude for
learning and problem-solving, contributing effectively to the team's objectives and enhancing
his skills in coding and testing.

This certificate acknowledges **Mr. SHISHIK DEVAIAH SP** 's successful completion of his
internship and recognises his commitment to personal and professional growth in the field of
software engineering.



Sunitha.V
Director IT Services



ABSTRACT

The "Online Fabric Store" project aims to create a comprehensive e-commerce platform dedicated to serving fabric enthusiasts and creative professionals. This initiative seeks to develop a user-centric website where customers can effortlessly browse, select, and purchase a wide array of fabrics online. The primary goal is to establish a virtual hub that caters to diverse needs, offering an extensive collection of fabrics in various materials, colors, designs, and textures. By leveraging this platform, customers ranging from designers to hobbyists can find the ideal fabric to suit their specific crafting and design requirements. This project underscores a commitment to enhancing accessibility, convenience, and choice in the realm of online fabric shopping, thereby fostering creativity and innovation within the global textile community.

This abstract delves into Fabriclore's innovative approach to e-commerce, leveraging React.js to enhance user experience and functionality on its online platform. It explores how React.js, a popular JavaScript library for building user interfaces, has enabled Fabriclore to create a seamless and responsive website. The abstract highlights key features such as dynamic product showcases, intuitive navigation, and personalized user interactions that React.js facilitates. It also discusses the benefits of using React.js in improving site performance, scalability, and maintenance efficiency. Ultimately, the abstract underscores Fabriclore's commitment to merging technology with textile artistry, providing a modern and engaging online shopping experience for fabric enthusiasts worldwide

ACKNOWLEDGEMENT

The satisfaction after completion of any task would be incomplete without mentioning the people who were constantly with us, people who made it possible and guided me in the most rightful path, encouraged me who made my effort come true. Here by I am proud to express my gratitude to all of them.

In the name of God, First and foremost I am thankful to our institution CAUVERY DEGREE COLLEGE providing facility to complete my graduation. I considered it is my privilege to express my gratitude and respect to all those who guided, inspired and helped me in the completion of this project. I owe a debt of gratitude to all of them who were so generous with their time and expertise. I also wish to thank them for their warm hospitality.

I would like to express my gratitude and thanks to our principal **Prof. M.B. KAVERIAPPA**, HOD of BCA department **Mr PEMMAIAH U.T** and external guide **Mrs SUNITHA**.

I also extend my sincere thanks and gratitude to my parents and my friends for the enthusiasm and infused to me during my project work and helping me in completing this course.

My sincere thanks and regards to one and all those who have helped me either directly or indirectly in completing the project work and the course.

DECLARATION

I hereby declare that this project work titled **“FABRICLORE(online fabric-store)”** submitted to the **“Cauvery Degree College Gonikoppal”**. Affiliated to **“Mangalore University”**, as partial fulfilment for the award of **“BACHELORS OF COMPUTER APPLICATION”**. We have done this project in a period of three months. I declare that this project is entirely based on the information provided by the company and the result is of my own efforts. We further declare that this project is based on the original study undertaken by SHISHIK DEVAIAH SP and has not formed a basic for the award of any degree/diploma of any other university/institution.

SHISHIK DEVAIAH S.P [Reg No:U05CU21S0035]

Index

| SL. No | Tables of content | Page No. |
|-----------|--|-------------|
| 1. | 1. Introduction | 1-2 |
| | 1.1 Title of the project | 1 |
| | 1.2 Purpose | 1 |
| | 1.3 Project Overview | 1 |
| | 1.4 Scope | 1 |
| | 1.5 Requirement / Functionalities of project | 1 |
| | 1.6 Objectives | 2 |
| 2. | 2. System Analysis | 3-23 |
| | 2.1 Identification of need | 3 |
| | 2.2 Preliminary Investigation | 4 |
| | 2.3 Feasibility Study | 5 |
| | 2.3.1 Technical Feasibility | 5 |
| | 2.3.2 Economical Feasibility | 5 |
| | 2.3.3 Operational Feasibility | 5 |
| | 2.4 Project Planning | 6 |
| | 2.5 Project Scheduling | 6 |
| | 2.6 Activity Chart | 7 |
| | 2.7 System Diagrams | 7 |

| | | |
|-----------|--|--------------|
| | 2.7.1 Context Flow Diagram | 7 |
| | 2.7.2 Data Flow Diagram | 8 |
| | 2.7.3 Entity Relationship Diagram | 13 |
| | 2.7.4 Use case | 14 |
| | 2.8 Software Requirement Specification | 16 |
| | 2.8.1 Introduction | 16 |
| | 2.8.2 Scope | 16 |
| | 2.8.3 Functional Requirements | 17 |
| | 2.8.4 Non-functional Requirements | 18 |
| | 2.8.5 Module list | 19 |
| | 2.9 Technical Specification | 20 |
| | 2.9 Language used | 20 |
| | 2.10 Hardware/Software Requirement specification | 21 |
| | 2.10.1 Developing desktop requirements | 21 |
| | 2.10.1.1 Hardware/Software used for client | 22 |
| | 2.10.1.2 Hardware/Software used for server | 23 |
| | 2.10.2 Constraints | 23 |
| 3. | 3. System Design | 24-30 |
| | 3.1 Introduction | 24 |
| | 3.1.1 Primary design phase | 24 |

| | | |
|-----------|-----------------------------------|--------------|
| | 3.1.2 Secondary design phase | 24 |
| | 3.2 Database design | 24 |
| | 3.3 Data dictionary | 25 |
| 4. | 4. Software Testing | 31-33 |
| | 4.1 Introduction | 31 |
| | 4.2 Test Plan | 31 |
| | 4.3 Test case design | 31 |
| | 4.3.1 Methods of test case design | 31 |
| | 4.4 Testing Strategy | 32 |
| | 4.5 Validation criteria | 32 |
| | 4.6 Importance of testing | 33 |
| 5. | 5. Coding | 34-67 |
| | 5.1 Introduction | 34 |
| | 5.2 Code List | 34 |
| 6. | 6. User Interface | 68-72 |
| | 6.1 Index page | 68 |
| | 6.2 Admin Login | 68 |
| | 6.3 Add Product | 69 |
| | 6.4 Add Staff | 69 |
| | 6.5 Customer Login | 70 |

| | | |
|-----------|------------------------|--------------|
| | 6.6 Customer Register | 70 |
| | 6.7 Add to cart | 71 |
| | 6.8 Cart | 71 |
| | 6.9 Order and Payment | 72 |
| | 6.10 Manage order | 72 |
| | 6.11 View Feedback | 72 |
| 7. | 7. Conclusion | 73-74 |
| | 7.1 Limitations | 73 |
| | 7.2 Future Scope | 73 |
| | 7.3 Conclusion | 73 |
| | 7.4 References | 74 |
| | 7.4.1 Book referred | 74 |
| | 7.4.2 Website referred | 74 |

List of Tables

| SL No. | Table No | Table Name | Page No. |
|--------|----------|--------------------|----------|
| 1 | 2.5 | Project Scheduling | 6 |
| 2 | 2.6 | Activity chart | 7 |
| 3 | 2.7.2 | Data Flow Diagram | 9 |
| 4 | 2.7.3 | ER Diagram | 13 |
| 5 | 2.7.4 | Use Case | 14 |
| 6 | 3.3.1 | User Table | 24 |
| 7 | 3.3.2 | Cart Table | 24 |
| 8 | 3.3.3 | Category Table | 25 |
| 9 | 3.3.4 | Feedback Table | 25 |
| 10 | 3.3.5 | Orders Table | 26 |
| 11 | 3.3.6 | Staff Table | 26 |
| 12 | 3.3.7 | Product Table | 27 |
| 13 | 3.3.8 | State Table | 28 |

List of Figures

| SL No. | Figure No. | Figure Title | Page No. |
|--------|------------|---------------------------|----------|
| 1 | 2.7.1 | Context Flow | 7 |
| 2 | 2.7.2.1 | Level 1 Data Flow Diagram | 9 |
| 3 | 2.7.2.1 | Level 2 Data Flow Diagram | 11 |
| 4 | 2.7.3 | ER Diagram | 13 |
| 5 | 2.7.4 | Use Case Diagram | 14 |

1. Introduction

1.1 Title of the project:

Online Fabric Store

1.2 Purpose

The objective behind initiating an online fabric store project revolves around developing an all-encompassing and user-centric e-commerce platform. This platform is tailored to enable customers to seamlessly explore, choose, and buy fabrics online. The overarching goal is to establish a virtual hub catering to fabric aficionados, designers, and individuals seeking top-notch fabrics for their creative ventures. The online fabric store project aspires to curate a diverse assortment of fabrics, spanning multiple materials, hues, designs, and textures. This extensive variety empowers customers to discover the perfect fabric that aligns with their precise crafting or design requisites.

1.3 Project Overview:

The fundamental idea behind the application is to provide customers with the opportunity to engage in virtual shopping through the internet. This allows them to seamlessly purchase items of their preference from the store's inventory. Product-related data is stored within an RDBMS on the server side. The server handles customer interactions, and once items are selected, they are dispatched to the address provided by the customer. The application has been structured into two primary modules: one catering to customers looking to make purchases, and the other addressing the needs of those customers.

1.4 Scope:

The endeavour should encompass a comprehensive compilation of product details along with their respective descriptions. This repository encompasses comprehensive information regarding both supplementary and complementary items linked to specific products. The platform will facilitate the processing and assessment of customer payments. Furthermore, it will encompass a comprehensive elucidation of all the functionalities attributed to a particular product. Ensuring meticulous payment record maintenance is also a crucial component.

1.5 Requirements / Functionalities of Online Shopping Store

- All the Items to the store can only be added through Administrator.
- Its optional that a customer may or may not register with our website / store.
- A customer can navigate between various products / fabrics and could also look for their detailed description. □ After finalizing, fabric cloths can be purchased through various payment gateways.

1.6 Objectives:

- With “online fabric store” customer can browse through the shop and purchase them online without having to visit the shop physically.
- The administration module will enable a system administrator to approve and reject order requests for customers.
- The main objective of the project is, establish a network among the people residing all over the world through which all the information related to products & any new updates can be easily accessed and it can be shared among the people.
- This system provides admin module to manage different domains details and customers to check out the details of different products.

2. System Analysis

System evaluation entails the compilation of empirical data to grasp the intricacies of the involved processes, pinpoint existing issues, and propose viable recommendations to enhance system performance.

The process of system analysis plays a pivotal role in the advancement and refinement of an internet-based fabric store. It entails a comprehensive assessment of the existing system, recognizing its strengths and vulnerabilities, and presenting remedies to amplify its efficacy, productivity, and user engagement. Presented below are the essential constituents of system analysis tailored to an online fabric store:

- Requirements Gathering
- Functional Analysis
- User Experience (UX) Evaluation
- System Integration
- Performance Analysis

- Security and Privacy
- Reporting and Analytics
- Scalability and Future Growth
- Documentation and Training

2.1. Identification of Need

1. **Extensive Array of Choices:** Customer preferences often span a broad spectrum of fabric options. Ensuring a diverse assortment of fabrics encompassing an array of colours, designs, textures, and materials caters to an array of customer tastes and project requisites.
2. **Variable Quantity Alternatives:** Offering distinct quantity choices, such as fat quarters, half yards, and full yards, provides customers the flexibility to procure the exact fabric volume required for their unique undertakings.
3. **Shipping Precision and Swiftness:** Dependable and effective shipping strategies prove indispensable. Providing choices for expedited shipping, real-time package tracking, and transparent disclosure of shipping costs and delivery schedules is highly valued by customers.
4. **Customer Assistance:** Furnishing responsive customer support via various communication avenues (e.g., email, live chat, phone) effectively addresses customer queries, extends aid, and rapidly resolves any concerns or issues.
5. **User-Friendly Navigation:** An intuitively structured website with seamless navigation augments the user journey. Customers should be able to seamlessly peruse diverse fabric categories, employ filters (e.g., fabric type, colour, price), and promptly locate their desired selections. By tending to these requisites, an online fabric store can cultivate a favourable customer experience, nurture patron allegiance, and cultivate a robust reputation within the fabric industry.

2.2. Preliminary Investigation

Following are the results of preliminary investigation of the requirements of the project. It is required for each simulation package to mimic the function, validations including the error message of the respective function. This may contain multiple screens to be navigated

through, may contain multiple tabs required for data entry. Fields may be mapped to different categories such as

- Drop down fields
- Date entries
- Text fields
- Popup boxes ▪ Check boxes and so on.

To ensure optimal learning outcomes, a distinct business scenario is crafted for every function. An audio component delivers the business case to the learner, who is then tasked with inputting pertinent data aligned with the contextual business landscape. Post input, the data undergoes validation to ascertain its accuracy. Should the user input an erroneous value that deviates from the stipulated business context, an error message surfaces. Once the user enters the appropriate value, the error message is concealed. Following successful data input on each screen, provided the validations are met, the user progresses to the subsequent stage. Upon concluding the data input process, a success message acknowledges the user's accomplishment. In its entirety, the system study phase encompasses the subsequent stages: □ Identification of issues and project initiation.

- Thorough background analysis
- Deductions or insights derived (system proposal)

2.3. Feasibility Study

At the initiation of a project, a critical aspect to ascertain involves evaluating the project or product's feasibility. Feasibility pertains to the degree to which pertinent data and information are immediately accessible or can be procured using existing resources, including staff, expertise, time, and equipment. It essentially serves as an assessment of the practicality and advantages tied to the creation of a software system for the organization. This evaluation is a recurring process that spans the entirety of the lifecycle.

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

- **Technical Feasibility:**

The realm of technical feasibility centres on the technological assets at an organization's disposal. This examination assists in ascertaining if the technical resources possess the required capacity and whether the technical team is equipped to translate concepts into a functional system. Furthermore, technical feasibility entails an assessment of the hardware, software, and other technical prerequisites tied to the proposed system. It encompasses not only the presence of adequate hardware and software but also their aptitude for effective implementation.

- **Economic Feasibility**

This evaluation often encompasses a comprehensive cost-benefit analysis of the project, aiding organizations in appraising the project's feasibility, expenses, and potential gains prior to committing financial resources. It concurrently functions as an autonomous appraisal of the project and bolsters the project's credibility. This endeavour enables decision-makers to ascertain the favourable economic advantages that the proposed project offers to the organization.

- **Operational Feasibility**

This evaluation entails conducting an in-depth study aimed at scrutinizing and ascertaining the extent to which the organization's requirements can be fulfilled through project completion. Operational feasibility assessments additionally explore the manner in which a project plan can effectively fulfil the requisites identified during the requirements analysis phase within the system development process.

2.4. Project Planning

In addressing real-world challenges within the industry landscape, software engineers, or even teams of engineers, must embrace a development approach encompassing comprehensive strategies, overarching processes, methodologies, tools, and distinct stages. Termed as a software engineering paradigm or process model, this strategy is meticulously chosen in tandem with the project's nature, its application, the designated methodologies and tools, as well as the requisite controls and deliverables.

Software Requirement Analysis: This phase involves the meticulous review of project requirements, which were furnished by the project manager in hard copy format. Upon a thorough assessment of the requirements, the design phase commences.

- **Design:** Following requirement determination, the focus shifts to the design phase. During this stage, we referred to numerous online sources to enhance our understanding and deliver optimal outcomes.
- **Code Generation:** The design specifications must be translated into executable code. Thus, after successfully concluding the design phase, we initiated the coding process. The functional approach was adopted for coding.
- **Testing:** With coding completed, we transitioned to the testing phase, where we subjected our system to rigorous assessments across various scenarios. This ensured that our system reliably produced desired outcomes for specified inputs.

2.7 System Diagrams

2.7.1 Context Flow Diagram

A context-free diagram serves as a high-level representation, encompassing a single process node that encapsulates the system's overarching function and its connections to external entities. Within the context flow diagram, the entire system is conceptualized as a singular process, illuminating the identification and visualization of its inputs, outputs, sources, and destinations.



Context Flow Diagram (CFD)

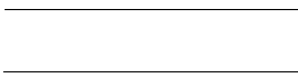

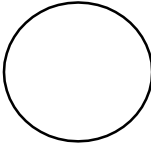

- **Circle** is to represent the system in terms of a single process. There will never be more than single process in any of the context diagram.
- **Arrow** to represent data flow.
- **Rectangle** in CFD is to represent any external entities affecting the system. Also there can be numerous external entities.

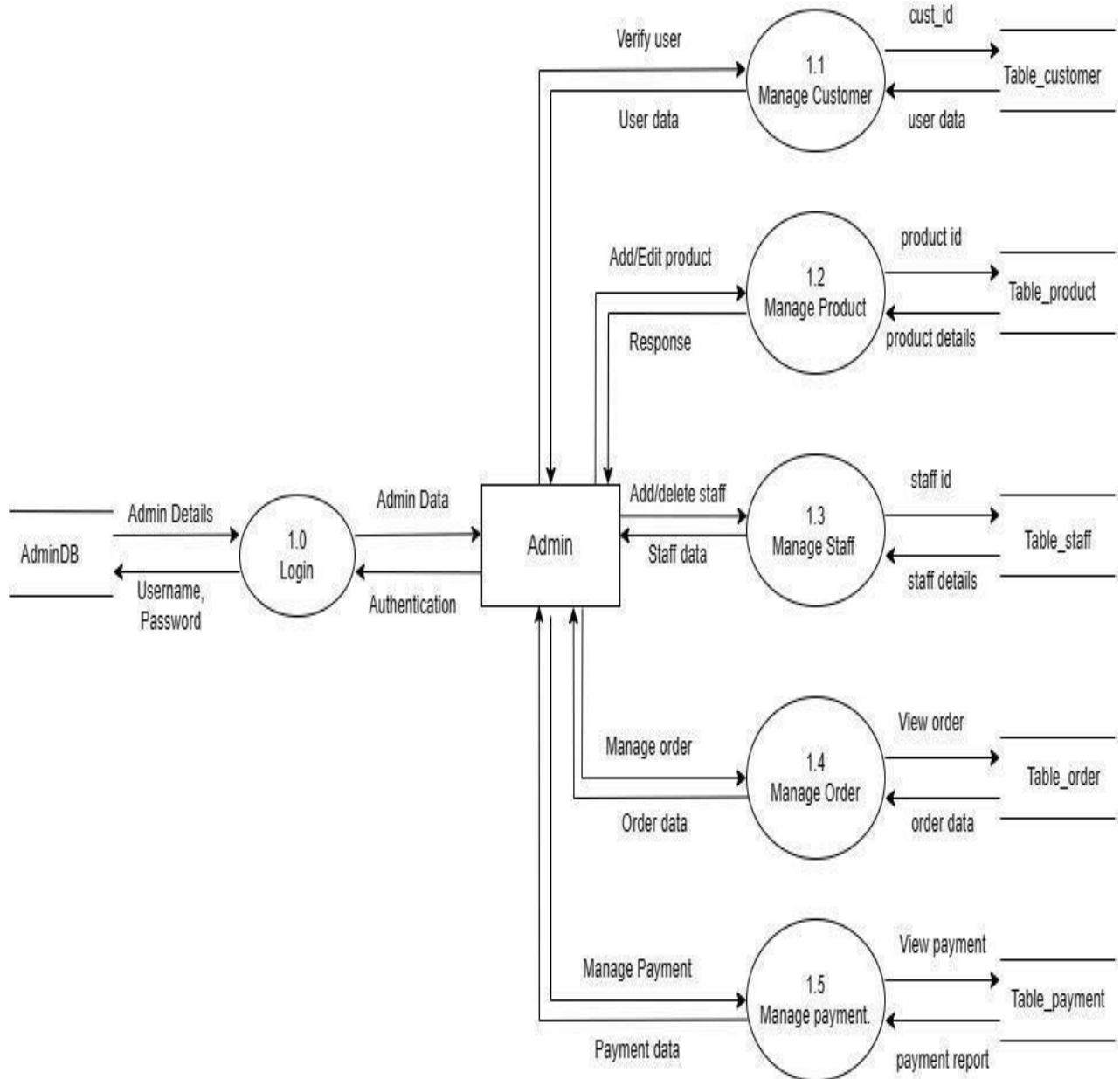
2.7.2 Data Flow Diagram

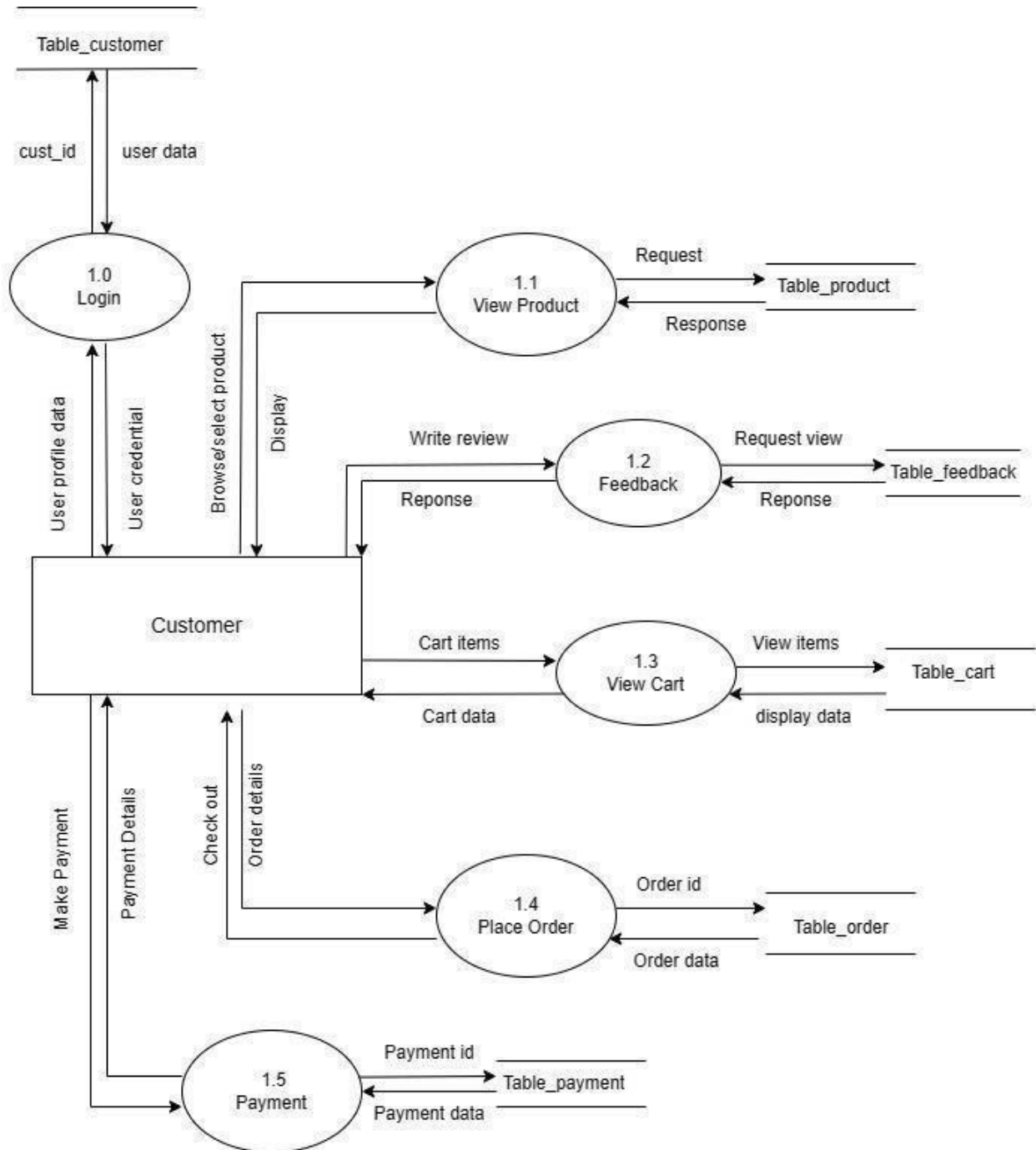
A Data Flow Diagram serves as a graphical representation illustrating the trajectory of data values as they traverse from their origins within entities through processes that modify them, to their eventual destinations within other entities. The core objective of this diagram lies in clarifying system requisites and pinpointing pivotal transformations that will eventually be translated into programs during system design. Hence, it serves as the inaugural stage of the design phase, orchestrating a functional decomposition of requirement specifications into their most granular level of detail. A DFD comprises a sequence of nodes connected by lines. These nodes symbolize data transformations, while the connecting lines denote the movement of data within the system. Within a DFD, one encounters processes

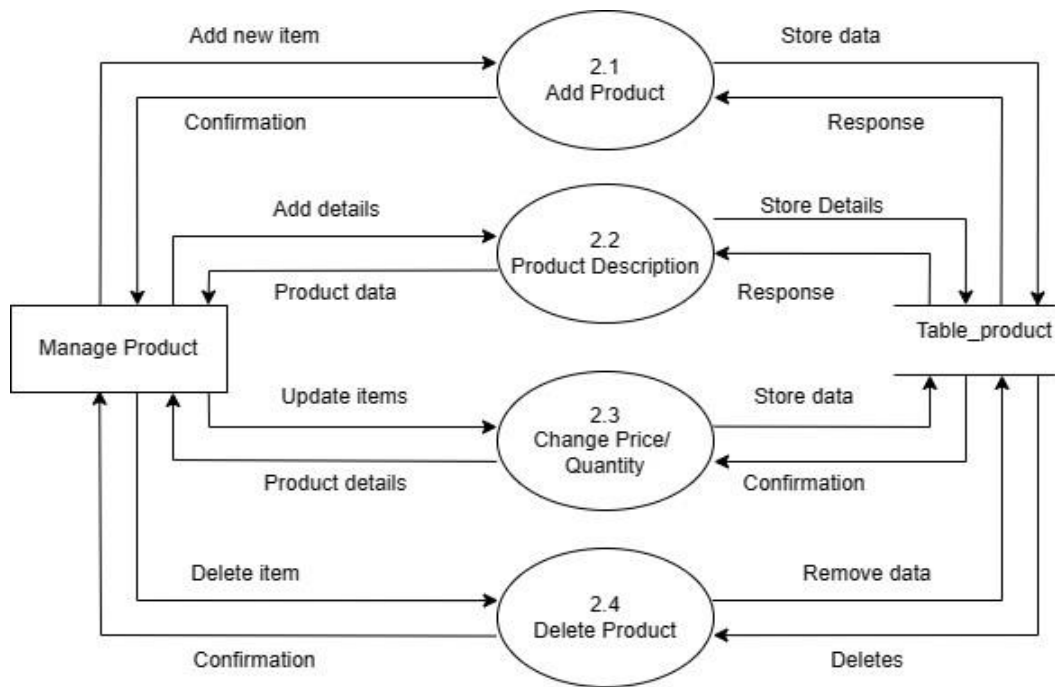
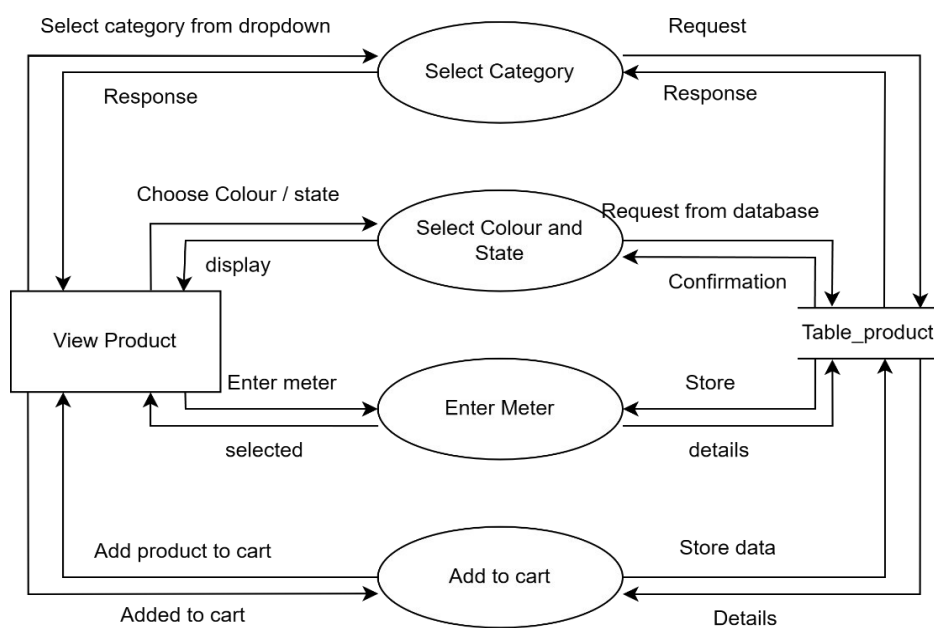
responsible for data manipulation, data flows governing data movement, actor entities

generating and consuming data, and data store entities passively housing data

| Name | Notations | Description |
|-------------------|---|--|
| Data Store |  | The data store as in figure Represents a logic file. A logic file can Represents either a store symbol which represents either a data structure or physical file on the disk |
| Data Flow |  | The data flow is used to describe the moment of information from one part of system to another part. Flow represents data in motion .it is a pipe line through which information flows. |
| Process |  | The circle or Bubble represents a process that transforms incoming data to outgoing data. Process shows a part of the system that transforms input to output |
| External Entities |  | A square defines a source or destinationof system data. External entities Represents any entity that supplies or to receive information from the system but is not a part of the system. |

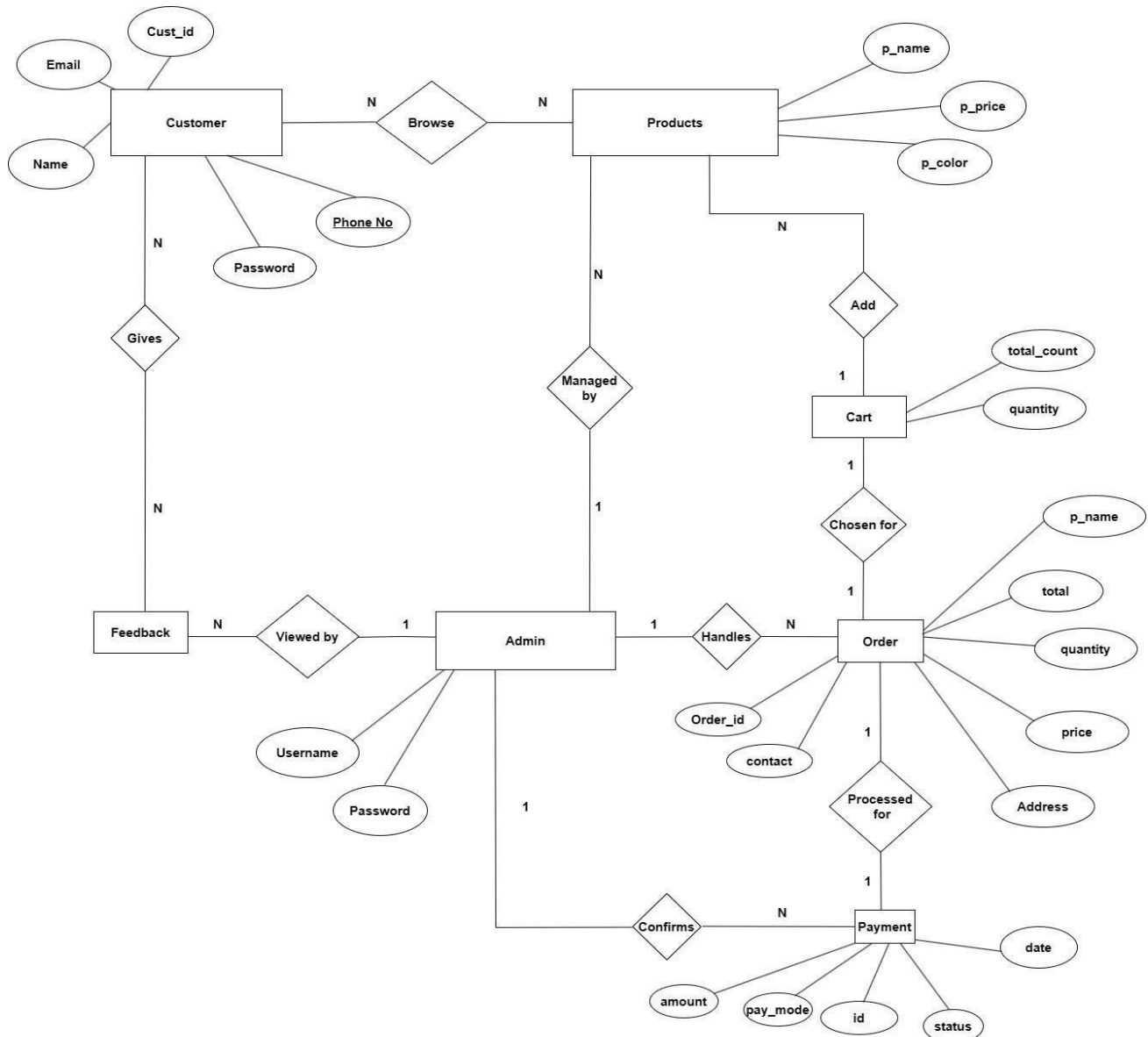
Level 1 DFD Admin

Level 2 DFD: Customer

Level 2 Admin DFD [Manage Product]**Level 2 Customer DFD [View Product]**

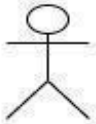
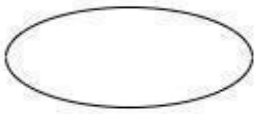
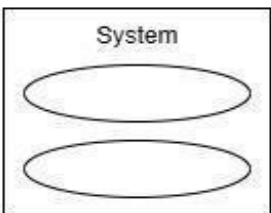
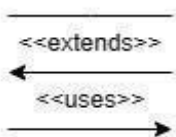
2.7.3 Entity Relationship Diagram

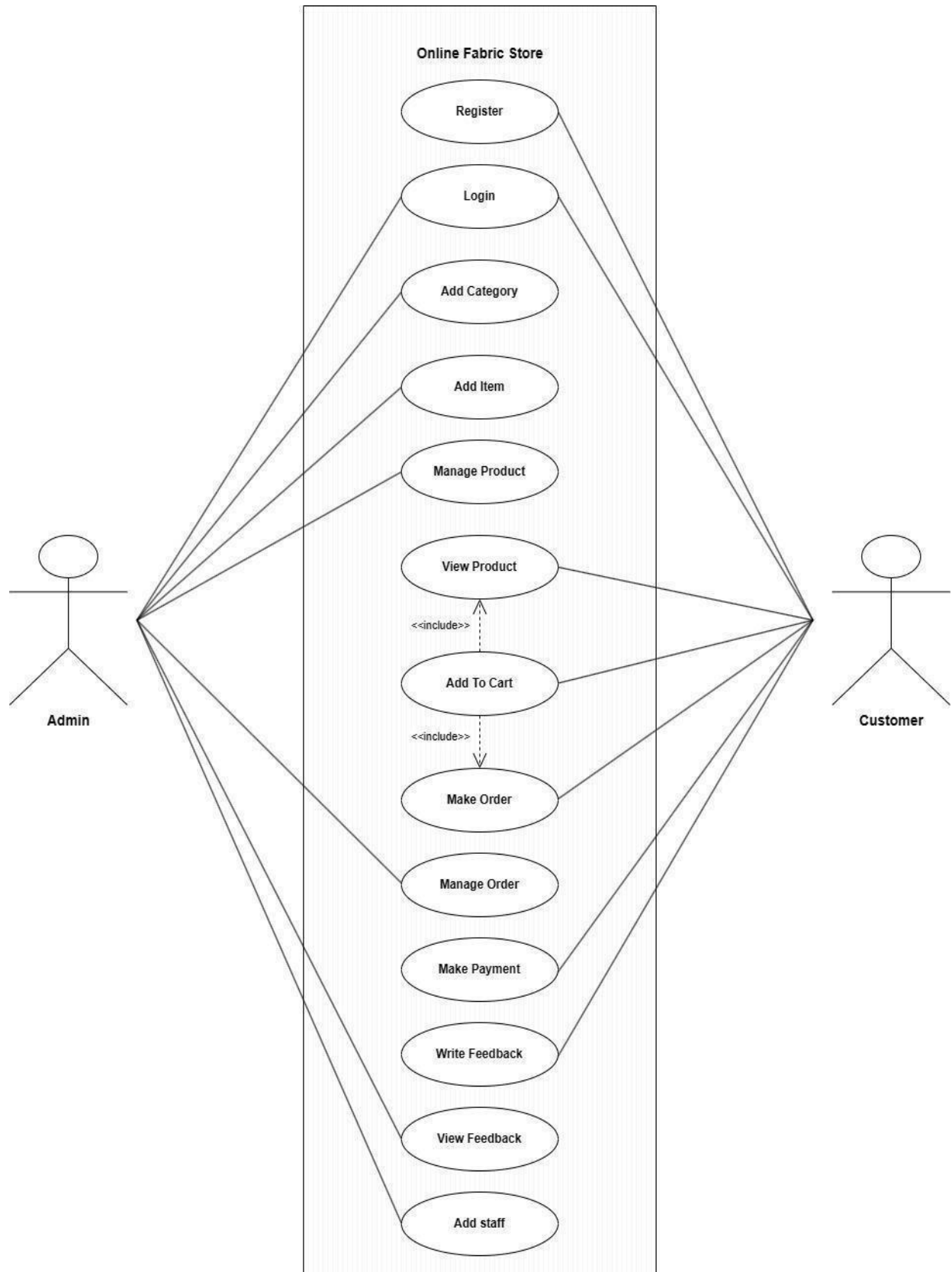
The Entity Relationship Diagram finds its application in contemporary database software engineering to visually present the logical architecture of a database. This technique constitutes a relational schema approach to database modelling, harnessed to outline the structure and methodology of a system. This methodology holds particular significance in the realm of database design. The resultant diagram, formulated through this technique, is referred to as an E-R diagram. An Entity-Relationship (ER) diagram stands as a specialized visual tool that elucidates the connections between entities within a database. ER diagrams incorporate distinct symbols to convey three types of information. Firstly, rectangular shapes are frequently employed to symbolize entities. Secondly, diamonds are commonly employed to represent relationships. Lastly, ovals find usage in denoting attributes.



2.7.4 Use Case

Use Case Diagrams embody the system's functionality as perceived by its users. These diagrams are instrumental in requirements gathering and analysis, serving to encapsulate the system's operational capabilities. Use cases are particularly attuned to portraying the system's conduct from an external perspective.

| Symbol | Reference Name |
|---|----------------|
|  | Actor |
|  | Use Case |
|  | System |
|  | Relationship |



2.8 Software Requirement Specification

SRS, denoting Software Requirements Specification, embodies a comprehensive document detailing the anticipated behaviour of a software system. This dossier encompasses functional requisites, along with non-functional mandates like performance targets and descriptors of quality attributes. The Software Requirements Specification, akin to a Business Requirements Specification (CONOPS) or Stakeholder Requirements Specification (StRS), delineates the software system's contours to be conceived. Within it, both functional and non-functional prerequisites are meticulously outlined, sometimes accompanied by a compilation of use cases delineating user-system interactions that the software must deliver to facilitate seamless engagement. The Software Requirements Specification serves as the bedrock for an accord between clients and contractors or suppliers, encapsulating how the software product should operate. (In endeavours driven by market considerations, these roles could correspond to marketing and development divisions.) This document emerges as a stringent assessment of requirements preceding the more intricate phases of system design, aiming to curtail subsequent overhauls. It further furnishes a pragmatic foundation for gauging product expenses, vulnerabilities, and timelines. Notably, the Software Requirements Specification catalogue itemizes sufficient and essential prerequisites integral to the project's evolution. To discern these requisites, the developer must attain a lucid, in-depth comprehension of the product in development. This is accomplished through comprehensive and continuous dialogues with the project team and clients, throughout the continuum of software development.

2.8.1 Introduction

The Virtual Cloth Emporium constitutes an electronic commerce arena facilitating customers to explore, opt for, and acquire a vast spectrum of apparel articles through online channels. This Software Requirements Specification delineates both the functional and non-functional prerequisites imperative for the fruition of the virtual store.

2.8.2 Scope The Online Cloth Store will provide the following features:

- **User Enrolment and Sign-In:** Customers possess the capability to register accounts and log in, accessing tailored functionalities and order tracking.

- **Item Collection:** Present an extensive roster of garment selections available for purchase, encompassing categories, dimensions, shades, and costs.
- **Exploration and Refinement:** Afford users the facility to seek specific items, augmented by filtering options to fine-tune search results grounded on diverse attributes.
- **Item Particulars:** Furnish intricate insights about each attire piece, spanning visuals, explanations, patron appraisals, and associated products.
- **Purchase Cart:** Empower users to incorporate items into their shopping carts, evaluate cart contents, and modify quantities prior to advancing to the checkout phase.
- **Protected Checkout:** Integrate a secure payment gateway, facilitating customers in finalizing their transactions.
- **Administrator Console:** Outfit store proprietors with an administrative interface to oversee products, monitor orders, and supervise user engagement.

2.8.3 Functional Requirements

2.8.3.1 User Registration and Login

- Fresh participants can enrol on the platform by submitting key particulars like their name, email, and password.
- For those already engaged, access is granted through entering their registered email and password.

2.8.3.2 Product Catalogue

- The platform will present an array of fabric commodities, featuring accompanying visuals and concise portrayals.
- Items shall be systematically organized into distinctive segments (e.g., Seasonal fabrics like summer, winter, rainy, etc.).
- Each product will be endowed with attributes encompassing dimensions, hues, and pricing.

2.8.3.3 Search and Filter • Users retain the capacity to initiate product searches through targeted keywords.

- The platform will extend users the privilege of product filtration, contingent on factors such as categories, quantities, hues, and pricing.

2.8.3.4 Product Details

- Opting for a product will unveil an extensive dossier comprising manifold visuals, a thoroughgoing delineation, customer evaluations, and linked items.

2.8.3.5 Shopping Cart • Participants have the capacity to include items within their shopping cart.

- The platform will showcase the prevailing cart contents, along with the aggregate sum.
- Users retain the autonomy to adjust item quantities within the cart or execute complete removals.

2.8.3.6 Secure Checkout

- The system shall provide a secure payment gateway to enable users to enter their payment information (credit card, PayPal, etc.) and complete the purchase.

2.8.3.8 Admin Panel

- The admin panel shall allow authorized administrators to manage the product catalogue, including adding, updating, or removing products.
- Admins can view and process customer orders.

2.8.4 Non-functional Requirements

2.8.4.1 Performance

- The platform shall handle a large number of concurrent users and maintain responsiveness under heavy load.

2.8.4.2 Security

- User passwords shall be securely stored using industry-standard encryption techniques.
- The payment gateway shall comply with Payment Card Industry Data Security Standard (PCI DSS) requirements.

2.8.4.3 Usability

- The user interface shall be intuitive and user-friendly to ensure a positive shopping experience.

2.8.5.1 Admin Modules:

- **Login:** Admin log in who has the control over all other accounts i.e., customer account. This is used to login into the system. Admin, customer should provide their credentials. It should contain following fields
 - Username
 - Password

Form is validated as below.

- Both fields should contain data.
- If credentials are correct home page should be displayed and login page otherwise.
- **Integrate Merchandise:** The repository boasts an array of diverse products. These products can be categorized through distinct naming conventions. Admins hold the authority to introduce novel products into the existing framework, complete with their respective particulars.
- **Inspect Inventory:** Admins gain access to a comprehensive overview encompassing all system users. Every user's particulars, except passwords, are discernible within this register.
- **Oversee Merchandise:** Admins are empowered to rearrange, eliminate, and supplement products in accordance with prevailing trends or the stock status of a specific product.
- **Gauge Customer Base:** Admins can scrutinize all customers who have engaged with the online store.
- **Govern Clientele:** Admins are vested with the prerogative to expunge or inhibit customers if any fraudulent activities are detected.
- **Administer Orders:** Admins retain the capacity to eliminate orders from the product lineup once the item is dispatched. This encompasses capturing, tracing, and fulfilling customer orders.

2.8.5.2 Customer Modules:

- **Enrollment:** Users are required to enroll within the system by submitting fundamental particulars, a prerequisite for accessing the product repository.
- **Log In:** Post-enrollment, customers can access the online store by entering valid login credentials. □ **Profile Management:** Users have the capacity to append personal data, including names and addresses, and can effectuate adjustments as needed.

- **Browse Merchandise:** Customers can peruse an array of product listings by leveraging names as search criteria, obtaining comprehensive insights encompassing pricing, variations, discounts, and more.
- **Incorporate into Cart:** A provision is furnished enabling users to incorporate desired items within their shopping cart, thereby facilitating an uninterrupted shopping journey.
- **Order Placement:** After selecting their preferred products, users can proceed to finalize their purchase by placing an order.
- **Payment Gateway:** This functionality enables customers to effectuate payment for their acquisitions using diverse methods, encompassing credit cards, debit cards, e-wallets, and other online payment alternatives. Sensitive customer data is encrypted and securely transmitted amid the app, the customer's financial institution, and the payment processor.

2.9 Technical Specification

2.9.1 Languages used

CSS: Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) operates as a style sheet language exclusively tasked with elucidating how a document, authored within a markup language like HTML, is presented. In conjunction with HTML and JavaScript, CSS stands as a cornerstone technology underpinning the World Wide Web (WWW). Distinct from HTML, CSS possesses the versatility to tailor presentation across diverse mediums, encompassing expansive screens, compact displays, and even print formats. The paramount design intention behind CSS is to segregate presentation from content—embracing layout, color schemes, and typography. This segregation yields enhanced content accessibility, heightened flexibility, and precise control over presentation attributes. CSS ushers in the potential for multiple web pages to share formatting by prescribing pertinent CSS instructions within a separate .css file, mitigating intricacy and redundancy within the structural content. Additionally, the demarcation of formatting and content grants the capacity to exhibit the identical markup page in varying styles to accommodate diverse rendering modalities—on-screen or through CSS-stipulated alternate styles for mobile devices. The term 'cascading' finds its roots in the priority framework that ascertains the application of style rules when multiple rules align with a specific element. This system of cascading priorities cultivates a sense of predictability.

React JS

ReactJS, alternatively referred to as React, stands as an open-source JavaScript library purpose-built for crafting user interfaces (UIs). Conceived by Facebook and introduced to the public in 2013, React revolves around the creation of reusable UI components and the adept management of application states. The cardinal objective of React revolves around delivering a declarative and resource-efficient approach for constructing intricate UIs. To achieve this, it harnesses a virtual Document Object Model (DOM) to effectuate efficient updates and rendering of components. Rather than directly manipulating the authentic DOM, ReactJS orchestrates the computation of minimal modifications essential to refresh the UI and then applies these adjustments to the virtual DOM, culminating in streamlined rendering.

Java Script

JavaScript emerges as a programming language that finds widespread application within web development endeavors. Initially conceived by Netscape, its purpose was to introduce dynamic and interactive elements to websites. While Java's influence on JavaScript is palpable, the syntax finds more commonalities with C. JavaScript stands as a text-based programming language operating on both the client-side and server-side, allowing for the creation of interactive web pages. Its implementation heightens the user experience by transforming static web content into a dynamic and interactive landscape.

JavaScript functions as a client-side scripting language, implying that the client's web browser processes its source code instead of the web server. This enables JavaScript functions to execute post-page loading, all without necessitating communication with the server. These functions often encompass pre-submit checks on web forms to ensure the fulfillment of mandatory fields. In such cases, JavaScript code can trigger error messages prior to transmitting data to the server. Analogous to server-side scripting languages such as PHP and ASP, JavaScript code can be seamlessly integrated anywhere within a webpage's HTML structure. However, only the output of server-side code becomes visible in the HTML rendering, whereas JavaScript code remains discernible within the webpage's source code. It can be referenced through a separate file with the .JS extension, also accessible for examination within the browser.

2.10 Hardware/Software Requirement Specification

2.10.1 Developing Desktop Requirements □ Operating System:

64- or 32-bit OS Windows 7 and above

2.10.1.1 Hardware/Software used for the client

Hardware

- Processor: 1 GHZ or higher CPU.
- Hard disk: 500 MB available internal storage.
- Memory: 512 MB of RAM is minimally recommended.
- Display: 2.8 inches or larger.

Software

- Operating System: Windows 7 or above
- Web Server: XAMPP
- Programming Languages: React JS, CSS, Java Script
- Web browsers: Google chrome/Mozilla Firefox/internet explorer
- Database: MYSQL

2.10.1.2 Hardware/Software used for the server

- **Operating System:** Linux, Unix, Windows
- **Web Server:** Apache Web Server

2.10.2 Constraints □ The system is designed to exclusively enable customer access post-registration.

□ Updating the system with fresh customer data should be seamlessly accommodated.

□ A roster featuring the most recent order updates will be visibly presented by the system.

3. System Design

3.1 Introduction

The objective inherent in the design process is to yield a model representation of a system, a blueprint poised for future system construction. System design constitutes the strategic delineation of a system's architecture, modules, interfaces, and data to effectively meet predetermined specifications. In essence, systems design can be perceived as the practical application of systems theory in the realm of development. This process entails translating software requirements into a structured portrayal of software constituents, inclusive of interfaces and requisite data, poised for implementation.

Contained within the system design document lies an intricate tapestry of the system's prerequisites, operational context, architecture at both system and subsystem levels, input and output formats, interplay between humans and machines, meticulous design specifications, processing logic, and external interfaces. Typically, the design phase unfolds across the following two sequential stages:

3.1.1 Primary Design Phase

During this stage, the system undergoes block-level design, wherein these blocks are formulated based on the analyses conducted during the problem identification phase. Distinct blocks are crafted to correspond to distinct functionalities, with the primary aim of curtailing information exchange between these blocks. Consequently, efforts are focused on consolidating activities demanding heightened interaction into a singular block.

3.1.2 Secondary Design Phase

In this phase, where the detailed design of every block is performed.

3.2 Database design

Database design entails the systematic arrangement of data in alignment with a specified database model. The designer delineates how data must be stored and establishes interrelationships between data elements. Armed with this insight, they commence the process of harmonizing data with the database model. The operational management of data falls under the purview of a database management system (DBMS). The term 'database design' can encompass various facets of an overarching database system's design. Primarily, it signifies

the logical blueprinting of foundational data structures intended for data storage. In the context of the relational model, these structures encompass tables and views. However, the terminology 'database design' also extends to encompass the broader endeavor of designing not just data structures, but also forms and queries integral to the comprehensive database application hosted within the DBMS.

3.3 Data Dictionary

3.3.1 User table

| Column | Type | Null | Default | Example |
|-------------|--------------|------|---------|------------------|
| Id(Primary) | int(11) | No | None | 001 |
| name | varchar(50) | No | None | Ajay |
| email | varchar(30) | No | None | ajay12@gmail.com |
| phone | varchar(15) | No | None | 9876543267 |
| password | varchar(50) | No | None | ***** |
| address | varchar(500) | No | None | Virajpet |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null |
|---------|-------|--------|--------|--------|-------------|-----------|------|
| PRIMARY | BTREE | Yes | No | id | 3 | A | No |

3.3.2 Cart Table

| Column | Type | Null | Default | Example |
|-------------|--------------|------|---------|-------------------|
| id(Primary) | int(11) | No | None | 01 |
| name | varchar(50) | No | None | Cinan |
| email | varchar(30) | No | None | cinankl@gmail.com |
| phone | varchar(15) | No | None | 9786567435 |
| product | varchar(100) | No | None | Chiffon Fabric |
| price | int(11) | No | None | 599 |
| status | int(11) | No | None | 1 |
| colour | varchar(10) | No | None | Green |
| meter | int(15) | No | None | 5 |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null |
|---------|-------|--------|--------|--------|-------------|-----------|------|
| PRIMARY | BTREE | Yes | No | id | 58 | A | No |

3.3.3 Category

| Column | Type | Null | Default | Example |
|-------------|--------------|------|---------|---------|
| id(Primary) | int(11) | No | None | 1 |
| category | varchar(100) | No | None | Value 9 |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null |
|---------|-------|--------|--------|--------|-------------|-----------|------|
| PRIMARY | BTREE | Yes | No | id | 3 | A | No |

3.3.4 Feedback Table

| Column | Type | Null | Default | Example |
|-------------|--------------|------|---------|-------------------|
| Id(Primary) | int(11) | No | None | 001 |
| email | varchar(30) | No | None | ajay12@gmail.com |
| product | varchar(100) | No | None | 9876543267 |
| message | varchar(500) | No | None | Very good product |
| date | varchar(10) | No | None | 12/6/2023 |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null |
|---------|-------|--------|--------|--------|-------------|-----------|------|
| PRIMARY | BTREE | Yes | No | id | 0 | A | No |

3.3.5 Orders Table

| Column | Type | Null | Default | Example |
|-------------|--------------|------|---------|-----------------------|
| id(Primary) | int(11) | No | None | 001 |
| name | varchar(50) | No | None | Seema |
| email | varchar(30) | No | None | seemashetty@gmail.com |
| phone | varchar(15) | No | None | 9787654517 |
| address | varchar(500) | No | None | Mangalore |
| product | varchar(100) | No | None | Poplin |
| price | int(11) | No | None | 25000 |
| colour | varchar(10) | No | None | Blue |
| status | int(11) | No | None | 2 |
| meter | int(15) | No | None | 50 |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null |
|---------|-------|--------|--------|--------|-------------|-----------|------|
| PRIMARY | BTREE | Yes | No | id | 5 | A | No |

3.3.6 Staff Table

| Column | Type | Null | Default | Example |
|-------------|-------------|------|---------|----------------------|
| id(Primary) | int(11) | No | None | 002 |
| name | varchar(50) | No | None | Rudra |
| email | varchar(50) | No | None | rudracarto@gmail.com |
| phone | varchar(50) | No | None | 7022334487 |
| password | varchar(50) | No | None | ***** |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null |
|---------|-------|--------|--------|--------|-------------|-----------|------|
| PRIMARY | BTREE | Yes | No | id | 1 | A | No |

3.3.7 Product Table

| Column | Type | Null | Default | Example |
|-------------|--------------|------|---------|-------------------|
| id(Primary) | int(11) | No | None | 001 |
| category | int(11) | No | None | Winter Collection |
| name | varchar(100) | No | None | Muslin |
| quantity | int(11) | No | None | 540 |
| price | int(11) | No | None | 65 |
| status | int(11) | No | None | 1 |
| image_red | varchar(100) | No | None | muslinred.jpg |
| image_blue | varchar(100) | No | None | muslinblue.jpg |
| image_green | varchar(100) | No | None | muslingreen.jpg |
| state | int(11) | No | None | 3 |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null |
|---------|-------|--------|--------|--------|-------------|-----------|------|
| PRIMARY | BTREE | Yes | No | id | 9 | A | No |

3.3.8 State Table

| Column | Type | Null | Default | Example |
|-------------|-------------|------|---------|-----------|
| id(Primary) | int(11) | No | None | 003 |
| state | varchar(50) | No | None | Karnataka |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null |
|---------|-------|--------|--------|--------|-------------|-----------|------|
| PRIMARY | BTREE | Yes | No | id | 4 | A | No |

4. Software Testing

4.1 Introduction

Embedded within the software development process, software testing emerges as an indispensable facet dedicated to validating the accuracy, thoroughness, and caliber of the evolved software. Its principal aim orbits around identifying discrepancies within the software—a pivotal step to guarantee its precision. Deficiencies in software functionality obstruct its capacity to yield outputs congruent with user requisites. These deficiencies, commonly referred to as 'errors,' materialize when any facet of the software exhibits incompleteness, incongruity, or inaccuracy. The expanse of errors can be categorized into three primary types: requirement errors, design errors, and programming errors. Consequently, forestalling the emergence of these errors necessitates meticulous evaluation of requirements for alignment with user demands, assiduous adherence of software design to requirements and notational standards, and a scrupulous examination of source code to ensure conformance to stipulated requirements, design documentation, and user expectations. These objectives find attainment through the efficacious mechanisms embedded within software testing.

4.2 Test Plan

A test plan functions as a strategic blueprint, charting the course for the implementation of testing protocols. This all-encompassing document encapsulates the intent, parameters, methodologies, and rationale underpinning the exertions of software testing. Grounded within this framework, the test plan designates the spectrum of test items, delineates the pivotal facets slated for assessment, elaborates upon the prescribed testing protocols, and assigns the personnel entrusted with executing these evaluative undertakings. Additionally, the plan underscores the precise test environment and methodologies earmarked for application. Ultimately, a meticulously crafted test plan crystallizes into a shared understanding between testers and users, shedding light on the pivotal role that testing plays within the domain of software.

4.3 Test case design

Distinct test cases need to be meticulously formulated for each individual module. The overarching strategies delineated within the test plan are further detailed in the specific

test methodologies to be employed, along with the criteria by which evaluations will be gauged. The construction of a test case design entails the creation of a comprehensive test scenario for every unit under examination. This includes the definition of inputs to be utilized within the test case, the parameters to be scrutinized, and the anticipated outcomes to be associated with that particular test case.

4.3.1 Methods for Test case design

There are 2 methods of test case design:

- White Box testing
- Black Box testing

White Box testing

White-box testing, often referred to as clear-box testing or structural testing, constitutes a software testing approach that scrutinizes the inner architecture and logical constructs of the software undergoing evaluation. In contrast to black-box testing, which centres on the external behaviour of the application, white-box testing delves into the intricacies of the software's source code and design, unravelling its operational mechanisms. The core aim of white-box testing centres on validating the testing of all possible paths and conditions embedded within the code, thus culminating in a thorough test coverage. This practice is typically orchestrated by developers, specialized testers, or automated testing tools.

Black Box testing

Black-box testing constitutes a software testing methodology wherein the evaluator assesses the software's functionality devoid of any access to its underlying code or structure. In this paradigm, the focal point resides in scrutinizing the software's input-output dynamics, while disregarding the intricacies of its internal design. The nomenclature "black-box" derives from the concept that the evaluator perceives the software as a sealed entity, impervious to insights into its internal mechanics.

4.4 Testing Strategy

The formulation of a software testing strategy is pivotal for conducting testing in an organized and methodical manner. This testing strategy delineates the hierarchy of testing levels to be employed, outlines the methodologies and techniques, and designates the tools

to be leveraged. Furthermore, the strategy shapes the selection of test cases, specifications, decisions regarding test cases, and consolidates the groundwork for their subsequent execution.

4.5 Validation Criteria

The validation criteria are as follows:

In the context of the Online Fabric Store system, input data provided by users undergoes validation prior to storage and subsequent display. The ensuing validations performed within this system encompass the following aspects:

A uniform and consistent visual appearance characterizes all screens, featuring harmonized colour combinations in the background to enhance the user experience.

- Preventing the duplication of primary key values is enforced as a core validation rule.
- For enhanced user convenience, all entries within combo boxes have been organized in alphabetical order, facilitating streamlined selection from these dropdown menus.

4.6 Importance of testing

System testing is a pivotal phase where the software is subjected to experimental scrutiny to confirm its robustness. In essence, this phase validates whether the software adheres to specifications and performs in alignment with user expectations. To facilitate this, unique test data is fed into the system for processing, and the resultant outcomes are meticulously evaluated. The significance of system testing is underscored by the need to ensure the software's congruence with user demands prior to its deployment. This iterative process revolves around subjecting the system to repeated rounds of testing against specifications to avert any potential failure upon implementation at the user's end.

5. Coding

5.1 Introduction

The coding or programming phase is dedicated to transmuting the system's design, conceived during the design phase, into executable code within a specific programming language. This code is intended to facilitate computer execution and execute the computational tasks specified by the design. The coding phase exercises a substantial influence over both the testing and maintenance stages. While the coding duration constitutes a minor fraction of the overall software cost, the lion's share is absorbed by the subsequent testing and maintenance phases. This dynamic underscores the paramount importance of coding with a focus not on curtailing implementation expenses, but rather on mitigating costs in later phases. Consequently, the objective should not merely simplify the programmer's task; instead, it should center around streamlining the tester's and maintainer's responsibilities.

5.2 Code List

Admin Login

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
function AdminLogin() {

  const navigate = useNavigate();

  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");

  const myStyle={
    backgroundImage:
    "url('https://i.pinimg.com/originals/fd/21/f3/fd21f3be5eb42fe18fae05422f297719.jpg')", height:'100vh', //
    marginTop:'-70px',
    fontSize:'50px',
    backgroundSize: 'cover',
    backgroundRepeat: 'no-repeat',
  };

  const adminLogin = () => { if (username === "admin" &&
    password === "123456") { navigate("/admin/orders");
```

```

    } else { alert("Invalid Login
    Credentials"); }
  }

  return (
    <div className="vh-100 f4" style={myStyle} >
      <div className="d-flex justify-content-center f4">
        <div class="card w-25 my-5 f4">
          <div class="card-header f4">
            <h5>Admin Login</h5>
          </div>
          <div class="card-body">
            <input type="text" placeholder="Username"
            className="form-control mb-3" value={username} onChange={(e) =>
            setUsername(e.target.value)} />
            <input type="password" placeholder="Password"
            className="form-control mb-3" value={password} onChange={(e) =>
            setPassword(e.target.value)} />
            <button type="button" className="btn btn-primary"
            onClick={adminLogin}>Login</button>
          </div>
        </div>
      </div>
    </div>
  );
}
export default AdminLogin;

```

Admin Panel

```

import React from 'react'; import { Route, Routes }
from 'react-router-dom'; import AdminNavbar from
'../components/AdminNavbar'; import AdminFeedback
from './AdminFeedback'; import AllOrders from
'./AllOrders'; import ManageProduct from
'./ManageProduct'; import ManageStaff from
'./ManageStaff'; import AdminLogin from
'./AdminLogin' function AdminPanel() { return (
  <>
    <AdminNavbar />
    <Routes>
      <Route path="/" element={<AdminLogin />} />

      <Route path="/orders" element={<AllOrders />} />
      <Route path="/products" element={<ManageProduct />} /> <Route
      path="/feedbacks" element={<AdminFeedback />} />

      <Route path="/staffs" element={<ManageStaff />} />

    </Routes>
  </>
)
}

```

```

    </>
  );
}
export default AdminPanel;

```

Admin Feedback

```

import Axios from 'axios';
import React, { useEffect, useState } from 'react';

function AdminFeedback() {

  const [feedbackList, setFeedbackList] = useState([]);

  useEffect(() => {
    Axios.get("http://127.0.0.1:3001/getfeedbacks", {
    }).then((response) => {
      setFeedbackList(response.data);
    }); },
  []);

  return (
    <div className="container">
      <div className="row g-4 my-4">
        <div className="col-12">
          <h3 className="title-sm">Feedback</h3>
        </div>
        <div className="col-12">
          <table className="table">
            <thead>
              <tr>
                <th>#</th>
                <th>Customer Email</th>
                <th>Product Name</th>
                <th>Message</th>
                <th>Date</th>
              </tr>
            </thead>
            <tbody>
              { feedbackList.map((data, key) => {
                return (
                  <tr>
                    <td>{key + 1}</td>
                    <td>{data.email}</td>
                    <td>{data.product}</td>
                    <td>{data.message}</td>
                    <td>{data.date}</td>
                  </tr>
                );
              })
            }
          </tbody>
        </table>
      </div>
    </div>
  );
}

```

```

    ))
    }
    </tbody>
  </table>
</div>
</div>
</div>
);
}
export default AdminFeedback;

```

All Orders

```

import React, { useState, useEffect } from 'react';
import Axios from 'axios';
function AllOrders() { const [orderList, setOrderList] =
  useState([]);

  const getMyOrders = () => {
    Axios.get("http://127.0.0.1:3001/getorders",)
      .then((response) => {
        console.log(response);
        setOrderList(response.data)
      });
  }

  useEffect(() => {
    getMyOrders(); },
    []);

  const orderStatus = (id, status, product, email, meter) => {
    Axios.post('http://127.0.0.1:3001/orderstatus',
      { id: id, status: status, product: product,
        email: email,
        meter: meter,
      }).then((response) => { if
        (response.data === "success") {
          getMyOrders();
        }
        else if (response.data === "error") {
          alert("You Reached Limit");
        }
      });
  }
}

```



```

const deleteOrder = (id) => { if (window.confirm("Are You Sure Want to
Delete ?") === true) {
    Axios.post('http://127.0.0.1:3001/deleteorder', {
        id: id,
    }).then((response) => {
        getMyOrders();
    });
}
}

return (
    <div className="container">
        <div className="row g-4 my-4">
            <div className="col-12">
                <h3 className="title-sm">Recent Orders</h3>
            </div>
            <div className="col-12">
                <table className="table">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Customer Name</th>
                            <th>Email Address</th>
                            <th>Phone Number</th>
                            <th>Address</th>
                            <th>Product Name</th>
                            <th>Price</th>
                            <th>Colour</th>
                            <th>Meters</th>
                            <th>Action</th>
                            <th>Delete</th>
                        </tr>
                    </thead>
                    <tbody>
                        { orderList.map((data, key) => {
                            return (
                                <tr>
                                    <td>{key + 1}</td>
                                    <td>{data.name}</td>
                                    <td>{data.email}</td>
                                    <td>{data.phone}</td>
                                    <td>{data.address}</td>
                                    <td>{data.product}</td>
                                    <td>{data.price}</td>
                                    <td>{data.colour}</td>
                                    <td>{data.meter}</td>
                                    <td>{parseInt(data.status) ? <button className="btn btn-success" disabled
                                    >Completed</button> : <button className="btn btn-warning" onClick={() =>

```

```

orderStatus(data.id, "1", data.product, data.email,
data.meter)}}>Complete</button>}</td>
                                <td><button className="btn btn-
danger" onClick={() => deleteOrder(data.id)}>Delete</button></td>
                                /* <td>
                                  <button type="button"
className="btn btn-danger" onClick={() => delStaff(data.id)} > Delete</button>
                                  </td> */
                                </tr>
                                );
                                })
                                }
                                </tbody>
                                </table>
                                </div>
                                </div>
                                </div>
                                );
                                }
export default AllOrders;

```

Manage Products

```

import React, { useEffect, useState } from 'react';
import Axios from 'axios';
function ManageProduct() {

  const [categoryList, setCategoryList] = useState([]);
  const [statelist, setStateList] = useState([]);
  const [category, setCategory] = useState("");
  const [state, setState] = useState("");

  const [name, setName] = useState(""); const
  [quantity, setQuantity] = useState("");
  const [price, setPrice] = useState("");

  const [redFile, setRedFile] = useState();
  const [redFileName, setRedFileName] =
  useState("");

  const [blueFile, setBlueFile] = useState();
  const [blueFileName, setBlueFileName] = useState("");

  const [greenFile, setGreenFile] = useState(); const
  [greenFileName, setGreenFileName] = useState("");
  const [productList, setProductList] = useState([]);

  let isValid = true;

```

```
const [errorMsg, setErrorMsg] = useState("");

const makeValidation = () => {
  if (category === "") {
    isValid = false;
    setErrorMsg("Category is required");
  } else if (state === "") {
    isValid = false;
    setErrorMsg("State is required");
  }
  else if (name === "") {
    isValid = false;
    setErrorMsg("Product Name is required");
  }
  else if (quantity === "") {
    isValid = false;
    setErrorMsg("Quantity is required");
  }
  else if (price === "") {
    isValid = false;
    setErrorMsg("Price is required");
  }
  else if (redFileName === "") {
    isValid = false;
    setErrorMsg("Image is required");
  }
  else if (blueFileName === "") {
    isValid = false;
    setErrorMsg("Image is required");
  }
  else if (greenFileName === "") {
    isValid = false;
    setErrorMsg("Image is required");
  } else { isValid =
    true;
    setErrorMsg("");
  }
}

useEffect(() => {
  getMyProducts();
}, []);

const saveRedFile = (e) => {
  setRedFile(e.target.files[0]);
  setRedFileName(e.target.files[0].name);
};

const saveBlueFile = (e) => {
  setBlueFile(e.target.files[0]);
  setBlueFileName(e.target.files[0].name);
}
```

```

};

const saveGreenFile = (e) => {
  setGreenFile(e.target.files[0]);
  setGreenFileName(e.target.files[0].name);
};

useEffect(() => {
  Axios.get("http://127.0.0.1:3001/getcategories", {
  }).then((response) => {
    setCategoryList(response.data);
  }); },
  []);

const getMyProducts = () => {
  Axios.get("http://127.0.0.1:3001/getmyproducts", {
  }).then((response) => {
    setProductList(response.data);
  });
}

useEffect(() => {
  Axios.get("http://127.0.0.1:3001/getstates", {
  }).then((response) => {
    setStateList(response.data);
  });
}, []);

const saveProduct = () => { makeValidation(); if
(isValid) { const formData = new FormData();
formData.append("category", category);
formData.append("name", name);
formData.append("quantity", quantity);
formData.append("price", price);
formData.append("redfile", redFile);
formData.append("redfilename", redFileName);
formData.append("bluefile", blueFile);
formData.append("bluefilename", blueFileName);
formData.append("greenfile", greenFile);
formData.append("greenfilename", greenFileName);
formData.append("status", 1);
formData.append("state", state);

    Axios.post("http://127.0.0.1:3001/saveproduct",
      formData
    ).then((response) => {
      getMyProducts();
    });
  }
}

```

```

const changeProductStatus = (id, status) => {
  Axios.post('http://127.0.0.1:3001/productstatus',
    { id: id, status: status,
  }).then((response) => {
    getMyProducts();
  });
}

const delProduct = (id) => { if (window.confirm("Are You Sure Want
to Delete ?") === true) {
  Axios.post('http://127.0.0.1:3001/delproduct', {
    id: id,
  }).then((response) => {
    getMyProducts();
  });
}
}
return (
  <div className="container-fluid">
    <div className="row justify-content-center">
      <div className="col-4 my-5">
        <div className="card">
          <div className="card-header">
            <h4>Add Products</h4>
          </div>
          <div className="card-body">
            {
              errorMsg ?
                <div class="alert alert-danger alert-msg-
cst mb-3" role="alert">
                  {errorMsg}
                </div> : ''
            }
            <label className="mb-2">Select Category</label>
            <select className="form-select mb-3" aria-
label="Default select example" onChange={(e) => setCategory(e.target.value)}>
              <option value="" selected>Select
Category</option>
              { categoryList.map((data, key) => {
                return ( <option
value={data.id}>{data.category}</option>
                );
              })
            }
            </select>

            <label className="mb-2">Select State</label>
            <select className="form-select mb-3" aria-
label="Default select example" onChange={(e) => setState(e.target.value)}>

```

```

        <option value="" selected>Select
State</option>
        { stateList.map((data, key) => {
            return ( <option
value={data.id}>{data.state}</option>
                );
            })
        }
    </select>

    <label className="mb-2">Product Name</label>
    <input type="text" className="form-control mb-3"
value={name} onChange={(e) => setName(e.target.value)} />
    <label className="mb-2">Product Quantity (In
Lengths)</label>
    <input type="number" className="form-control mb-3"
value={quantity} onChange={(e) => setQuantity(e.target.value)} />
    <label className="mb-2">Product Price</label>
    <input type="number" className="form-control mb-3"
value={price} onChange={(e) => setPrice(e.target.value)} />

    <label className="mb-2">Upload Red Image</label>

    <input type="file" className="form-control mb-3"
onChange={saveRedFile} />

    <label className="mb-2">Upload Blue Image</label>

    <input type="file" className="form-control mb-3"
onChange={saveBlueFile} />

    <label className="mb-2">Upload Green Image</label>

    <input type="file" className="form-control mb-3"
onChange={saveGreenFile} />

    <button type="button" className="btn btn-primary"
onClick={saveProduct}>Save</button>
    </div>
  </div>
</div>
<div className="col-8 my-5">
  <table className="table">
    <thead>
      <tr>
        <th>#</th>
        <th>ProductName</th>

```

```

        <th>Length</th>
        <th>Product Price</th>
        <th>Action</th>
      </tr>
    </thead>
    <tbody>
      { productList.map((data, key) => {
        return (
          <tr>
            <td>{key + 1}</td>
            <td>{data.name}</td>
            <td>{data.quantity}</td>
            <td>{data.price}</td>
            <td>
              {parseInt(data.status) ?
<button type="button" className="btn btn-warning" onClick={() =>
changeProductStatus(data.id, "0")}>Deactivate</button> : <button
type="button" className="btn btn-success" onClick={() =>
changeProductStatus(data.id, "1")}>Activate</button>}
            </td>
            <td>
              <button type="button"
className="btn btn-danger" onClick={() => delProduct(data.id)} >
Delete</button>
            </td>
          </tr>
        );
      })
    }
  </tbody>
</table>
</div>
</div>
</div>
);
}
export default ManageProduct;

```

Manage Staff

```

import React, { useEffect, useState } from
'react'; import Axios from 'axios'; import {
useNavigate } from 'react-router-dom'; function
ManageStaff() {

  const navigate = useNavigate();

  const [name, setName] = useState(""); const
[email, setEmail] = useState(""); const

```

```
[phone, setPhone] = useState(""); const
[password, setPassword] = useState(""); const
[staffList, setStaffList] = useState([]);

var isValid = true;
const [alertMsg, setAlertMsg] = useState("");

const makeValidation = () =>
{ if (name === "") {
  isValid = false;
  setAlertMsg("Name is
  required");
}
else if (email === "") {
  isValid = false;
  setAlertMsg("Email Address is required");
}
else if (phone === "") {
  isValid = false;
  setAlertMsg("Phone Number is required");
}
else if (password === "") {
  isValid = false;
  setAlertMsg("Password is required");
} else { isValid =
true;
}
}

useEffect(() => {
  getStaffList(); },
  []);

const getStaffList = () => {
  Axios.get("http://127.0.0.1:3001/getstafflist", {
  }).then((response) => {
    setStaffList(response.data);
  });
}

const delStaff = (id) => { if (window.confirm("Are You Sure Want
to Delete ?") === true) {
  Axios.post('http://127.0.0.1:3001/delstaff', {
    id: id,
  }).then((response) => {
    getStaffList();
  });
}
}
```



```

const createAccount = () =>
{
  makeValidation();
  if (isValid) {
    Axios.post('http://127.0.0.1:3001/staffregister',
      { name: name, email: email, phone: phone,
        password: password,
      }).then((response) => {
        getStaffList();
      });
  }
}

return (
  <div className="container-fluid">
    <div className="row justify-content-center">
      <div className="col-4 my-5">
        <div className="card">
          <div className="card-header">
            <h4>Add Staff</h4>
          </div>
          <div className="card-body">
            {alertMsg ? <div className="alert alert-danger
alert-msg-cst" role="alert">
              {alertMsg}</div> : ''}
            <input type="text" className="form-control mb-3"
placeholder="Full Name" value={name} onChange={(e) => setName(e.target.value)}
/>
            <input type="email" className="form-control mb-3"
placeholder="Email Address" value={email} onChange={(e) =>
setEmail(e.target.value)} />
            <input type="number" className="form-control mb-
3"
placeholder="Phone Number" value={phone} onChange={(e) =>
setPhone(e.target.value)} />
            <input type="password" className="form-control
mb-
3" placeholder="Password" value={password} onChange={(e) =>
setPassword(e.target.value)} />
            <button type="button" className="btn btn-dark"
onClick={createAccount}>Create</button>
          </div>
        </div>
      </div>
      <div className="col-8 my-5">
        <table className="table">
          <thead>
            <tr>
              <th>#</th>
              <th>Name</th>
              <th>Email Address</th>
              <th>Phone Number</th>
            </tr>
          </thead>
        </table>
      </div>
    </div>
  </div>
)

```

```

        <th>Password</th>
        <th>Action</th>
      </tr>
    </thead>
    <tbody>
      { staffList.map((data, key) => {
        return (
          <tr>
            <td>{key + 1}</td>
            <td>{data.name}</td>
            <td>{data.email}</td>
            <td>{data.phone}</td>
            <td>{data.password}</td>
            <td>
              <button type="button"
className="btn btn-danger" onClick={() => delStaff(data.id)} > Delete</button>
            </td>
          </tr>
        );
      })
    }
  </tbody>
</table>
</div>
</div>
</div>
);
}
export default ManageStaff;

```

Feedback form

```

import Axios from 'axios'; import React, {
  useState, useEffect } from 'react'; import Navbar
  from '../components/Navbar';
function FeedbackForm() { const [productList,
  setProductList] = useState([]);

  useEffect(() => {
    Axios.get("http://127.0.0.1:3001/getproducts", {
    }).then((response) => {
      setProductList(response.data);
    }); },
  []);

  const [product, setProduct] = useState("");
  const [message, setMessage] = useState("");

  var isValid = true;

```

```

const makeValidation = () => {
  if (product === "") {
    isValid = false;
    alert("Select Product");
  }
  else if (message === "") {
    isValid = false;
    alert("Message is Required");
  } else { isValid = true;
  }
}

const submitFeedback = () =>
{ makeValidation(); if
(isValid) {
  Axios.post("http://127.0.0.1:3001/addfeedback",
    { email: sessionStorage.getItem("EMAIL"),
      product: product, message: message,
      date: new Date().getDate() + "/" + new Date().getMonth() + "/"
+ new Date().getFullYear(),
    }).then((response) => {
      alert("Thanks For Your Feedback")
    });
}
}

return (
  <>
    <Navbar />
    <div className="container">
      <div className="row g-5 my-4">
        <div className="col-12">
          <h3 className="title-sm text-center">Send
Feedback</h3>
          </div>
          <div className="col-12 d-flex justify-content-center
align-items-center flex-column">
            <select className="form-select w-50 mb-4" aria-
label="Default select example" onChange={(e) => setProduct(e.target.value)}>
              <option value="" selected>Select Product</option>
              { productList.map((data, key) => {
                return (
                  <option
value={data.name}>{data.name}</option>
                );
              })
            }
            </select>
            <textarea className="form-control w-50 mb-4" rows="6"

```

```

value={message} onChange={(e) => setMessage(e.target.value)}></textarea>
                    <button type="button" className="btn btn-primary"
onClick={submitFeedback}>Submit</button>
                </div>
            </div>
        </div>
    </>
    );
}
export default FeedbackForm;

```

Forgot Password

```

import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import NavBar from '../components/Navbar';
import Axios from 'axios';
import { useNavigate } from 'react-router-dom';

function ForgotPassword() {

    const navigate = useNavigate();

    const [email, setEmail] = useState("");
    const [password, setPassword] =
    useState("");

    var isValid = true;
    const [alertMsg, setAlertMsg] = useState("");

    const myStyle={
        backgroundImage:
"url('https://i.pinimg.com/originals/fd/21/f3/fd21f3be5eb42fe18fae05422f29771
9.jpg')", height:'100vh', //
        marginTop:'-70px',
        fontSize:'50px',
        backgroundSize:
        'cover',
        backgroundRepeat:
        'no-repeat',
    };
}

```

```

const makeValidation = () => {
  if (email === "") {
    isValid = false;
    setAlertMsg("Email Address is required");
  }
  // else if (password === "") {
  //   isValid = false;
  //   setAlertMsg("Password is required");
  // } else {
  isValid = true;
  }
}

const forgot = () => {
  makeValidation();
  if (isValid) {
    Axios.post('http://127.0.0.1:3001/forgotpassword', {
      email: email,
      password: password,
    }).then((response) => { if
      (Object.keys(response.data).length > 0) {
        //   sessionStorage.setItem("ID", response.data[0].id);
        //   sessionStorage.setItem("NAME", response.data[0].name);
        //   sessionStorage.setItem("EMAIL",
response.data[0].email);
        //   sessionStorage.setItem("PHONE",
response.data[0].phone);
        navigate("/");
      } else { setAlertMsg("Invalid
Email");
      }
    });
  }
}

return (
  <>
    <NavBar />
    <div className=" h4" style={myStyle} >
      <div className="d-flex justify-content-center h4">
        <div className="card w-25 my-5 h4">
          <div className="card-header h4">
            <h5>Forgot Password</h5>
          </div>
          <div className="card-body ">
            {alertMsg ? <div className="alert alert-danger
alert-msg-cst" role="alert">
              {alertMsg}
            </div> : ''}
            <input type="email" className="form-control mb-3

```

```

h4" placeholder="Email Address" value={email} onChange={(e) =>
setEmail(e.target.value)} />
        {/* <input type="password" className="form-
        control
mb-3 h4" placeholder="Password" value={password} onChange={(e) =>
setPassword(e.target.value)} /> */}
        <button type="button" className="btn btn-dark h4"
onClick={forgot}>Submit</button>
        {/* <Link to="/register" type="button"
className="btn btn-outline-primary ms-2 h4">Register</Link>
        <Link to="/forgot" type="link" className=" link
sm ms-2 h7" >Forgot Password</Link> */}

        </div>

    </div>
  </div>
</div>
</>
);
}
export default ForgotPassword;

```

Home

```

import React, { useEffect, useState } from
'react'; import Axios from 'axios'; import Navbar
from '../components/Navbar'; import { useNavigate
} from 'react-router-dom';
function Home() {

    const navigate = useNavigate(); const
[meter, setMeter] = useState("");

    const [productList, setProductList] =
useState([]); const [cartList, setCartList] =
useState([]); const [customerName, setCustomerName] =
useState(sessionStorage.getItem("NAME"));
    const [emailAddress, setEmailAddress] =
useState(sessionStorage.getItem("EMAIL"));
    const [phoneNumber, setPhoneNumber] =
useState(sessionStorage.getItem("PHONE")); const [address, setAddress] =
useState(sessionStorage.getItem("ADDRESS")); const [selectColor,
setSelectColor] = useState("Red");

    useEffect(() => {
        Axios.get("http://127.0.0.1:3001/getproducts", {
        }).then((response) => {
            setProductList(response.data);
        });
    });

```

```

    }, []);

    useEffect(() => { localStorage.setItem('MYCART',
      JSON.stringify(cartList));
    }, [cartList]);

    const addToCart = (name, price) => { if
      (sessionStorage.getItem("EMAIL")) { alert("Added Successfully")
      setCartList([...cartList, { name: name, price: price, colour:
selectColor }]);

      Axios.post('http://127.0.0.1:3001/addtocart',
        { name: customerName, email: emailAddress,
        phone: phoneNumber, colour: selectColor,
        meter: meter, product: name, price: price,
        status: "0",
        }) } else {
      navigate("/login");
    }
  }

  return (
    <>
    <Navbar />
    <div className="container">
      <div className="row g-5 my-4">
        <div className="col-12 text-center">
          <h3 className="title-sm">All Fabrics Collections</h3>
        </div>
        <div className="col-12 text-end">
          <button type="button" className="color-btn"
onClick={() => setSelectColor("Red")}>Red</button>
          <button type="button" className="color-btn"
onClick={() => setSelectColor("Blue")}>Blue</button>
          <button type="button" className="color-btn"
onClick={() => setSelectColor("Green")}>Green</button>
        </div>
        <div className="col-12">
          <div className="row row-cols-4">
            {
              productList.map((data, key) => {
                return (
                  <div className="col">
                    <div className="card h-100
product-card">


{ selectColor === "Red" ?
                        <img
src={"http://127.0.0.1:3001/images/" + data.image_red} alt="product"
className="img-fluid" /> : ''
                    }


```

```

        { selectColor === "Blue" ?
          <img
src={"http://127.0.0.1:3001/images/" + data.image_blue} alt="product"
className="img-fluid" /> : ''
        }
        { selectColor === "Green" ?
          <img
src={"http://127.0.0.1:3001/images/" + data.image_green} alt="product"
className="img-fluid" /> : ''
        }
        <div className="card-body">

          <h5 className="card-
title">{data.name}</h5>
          <p className="card-text- bold mb-2">₹ {data.price}</p>
          {data.quantity ? <p
className="card-text">Available Quantity: {data.quantity} Meters</p> : <p
className="card-text fw-bold text-danger">Not Available</p>}
          {data.quantity ? <p
className="card-text">Enter Meters: <input type="text" placeholder="Meters"
required value={data.meter} onChange={(e) => setMeter(e.target.value)}></p> :
<p className="card-text fw-bold text-danger"></p>}

          </div>
          <div className="card-
            footer"> {data.quantity
            &&
data.quantity}>meter ? <button className="card-btn" onClick={() =>
addToCart(data.name, data.price)}>Add to Cart</button> : 'not available
length'}

          </div>
        </div>

      </div>
    );
  }
export default Home;

```


Login

```
import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import NavBar from '../components/Navbar';
import Axios from 'axios'; import {
useNavigate } from 'react-router-dom';
function Login() {

    const navigate = useNavigate();

    const [email, setEmail] = useState("");
    const [password, setPassword] =
    useState("");

    var isValid = true;
    const [alertMsg, setAlertMsg] = useState("");

    const makeValidation = () => { if (email === "")
    { isValid = false; setAlertMsg("Email
    Address is required"); }
    else if (password === "") {
        isValid = false;
        setAlertMsg("Password is required");
    } else { isValid =
    true;
    }
    }

    const myStyle={
        backgroundImage:
        "url('https://i.pinimg.com/originals/fd/21/f3/fd21f3be5eb42fe18fae05422f29771
        9.jpg')", height: '100vh', //
        marginTop: '-70px',
        fontSize: '50px',
        backgroundSize: 'cover',
        backgroundRepeat: 'no-
        repeat',
    };

    const doLogin = () => {
        makeValidation();
        if (isValid) {
            Axios.post('http://127.0.0.1:3001/login', {
                email: email,
```

```

        password: password,
      }).then((response) => { if (Object.keys(response.data).length >
        0) { sessionStorage.setItem("ID", response.data[0].id);
        sessionStorage.setItem("NAME", response.data[0].name);
        sessionStorage.setItem("EMAIL", response.data[0].email);
        sessionStorage.setItem("PHONE", response.data[0].phone);
        sessionStorage.setItem("ADDRESS",
response.data[0].address);
        navigate("/");
      } else { showAlertMsg("Invalid Login
        Credentials"); }
    });
  }
}

return (
  <>
    <NavBar />
    <div className=" " style={myStyle} >
      <div className="d-flex justify-content-center" >
        <div className="card border-dark w-25 my-5" >
          <div className="card-header">
            <h5>Login</h5>
          </div>
          <div className="card-body">
            {alertMsg ? <div className="alert alert-danger
alert-msg-cst" role="alert">
              {alertMsg}
            </div> : ''}
            <input type="email" className="form-control mb-3"
placeholder="Email Address" value={email} onChange={(e) =>
setEmail(e.target.value)} />
            <input type="password" className="form-control
mb-
3" placeholder="Password" value={password} onChange={(e) =>
setPassword(e.target.value)} />
            <button type="button" className="btn btn-dark"
onClick={doLogin}>Login</button>
            <Link to="/register" type="button" className="btn
btn-outline-dark ms-2">Register</Link>
            <Link to="/forgotpassword" type="button"
className="btn btn-outline-dark ms-2">Forgot Password</Link>

          </div>
        </div>
      </div>
    </div>
  </>
);

```

```

    }
    export default Login;

```

My Cart

```

import Axios from 'axios'; import React, {
  useEffect, useState } from 'react'; import NavBar
  from '../components/Navbar';
function MyCart() { const [items, setItems] =
  useState([]);

  const [customerName, setCustomerName] =
    useState(sessionStorage.getItem("NAME"));
  const [emailAddress, setEmailAddress] =
    useState(sessionStorage.getItem("EMAIL"));
  const [phoneNumber, setPhoneNumber] =
    useState(sessionStorage.getItem("PHONE"));
  const [address, setAddress] =
    useState(sessionStorage.getItem("ADDRESS"));
  const [successMsg, setSuccessMsg] =
    useState("");

  const [cartList, setCartList] = useState([]);

  const getMyCart = () => {
    Axios.post("http://127.0.0.1:3001/getcart",
      { email: emailAddress,
      })
      .then((response) => {
        console.log(response);
        setCartList(response.data)
      });
  }
  useEffect(() => {
    getMyCart();
  }, [cartList]);

  const makeOrder = (name, price, colour, meter, id ) => {

    Axios.post('http://127.0.0.1:3001/makeorder',
      { name: customerName, email: emailAddress,
        phone: phoneNumber, address: address, id:
        id, product: name, price: price, colour:
        colour, meter: meter, status: "0",

```

```

    }).then((response) => {
      setSuccessMsg("Ordered Successfully") });

    var options={

      key:"rzp_test_w4rbtuG6JGSqcq",
      key_secret:"miwnJloCBnBNaK9FRkpHq6Jn", amount: price *100,
      currency:"INR", name:"Fabriclore", description:name, handler:
      function(response){ alert(response.razorpay_payment_id);
    }, prefill: {
      name:customerName,
      email:emailAddress,
      contact:phoneNumber,
    }, notes:{ address:"Razorpay Corporate
      Office"
    }, theme:{
      color:"#3399cc"
    }

    }; var pay= new
    window.Razorpay(options); pay.open();

  }

```

```

useEffect(() => { const items =
  JSON.parse(localStorage.getItem('MYCART')); if (items) {
    setItems(items);
  }
}, []);

return (
  <>
    <NavBar />
    <div className="container">
      <div class="row g-5 my-4">
        <div className="col-12">
          <h3 className="title-sm text-center">Your Cart</h3>
        </div>
        { successMsg ? <div class="col-12">

```

```

    <div class="alert alert-success alert-msg-cst"
role="alert">
        {successMsg}
    </div> </div> : ''
}
<div className="col-12">
    <table className="table">
        <thead>
            <tr>
                <th>#</th>
                <th>Product Name</th>
                <th>Product Price</th>
                <th>Colour</th>
                <th>Meters</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            { cartList.map((data, key) => {
                return (
                    <tr>
                        <td>{key + 1}</td>
                        <td>{data.product}</td>
                        <td>₹ {data.price}</td>
                        <td>{data.colour}</td>
                        <td>{data.meter}</td>
                        <td>
                            <button className="btn
btn-warning" onClick={() => makeOrder(data.product, data.price, data.colour,
data.meter, data.id )}>Make An Order</button>

                        </td>
                    </tr>
                </tr>
            )};
        })
    }
</tbody>
</table>
</div>
</div>
</div>
</>
);
}
export default MyCart;

```

Product Result

```
import React, { useEffect, useState } from
'react'; import Axios from 'axios'; import {
useParams } from "react-router-dom"; import NavBar
from '../components/Navbar'; import { useNavigate
} from 'react-router-dom';
function ProductResult() { const
navigate = useNavigate();

let { id } = useParams();
const [meter, setMeter] = useState("");

const [productList, setProductList] =
useState([]); const [cartList, setCartList] =
useState([]); const [customerName,
setCustomerName] =
useState(sessionStorage.getItem("NAME"));
const [emailAddress, setEmailAddress] =
useState(sessionStorage.getItem("EMAIL"));
const [phoneNumber, setPhoneNumber] =
useState(sessionStorage.getItem("PHONE")); const [address, setAddress] =
useState(sessionStorage.getItem("ADDRESS")); const [selectColor,
setSelectColor] = useState("Red");

useEffect(() => { localStorage.setItem('MYCART',
JSON.stringify(cartList));
}, [cartList]);

useEffect(() => {
Axios.post("http://127.0.0.1:3001/productresult", {
id: id,
}).then((response) => {
setProductList(response.data);
});
}, [id]);

const addToCart = (name, price) => { if
(sessionStorage.getItem("EMAIL")) { alert("Added Successfully")
setCartList([...cartList, { name: name, price: price, colour:
selectColor }]);

Axios.post('http://127.0.0.1:3001/addtocart',
{ name: customerName, email: emailAddress,
phone: phoneNumber,
colour: selectColor,
meter: meter,
product: name,
```

```

        price: price,
        status: "0",
      }) } else {
        navigate("/login");
      }
    }

    return (
      <>
        <NavBar />
        <div className="container">

          <div className="row g-5 my-4">
            <div className="col-12">
              <h3 className="title-sm text-center">Product
Result</h3>
            </div>
            <div className="col-12 text-end">
              <button type="button" className="color-btn"
onClick={() => setSelectedColor("Red")}>Red</button>
              <button type="button" className="color-btn"
onClick={() => setSelectedColor("Blue")}>Blue</button>
              <button type="button" className="color-btn"
onClick={() => setSelectedColor("Green")}>Green</button>
            </div>
            <div className="col-12">
              <div className="row row-cols-4">
                {
                  productList.map((data, key) => {
                    return (
                      <div className="col">
                        <div className="card h-100
product-card">
                          { selectColor === "Red" ?
                            <img
src={"http://127.0.0.1:3001/images/" + data.image_red} alt="product"
className="img-fluid" /> : ''
                            </div>
                          { selectColor === "Blue" ?
                            <img
src={"http://127.0.0.1:3001/images/" + data.image_blue} alt="product"
className="img-fluid" /> : ''
                            </div>
                          { selectColor === "Green" ?
                            <img
src={"http://127.0.0.1:3001/images/" + data.image_green} alt="product"
className="img-fluid" /> : ''
                            </div>
                        <div className="card-body">

```

```

<h5 className="card-
title">{data.name}</h5>
    <p className="card-text- bold mb-2">₹ {data.price}</p>
        {data.quantity ? <p
className="card-text">Available Quantity: {data.quantity} Meters</p> : <p
className="card-text fw-bold text-danger">Not Available</p>}
        {data.quantity ? <p
className="card-text">Enter Meters: <input type="text" placeholder="Meters"
value={data.meter} onChange={(e) => setMeter(e.target.value)}/></p>: <p
className="card-text fw-bold text-danger"></p>}

    </div>
    <div className="card-
        footer"> {data.quantity
        &&
data.quantity>=meter ? <button className="card-btn" onClick={() =>
addToCart(data.name, data.price)}>Add to Cart</button> : 'not available
length'}

    </div>
</div>

</div>

    );
    })
    }
</div>

</div>
</div>
</div>
</div>
</div>
    );
}
export default ProductResult;

```

Registration

```

import React, { useState } from 'react';
import Axios from 'axios'; import {
useNavigate } from 'react-router-dom'; import
NavBar from '../components/Navbar'; import
validator from 'validator'

function Registration() {

    const navigate = useNavigate();

```



```

const [name, setName] = useState(""); const
[email, setEmail] = useState(""); const
[phone, setPhone] = useState(""); const
[password, setPassword] = useState("");
const [address, setAddress] = useState("");

var isValid = true;
const [alertMsg, setAlertMsg] = useState("");

const myStyle={
  backgroundImage:
"url('https://i.pinimg.com/originals/fd/21/f3/fd21f3be5eb42fe18fae05422f29771
9.jpg')", height: '100vh', //
  marginTop: '-70px',
  fontSize: '50px',
  backgroundSize: 'cover',
  backgroundRepeat: 'no-
repeat',
};

const makeValidation = () =>
{ if (name === "") {
  isValid = false;
  setAlertMsg("Name is required");
}
else if (email === "") {
  isValid = false;
  setAlertMsg("Email Address is required");
}
else if (phone === "" && phone.length !== 10) {
  isValid = false;
  setAlertMsg("Phone Number is required");
}
else if (phone.length !== "10") {
  isValid = false;
  setAlertMsg("Phone Number is not valid");
}
else if (password === "") {
  isValid = false;
  setAlertMsg("Password is required");
}
else if (address === "") {
  isValid = false;
  setAlertMsg("Address is required");
}
else if (!(validator.isEmail(email))) {
  // console.log("123");
  isValid=false;
  setAlertMsg("Enter Valid Email");
}

```

```

        } else {
          isValid = true;
        }
      }

const createAccount = () =>
{ makeValidation(); if
(isValid) {
  Axios.post('http://127.0.0.1:3001/register',
    { name: name, email: email, phone: phone,
      password: password, address: address,
    }).then((response) => {
      navigate("/login");
    });
}

return (
  <>
    <NavBar />
    <div className="" style={myStyle} >
      <div className="d-flex justify-content-center">
        <div className="card border-dark w-50 my-5">
          <div className="card-header">
            <h5>Create New Account</h5>
          </div>
          <div className="card-body">
            {alertMsg ? <div className="alert alert-danger
alert-msg-cst form-control" role="alert">
              {alertMsg}</div> : ''}
            <input type="text" className="form-control mb-3"
placeholder="Full Name" value={name} onChange={(e) => setName(e.target.value)}
/>
            <input type="email" className="form-control mb-3"
placeholder="Email Address" value={email} onChange={(e) =>
setEmail(e.target.value)} />
            <input type="tele" className="form-control mb-3"
placeholder="Phone Number" minlength="10" maxlength="10" pattern="^[0-9-
+\s()]*$" value={phone} onChange={(e) => setPhone(e.target.value)} />
            <input type="password" className="form-control
mb-
3" placeholder="Password" value={password} onChange={(e) =>
setPassword(e.target.value)} />
            <textarea className="form-control mb-4" rows="6"
placeholder="Address" value={address} onChange={(e) =>
setAddress(e.target.value)}></textarea>
            <button type="button" className="btn btn-dark"
onClick={createAccount}>Create</button>
          </div>
        </div>
      </div>
    </div>
  </div>
)

```

```

        </div>
    </>
    );
}
export default Registration;

```

State Result

```

import React, { useEffect, useState } from
'react'; import Axios from 'axios'; import {
useParams } from "react-router-dom"; import NavBar
from '../components/Navbar'; import { useNavigate
} from 'react-router-dom';
function StateResult() { const navigate
= useNavigate();

    let { id } = useParams();
    const [meter, setMeter] = useState("");

    const [productList, setProductList] =
    useState([]); const [cartList, setCartList] =
    useState([]); const [customerName,
    setCustomerName] =
    useState(sessionStorage.getItem("NAME"));
    const [emailAddress, setEmailAddress] =
    useState(sessionStorage.getItem("EMAIL"));
    const [phoneNumber, setPhoneNumber] =
    useState(sessionStorage.getItem("PHONE")); const [address, setAddress] =
        useState(sessionStorage.getItem("ADDRESS")); const [selectColor,
        setSelectColor] = useState("Red");

    useEffect(() => { localStorage.setItem('MYCART',
        JSON.stringify(cartList));
    }, [cartList]);

    useEffect(() => {
        Axios.post("http://127.0.0.1:3001/stateresult", {
            id: id,
        }).then((response) => {
            setProductList(response.data);
        });
    }, [id]);

    const addToCart = (name, price) => { if
        (sessionStorage.getItem("EMAIL")) { alert("Added Successfully")
        setCartList([...cartList, { name: name, price: price, colour:
        selectColor }]);
    }

```

```

        Axios.post('http://127.0.0.1:3001/addtocart',
          { name: customerName, email: emailAddress,
            phone: phoneNumber, colour: selectColor,
            meter: meter, product: name, price: price,
            status: "0",
          }) } else {
          navigate("/login");
        }
      }

    return (
      <>
        <NavBar />
        <div className="container">
          <div className="row g-5 my-4">
            <div className="col-12">
              <h3 className="title-sm text-center">State Result</h3>
            </div>
            <div className="col-12 text-end">
              <button type="button" className="color-btn"
onClick={() => setSelectColor("Red")}>Red</button>
              <button type="button" className="color-btn"
onClick={() => setSelectColor("Blue")}>Blue</button>
              <button type="button" className="color-btn"
onClick={() => setSelectColor("Green")}>Green</button>
            </div>
            <div className="col-12">
              <div className="row row-cols-4">
                {
                  productList.map((data, key) => {
                    return (
                      <div className="col">
                        <div className="card h-100
product-card">


63



Cauvery college gonikoppal Dept. of BCA


```

```

        <h5 className="card-
title">{data.name}</h5>
        <p className="card-text- bold mb-2">₹ {data.price}</p>
        {data.quantity ? <p
className="card-text">Available Length: {data.quantity} Meters</p> : <p
className="card-text fw-bold text-danger">Not Available</p>}
        {data.quantity ? <p
className="card-text">Enter Meters: <input type="text" placeholder="Meters"
value={data.meter} onChange={(e) => setMeter(e.target.value)}></p> : <p
className="card-text fw-bold text-danger"></p>}

        </div>
        <div className="card-footer">
        {data.quantity &&
data.quantity>=meter ? <button className="card-btn" onClick={() =>
addToCart(data.name, data.price)}>Add to Cart</button> : 'not available
length'}

        </div>
    </div>

    </div>

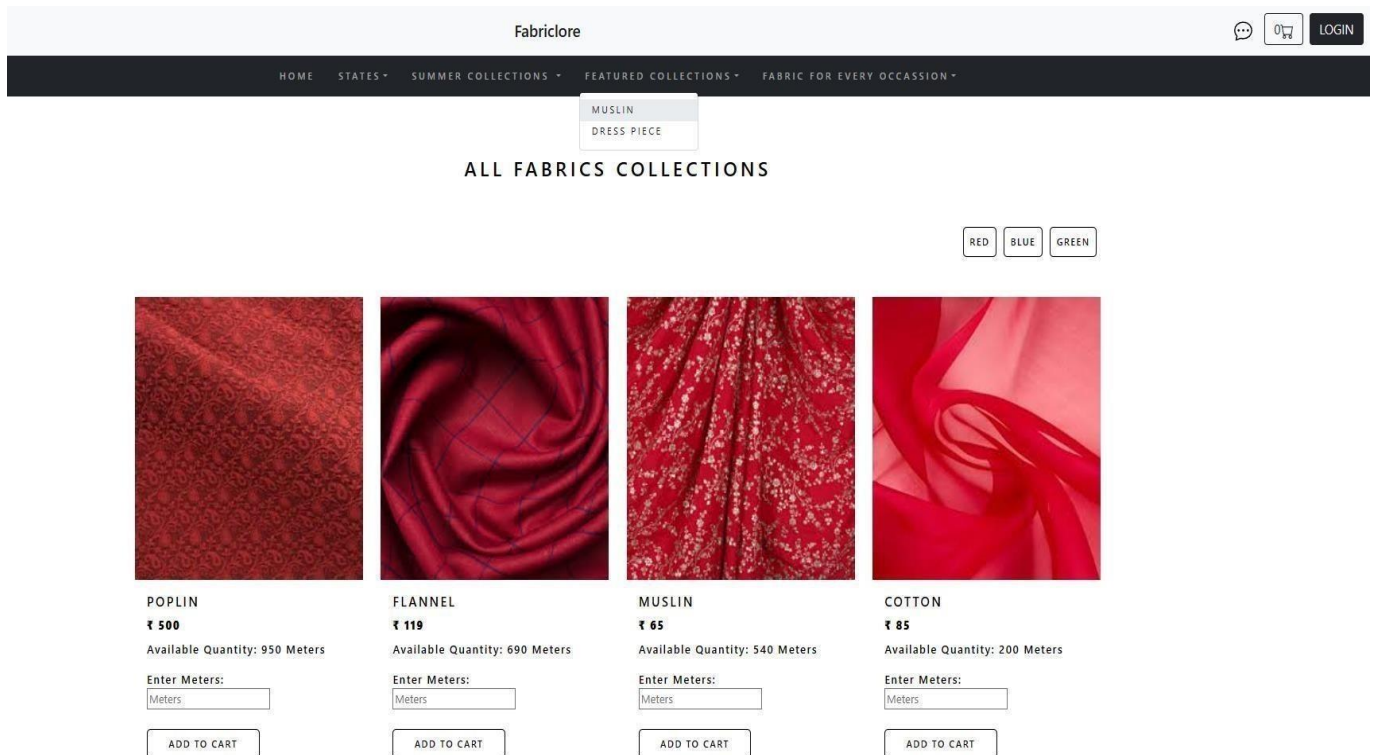
    );
    })
  </div>
</div>

</div>
</div>
</>
);
} export default
StateResult;

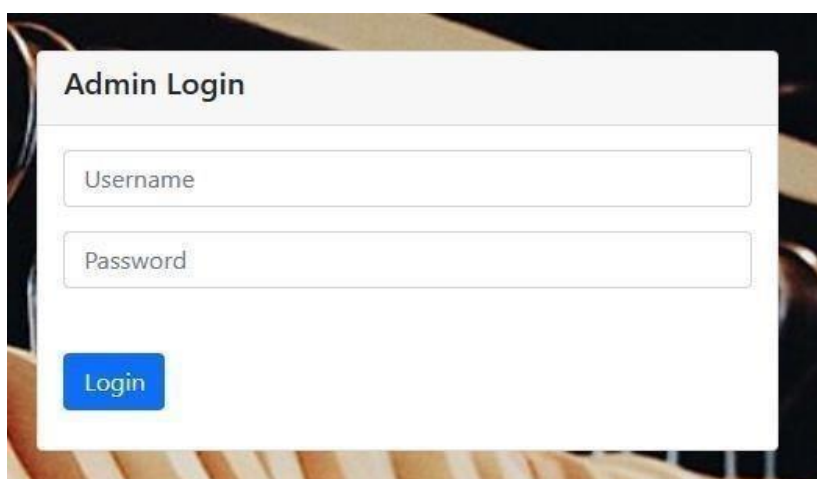
```

6. User Interface

6.1 Index Page



6.2 Admin Login



6.3 Add Product

Add Products

Select Category

Select Category

Select State

Select State

Product Name

Product Quantity (In Lengths)

Product Price

Upload Red Image

Choose File No file chosen

Upload Blue Image

Choose File No file chosen

Upload Green Image

Choose File No file chosen

Save

| # | ProductName | Length | Product Price | Action | |
|---|---|--------|---------------|------------|--------|
| 1 | chiffon | 400 | 100 | Deactivate | Delete |
| 2 | cotton | 200 | 85 | Deactivate | Delete |
| 3 | Dress piece | 500 | 40 | Deactivate | Delete |
| 4 | Flannel | 690 | 119 | Deactivate | Delete |
| 5 | frock | 215 | 50 | Deactivate | Delete |
| 6 | LEHERIYA PRINT COTTON FABRIC | 0 | 599 | Deactivate | Delete |
| 7 | Muslin | 540 | 65 | Deactivate | Delete |
| 8 | poplin | 950 | 500 | Deactivate | Delete |
| 9 | STRIPES LEHERIYA SCREEN PRINT COTTON FABRIC | 11 | 186 | Deactivate | Delete |

6.4 Add Staff

Add Staff

Full Name

Email Address

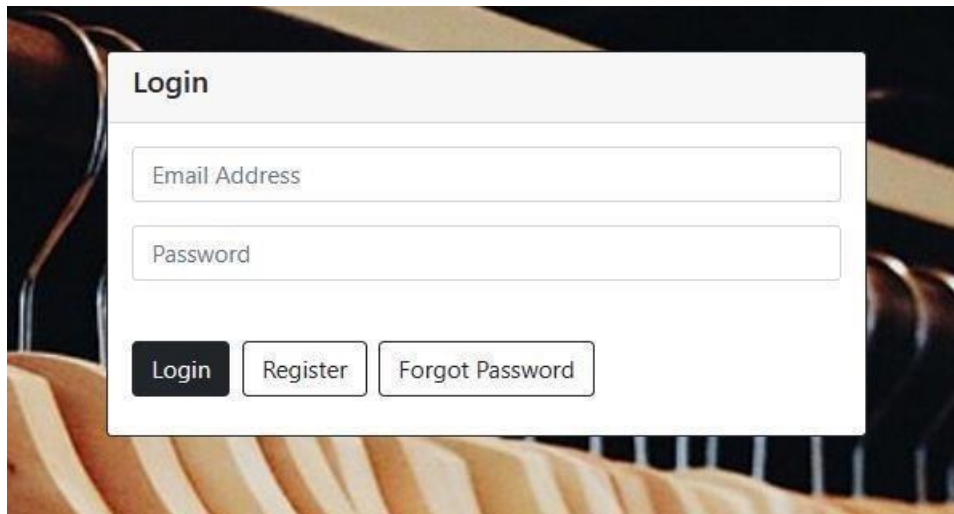
Phone Number

Password

Create

| # | Name | Email Address | Phone Number | Password | Action |
|---|-------|-----------------|--------------|----------|--------|
| 1 | Rudra | rudra@gmail.com | 9867543888 | rudra | Delete |

6.5 Customer Login



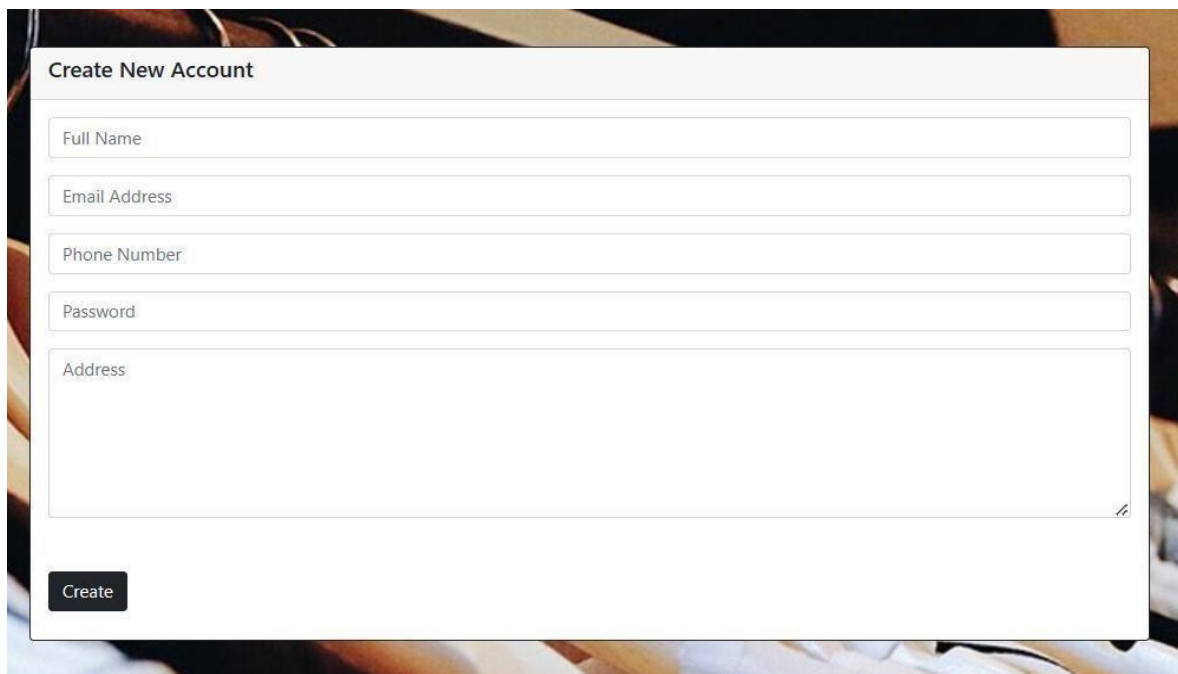
Login

Email Address

Password

Login Register Forgot Password

6.6 Customer Register



Create New Account

Full Name

Email Address

Phone Number

Password

Address

Create

6.7 Add Product to cart



FLANNEL

₹ 119

Available Quantity: 690 Meters

Enter Meters:

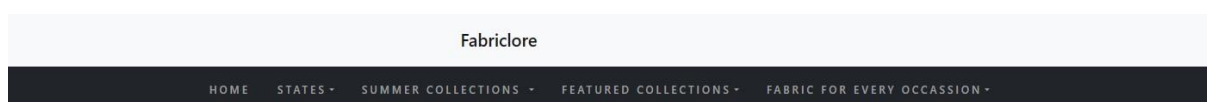
ADD TO CART

localhost:3000 says

Added Successfully

OK

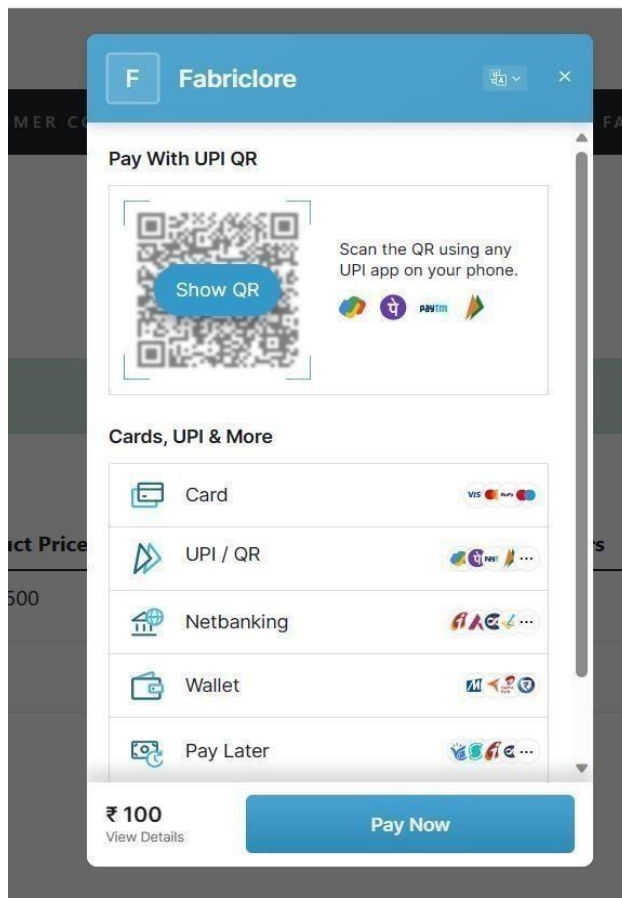
6.8 Cart



YOUR CART

| # | Product Name | Product Price | Colour | Meters | Action |
|---|--------------|---------------|--------|--------|-------------------------------|
| 1 | Flannel | ₹ 4760 | Blue | 40 | Make An Order |
| 2 | poplin | ₹ 0 | Red | 0 | Make An Order |

6.9 Place an Order and payment



6.10 Manage Order

Fabriclore Admin

OrdersStaffsFeedbacksProductsLogout

RECENT ORDERS

| # | Customer Name | Email Address | Phone Number | Address | Product Name | Price | Colour | Meters | Action | Delete |
|---|----------------|-------------------------|--------------|-------------------|--------------|-------|--------|--------|-----------|--------|
| 1 | Ajay Aiyappa | ajayaiyappa@gmail.com | 9876598765 | Madikeri | Flannel | 4760 | Blue | 40 | Complete | Delete |
| 2 | Neema Muthanna | neemamuthanna@gmail.com | 9876543219 | jky vill and post | poplin | 500 | Red | 1 | Complete | Delete |
| 3 | Neema Muttanna | neemamuttanna@gmail.com | 9876543210 | madikeri | frock | 750 | Green | 15 | Completed | Delete |
| 4 | Neema Muttanna | neemamuttanna@gmail.com | 9876543210 | madikeri | Muslin | 650 | Red | 10 | Completed | Delete |
| 5 | Neema Muttanna | neemamuttanna@gmail.com | 9876543210 | madikeri | poplin | 25000 | Red | 50 | Completed | Delete |
| 6 | seema | seema@gmail.com | 8796859493 | seems like | Flannel | 1190 | Red | 10 | Completed | Delete |
| 7 | Neema Muttanna | neemamuttanna@gmail.com | 9876543210 | madikeri | frock | 3500 | Green | 70 | Completed | Delete |

View Feedback

Fabriclore Admin

[Orders](#) [Staffs](#) [Feedbacks](#) [Products](#)[Logout](#)

FEEDBACK

| # | Customer Email | Product Name | Message | Date |
|---|-----------------------|--------------|--|-----------|
| 1 | ajayaiyappa@gmail.com | poplin | Hi Fabriclore I'm Ajay Aiyappa, I received your product and I'm really happy with it. Thank you!!! | 11/7/2023 |
| 2 | seema@gmail.com | Flannel | Your product was soooo good! I really loved it | 29/2/2023 |

7. Conclusion

7.1 Limitations

- **Limited Fabric Inspection:** Shopping for fabric online lacks the physical touch and inspection of the material. This absence makes it challenging to accurately evaluate factors like texture, weight, and quality. Moreover, variations in screen displays might cause colours to appear different online than in reality, leading to discrepancies between the visual representation and the actual product.
- **Shipping Costs and Delays:** Shipping expenses, particularly for heavy or bulky fabrics, can accumulate quickly. Furthermore, shipping delays or mishaps may disrupt your project's timeline, causing inconvenience.
- **Inaccurate Colour Representation:** Variations in screen settings can result in different colour appearances for fabrics online. This discrepancy can lead to dissatisfaction when the received fabric doesn't align with the perceived colour from the online image.
- **Inconsistent Sizing:** Fabric measurements might lack the precision you require. Variability in cutting and measuring can result in slightly different dimensions than what you originally ordered, potentially affecting your project's outcome.

7.2 Scope for enhancement (Future scope)

- Introduce a feature enabling customers to personalize fabrics according to their preferred colour, pattern, and design choices.
- Incorporate a virtual fabric preview tool that grants customers the ability to visualize how fabrics will appear in various applications like clothing and upholstery.
- Deliver tailored recommendations based on customers' browsing and purchase histories, enhancing their shopping experience.
- Provide comprehensive insights into the sourcing and production methods of your fabrics, underscoring your dedication to sustainability and ethical practices.
- Develop and curate educational content, including video tutorials, blog articles, and instructional guides covering sewing, crafting, and optimizing the use of different fabric types.

- Arrange online workshops or classes in collaboration with experts to empower customers in enhancing their sewing and crafting skills.

7.3 Conclusion

To conclude, the emergence of online fabric stores has significantly reshaped the fabric shopping landscape. These platforms have ushered in a new era of convenience, diversity, and accessibility in the realm of fabric acquisition. The conventional fabric shopping journey has been redefined, allowing customers to seamlessly explore an extensive array of fabric choices from the comfort of their homes. The added advantages of price comparison, customer reviews, and well-informed decision-making further enhance the appeal of online fabric shopping. Beyond merely expanding options for crafters, designers, and DIY enthusiasts, these online stores have also granted smaller fabric vendors a gateway to a global clientele. As technology continues its advancement, the online fabric store industry is poised to undergo further transformation, potentially delivering more personalized shopping encounters and innovative interactions with fabrics.

7.4 References

7.4.1 Books Referred

"The Digital Fabric Revolution: The Innovators' Guide to the Future of Textiles"

7.4.2 Websites

<https://www.fabric.com/>

<http://www.w3schools.com/css/js/default.asp>