

FESTIVE CRACKERS BOOKING

A MINI PROJECT REPORT

Submitted by

RUBITHA D

(REG.NO:24MCR083)

SIVA K

(REG.NO:24MCR101)

THIRISURYA B

(REG.NO:24MCR117)

*in partial fulfillment of the requirements
for the award of the degree
of*

**MASTER OF COMPUTER APPLICATIONS
IN
DEPARTMENT OF COMPUTER APPLICATIONS**



KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 052

DECEMBER 2024

**DEPARTMENT OF COMPUTER APPLICATIONS
KONGU ENGINEERING COLLEGE**

**(Autonomous)
PERUNDURAI ERODE-638052**

DECEMBER 2024

BONAFIED CERTIFICATE

This is to certify that the Project report entitled **FESTIVE CRACKERS BOOKING** is the bonafied record of project work done by **D. RUBITHA (24MCR083)** and **K. SIVA (24MCR101)** and **B. THIRISURYA (24MCR117)** in partial fulfilment of the requirements for the award of the Degree of Master of computer application in of Anna university, Chennai during the year 2024-2026.

SUPERVISOR

HEAD OF THE DEPARTMENT

(Signature with seal)

Date:

Submitted for the end semester viva-voce examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I affirm that the Project Report entitled “**FESTIVE CRACKERS BOOKING**” being submitted in partial fulfilment of the requirements for the award of Master of Engineering is the original work carried out by me. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Date:

D.RUBITHA
(24MCR083)

K. SIVA
(24MCR101)

B. THIRISURYA
(24MCR117)

I certify that the declaration made above by the candidates is true to the best of my knowledge.

Date:

Name and Signature of the Supervisor

ABSTRACT

This project is entitled as “**FESTIVE CRACKERS BOOKING**” is a web application for the crackers booking provides a digital platform to streamline various aspects of crackers production. It offers an organized repository for managing module descriptions, facilitating efficient data storage, retrieval, and content management. The front-end is built using React JS combined with HTML and CSS to create a user-friendly and visually appealing interface. JavaScript is used for the purpose of Front-end validation, it ensures that user inputs are checked before they are sent to the server, improving data accuracy and reducing unnecessary server requests.

The back-end is powered by Node.js, while Firebase handles authentication and real-time database management. We’ve integrated Firebase for both authentication and real-time database management. This allows users to create personalized accounts, securely store their information, and enjoy real-time updates when browsing products. The system is built to handle all the essential e-commerce functionalities smoothly, like displaying products, providing detailed information, and processing orders.

This application offers a user-friendly interface with authentication and authorization features, ensuring data security. Overall, the Festive Crackers Store is built with the user in mind, offering a reliable and responsive experience whether they’re on desktop or mobile. The application is built to handle high volumes of user traffic and provide a smooth, interactive shopping experience. We aim to make the festive shopping experience easy, secure, and enjoyable for everyone.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved correspondent **THIRU.A.K.ILANGO, B.Com.,M.B.A., LL.B.** and our Principal **Dr.V.BALUSAMY B.E(Hons)., M.Tech, PhD.** Kongu Engineering College, Perundurai for providing us with the facilities offered

We convey our gratitude and heartfelt thanks to our Head of the Department **Dr.A.TAMILARASI MSc., MPhil., PhD.**, Department of Computer Applications, Kongu Engineering College, for her perfect guidance and support that made this work to be completed successfully.

We also wish to express my gratitude and sincere thanks to our project coordinator **MRS. S. HEMALATHA MCA.**, Associate Professor, Department of Computer Applications, Kongu engineering College, who have motivated us in all aspects for completing the project in the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide **Mr. S. B. KARTIKEYAN M.C.A.**, Department of Computer Applications, Kongo Engineering College for giving her valuable guidance and suggestions which helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping overwhelm in all proceedings. We bow our heart with heartfelt thanks to all those who thought us their warm services to succeed and achieve our work.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	iv
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
	LIST OF SYMBOLS	x
1	INTRODUCTION	1
2	SYSTEM ANALYSIS	3
	2.1 EXISTING SYSTEM	3
	2.2 PROPOSED SYSTEM	3
	2.2.1 Advantages of Proposed system	4
	2.3 FEASIBILITY STUDY	4
	2.3.1 Economic Feasibility	4
	2.3.2 Technical Feasibility	6
	2.3.3 Operational Feasibility	6
3	SYSTEM REQUIREMENTS	7
	3.1 HARDWARE REQUIREMENTS	7
	3.2 SOFTWARE REQUIREMENTS	7
	3.2.1 Front End	8
	3.2.2 Script Language	9
	3.2.3 Back End	9
4	SYSTEM DESIGN	12
	4.1 MODULE DESCRIPTION	12
	4.1.1 Registration Information	12
	4.1.2 Product Details	12

	4.1.3 Order Details	12
	4.1.4 Administration Module	13
	4.1.5 Login Module	13
	4.2 USECASE DIAGRAM	14
	4.2 SYSTEM FLOW DIAGRAM	15
	4.4 ER DIAGRAM	17
	4.5 DATA FLOW DIAGRAM	18
	4.6 DATABASE DESIGN	19
	4.6.1 Data Security	19
	4.6.2 Data Validation	20
	4.6.3 Data Protection	20
	4.6.4 Scale with multiple database	20
	4.7 INPUT DESIGN	21
	4.8 OUTPUT DESIGN	22
5	SYSTEM TESTING	23
	5.1 UNIT TESTING	23
	5.2 INTEGRATION TESTING	24
	5.3 VALIDATION TESTING	25
	5.4 WHITE BOX TESTING	26
6	SYSTEM IMPLEMENTATION	27
7	CONCLUSION & FUTURE ENHANCEMENTS	29
	7.1 CONCLUSION	29
	7.2 FUTURE ENHANCEMENTS	29
	APPENDIX 1- SAMPLE CODING	30
	APPENDIX 2- SAMPLE SCREENSHOTS	46
	REFERENCES	49

LIST OF FIGURES

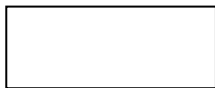
FIGURE No.	FIGURE NAME	PAGE No.
4.1	Use case Diagram	14
4.2	User System Flow Diagram	15
4.3	Admin System Flow Diagram	16
4.4	ER Diagram	17
4.5	DFD for user Activities	18
A 2.1	Home Page	42
A 2.2	Account Page	42
A 2.3	Product Page	43
A 2.4	Order Page	43
A 2.5	Order Confirm Page	44
A 2.6	Cart Page	54

LIST OF ABBREVIATIONS

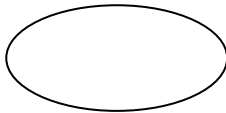
ABBREVIATION	EXPANSION
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
JSON	JavaScript Object Notation
JS	Java Script

LIST OF SYMBOLS

1. USECASE DIAGRAM SYMBOLS

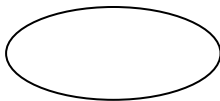


1. **Actor** – Actors are external entities that interact with the system and these can include users, other systems, or hardware devices

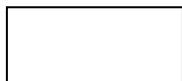


2. **Use Case**- Use cases are like scenes in the play. They represent specific things your system can do. In the online shopping system

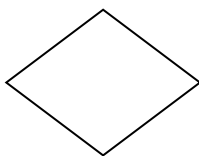
2. SYSTEM FLOW DIAGRAM SYMBOLS



1. **Terminator**- A system flowchart begins and ends with an oval symbol

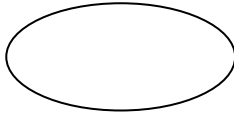


2. **Process**- A rectangle then represents a process or an activity

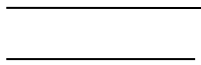


3. **Decision making**- One of the major symbols used in system flowcharts is the decision boxes represented by a diamond shape. They represent the decisions made, and all arrows coming out of these decision boxes show alternate pathways for data

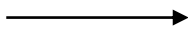
3. DATA FLOW-DIAGRAM SYMBOLS



1. **Process-** It is represented by a circle and depicts how the data is handled and processed in the system



2. **Data Store-** It is represented by two parallel lines and depicts a location where data is stored in the system



3. **Data Flow-** It is represented by directional lines and depicts the flow of data from one location to another.

CHAPTER 1

INTRODUCTION

The “**Festive Crackers Booking**” is a web-based application using React JS as front-end development and NODEJS, FIREBASE as backend development tools and to manage the database. The main goal of the project is to deliver the product to reach their customer smartly and to gradually increase the number of customers effectively. Further, features may be included to generate and maintain the stock details automatically while purchasing. Shop with confidence using our secure and easy checkout process. Your personal information is protected, and our payment options make transactions hassle-free. Securing applications built with Node.js and Firebase involves implementing best practices for both the server-side (Node.js) and the cloud-based backend (Firebase).

Now, the project is developed like online shopping and it also maintains the customer details and purchase details, etc. It includes some tasks such as customer details, product details, product price, purchase details, etc. It is very hard to store the history of the data. It's cost-effective and expensive. This application contains a general organization profile and purchase details. Each new stock is added and entitled with the name and price and maintained. Enable notifications to receive updates on new arrivals, promotions, and order status.

The customer will receive a notification through WhatsApp to ensure order confirmation and the status of delivery is sent as a notification to the customer. There is no backup issue because report generation is available. Be the first to know about sales events and restocks, so you never miss out on your favorite items. Stay in the loop with real-time order tracking. Receive updates on the status of your purchases from confirmation to delivery, ensuring you are always in control.

Here, the login page is created to protect the stock from the threat and misuse of the product. This product includes the direction of the shop to buy it offline when necessary. It also includes the customers can choose the product and add it to a cart so they can buy it when they want.

They can also use cash on delivery to buy a product. It also involved the shop number, details, and the products that are sold in a shop and also there is an option where we can communicate with the owner of the shop. Briefly, first, the home page which describes the description of the shop navigation bar is also there for navigating from one page to another. The filter option is one of the effective options that is used to sort the clothes by type. Finally, users can buy it.

OBJECTIVE OF PROJECT

- Digital transformation
- User-friendly interface
- Efficient and faster accessibility

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The existing system was carried out manually and through a web application, the customers can make orders. The existing system description shows how to increase the number of customers smartly. The product is available orderly in exclusive shops and select retailers for a short while. It is not user-friendly and interactive. The accurate result could not be found because of the manual calculations and efficient access. So, there is a need for a proposed system to rectify the drawbacks of manual calculation and to increase the number of customers gradually.

2.1.1 DRAWBACKS OF EXISTING SYSTEM

The disadvantages of the existing system are:

- Based on the customers requirement delivery will be provided
- Quality assurance
- Fake offers and discounts

2.2 PROPOSED SYSTEM

The proposed system helps to quickly access the application and efficiently order the crackers. Our project involves developing a robust system for efficient data management and reporting, featuring dynamic report generation in both Excel and JSON formats. There is no existing system available and this was one of the main drawbacks. By this project, the current drawback is rectified. It is useful for all kinds of users.

2.2.1 Advantages of Proposed System

The proposed system has the following advantages:

- Reduces human effort compared to other conventional methods.
- Accuracy and error-free transaction process.
- Reports are easily prepared and efficient.
- User Data is maintained by the database and reduces data loss.
- Help the user to reduce the workload and mental conflict.
- Long-distance people can also buy crackers easily.

2.3 FEASIBILITY STUDY

A feasibility study is a crucial step in the early stages of planning and decision-making for various types of projects, ventures, or initiatives. Its primary purpose is to assess whether a proposed project or idea is viable, practical, and financially sustainable. Although feasibility studies can help project managers determine the risk and return of pursuing a plan of action, several steps should be considered before moving forward. Feasibility studies help stakeholders make informed decisions by analyzing various aspects of the project, including technical, economic, legal, operational, scheduling, and environmental factors.

2.3.1 Economic Feasibility

Economic feasibility is an essential aspect of evaluating the viability of a project, investment, or business initiative. It involves assessing whether the expected benefits and returns from the endeavor outweigh the associated costs and risks. Economic feasibility involves various situations such as cost-benefit analysis, payback period, risk assessment, market research, sensitivity analysis, and discounted cash flow (DCF) analysis. The user can be given responses to asking questions, justification of any capital outlay is that it will reduce expenditure or improve the quality of service or goods, which in turn may be expected to provide increased profits.

The project includes free sources and is built efficiently. So it is cost-beneficial and it does not affect the manual process of purchasing clothes.

1. Cost-Benefit Analysis: Economic feasibility involves analyzing the total costs associated with a project, including initial investment, operational expenses, and potential risks.

2. Return on Investment (ROI): It assesses the expected returns from the project, such as revenue, profits, or other benefits. ROI is a key metric used to determine whether the investment is financially worthwhile.

3. Payback Period: This indicates the time it takes for the project to recoup its initial investment through cash flows or earnings. A shorter payback period is generally more favorable.

4. Risk Assessment: Evaluating the risks associated with the project, including market risks, financial risks, and external factors, and considering how these might impact the economic feasibility.

5. Market Research: Understanding the market demand and competition to ensure that there is a sufficient market for the product or service.

6. Sensitivity Analysis: Testing the project's feasibility under different scenarios, such as best-case, worst-case, and most likely situations, to account for uncertainties.

7. Discounted Cash Flow (DCF) Analysis: Assessing the present value of future cash flows to determine whether the project will generate a positive net present value (NPV).

Ultimately, economic feasibility helps stakeholders make informed decisions about whether to proceed with a project, investment, or business idea based on its potential for financial success. It's a critical step in the decision-making process to ensure that resources are allocated effectively

2.3.2 Technical feasibility

Technical feasibility is an assessment of whether a proposed project, product, or initiative can be successfully implemented from a technological standpoint. It involves evaluating whether the necessary technology, skills, and resources are available to carry out the project effectively. Here are some key points to consider when assessing technical feasibility such as available technology, skill and expertise, resource availability, compatibility, risk assessment, cost and budget, regulatory and legal compliance, and alternate solutions. A positive technical feasibility assessment indicates that the project can be executed with the available technology, resources, and skills. A negative assessment suggests that technical challenges may impede successful implementation, requiring reevaluation or potentially abandoning the project.

2.3.3 Operational feasibility

Operational feasibility is an assessment of whether a proposed project or initiative can be effectively and efficiently integrated into an organization's existing operations. It focuses on evaluating how well the project aligns with the practical aspects of an organization, such as its people, processes, and systems. There are some key points such as process integration, resource availability, user acceptance, training requirements, change management, scalability, return on investment (ROI), maintenance, and support.

A positive operational feasibility assessment indicates that the project can be smoothly integrated into the organization's operations, with minimal disruption and maximum effectiveness. A negative assessment suggests that operational challenges, such as resistance from stakeholders or the unavailability of necessary resources, may impede successful implementation, necessitating further evaluation or potential adjustments to the project plan.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

Processor	Intel Core i3
RAM	4 GB
Hard Disk	528 GB
Keyboard	Standard102 keys

3.2 SOFTWARE SPECIFICATIONS

Operating System	Windows 10 & above
Environment	Chrome, Visual Studio Code
Frontend	React JS, HTML, CSS
Backend	NodeJS, FIREBASE

3.2.1 FRONT END

REACTJS:

ReactJS, often referred to simply as React, is an open-source JavaScript library developed and maintained by Facebook. It is primarily used for building user interfaces (UI) or user interface components for single-page applications (SPAs) where UI updates are frequent. React allows developers to create reusable UI components and efficiently manage the state of an application. React components have lifecycle methods that developers can tap into to perform actions at different stages of a component's existence, such as mounting, updating, and unmounting. React follows a unidirectional data flow, meaning data changes flow in a single direction from parent components to child components. This helps maintain a predictable state and makes it easier to debug. Here are some key concepts and features of React such as Declarative Syntax, Component-Based Architecture, Virtual DOM, JSX (JavaScript XML), Unidirectional Data Flow, State and Props, and lifecycle methods.

HTML & CSS

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are essential tools for building websites, each with its strengths and limitations. HTML is easy to learn, making it beginner-friendly, and since it's supported by all browsers, it's widely accessible. It's also free to use and helps with SEO through semantic tags, which make content easier for search engines to understand. However, HTML is mostly limited to organizing content and cannot handle design or interactivity on its own. It can also get repetitive, especially in larger projects, and without additional technologies like CSS or JavaScript, web pages remain static and basic.

In short, HTML provides the structure, and CSS handles the appearance, working together to create modern, functional, and visually appealing websites. However, both require careful management, especially in larger projects, to avoid issues and ensure the final product is both effective and easy to maintain.

3.2.2 SCRIPT LANGUAGE

JAVASCRIPT:

JavaScript is a versatile, high-level programming language widely used for creating dynamic and interactive web content. It is one of the core technologies of the web, alongside HTML and CSS, and primarily runs on the client side within web browsers. JavaScript allows developers to build interactive user interfaces, validate forms, update content dynamically without reloading the page, and add animations to enhance user experience.

It is event-driven, lightweight, and features dynamic typing, making it highly flexible for various applications. Beyond front-end development, JavaScript can also be used on the back end with frameworks like Node.js, enabling full-stack development. Its cross-platform compatibility, vast ecosystem of libraries and frameworks, and ease of learning make it a popular choice among developers. However, care must be taken to address potential security issues, as JavaScript code runs directly in the browser.

3.2.3. BACK END

NODEJS:

Node.js (Node) is an open-source, cross-platform runtime environment for executing JavaScript code. Node is used extensively for server-side programming, making it possible for developers to use JavaScript for client-side and server-side code without needing to learn an additional language. Node is sometimes referred to as a programming language or software development framework, but neither is true; it is strictly a JavaScript runtime. Node incorporates the V8 JavaScript engine, the same one used in Google Chrome and other browsers. A Node application runs in a single process. Node does not create a new thread for every request, as is often the case with traditional server-side programs. In this way, a Node server can handle thousands of concurrent connections without having to contend with thread concurrency issues or the overhead multithreading brings. Node.js is event-driven and runs asynchronously. Code written for the Node environment does not follow the traditional model of receive process, send, wait, and receive found in other systems.

Instead, Node implements an event loop that processes incoming requests as they stack up in the event queue, handling small requests one after the other without waiting for responses.

ADVANTAGES OF NODEJS

- Single Language, Node.js allows developers to use JavaScript for both server-side and client-side scripting, promoting a unified language across the entire stack.
- Asynchronous and Non-blocking I/O, Node.js is designed to handle asynchronous operations efficiently. It uses an event-driven, non-blocking I/O model, making it suitable for handling a large number of concurrent connections.
- Fast Execution, Node.js is built on the V8 JavaScript engine, which compiles JavaScript directly into machine code. This results in high performance and fast execution of code.

FIREBASE:

Firebase is a Cloud-hosted, NoSQL database that uses a document model. It can be horizontally scaled while letting you store and synchronize data in real time among users. This is great for applications that are used across multiple devices such as mobile applications. Firebase is optimized for offline use with strong user-based security that allows for server-less apps as well. Firebase is built on the Google infrastructure and is built to scale automatically. The Firebase APIs are packaged into a single SDK that can be expanded to multiple platforms and languages. This includes C++ and Unity, which are both popular for mobile development.

Here's a deeper look at Firebase Database such as Real-time Database, JSON Data Structure, Real-time Synchronization, Offline Support, Security Rules, Authentication Integration, Scalability, SDKs for Various Platforms, Serverless Architecture, Analytics and Monitoring. The flexibility of the NoSQL model allows developers to adapt the data structure to the specific needs of their application.

The choice between a Realtime Database and a Cloud Fire store often depends on the application's requirements and the desired features of the database.

ADVANTAGES OF FIREBASE

- **Hosting**, Firebase Hosting allows developers to deploy web applications quickly, with features like automatic SSL certificate provisioning and global content delivery through a Content Delivery Network (CDN).
- **Cloud Storage**, Firebase Storage offers scalable and secure cloud storage for user-generated content, such as images, videos, and files.
- **Authentication and Security**, Firebase provides secure and scalable authentication, and security rules can be defined to control access to data in the Realtime Database and Cloud Fire store

CHAPTER 4

SYSTEM DESIGN

4.1 MODULE DESCRIPTION

The project contains the following modules such as:

- Homepage
- Product
- Add to cart
- Orders
- Profile

4.1.1 Registration Form

This module is designed to collect information about the customers. Customers are an important asset for any company. The details such as customer name, contact number, and email ID are collected. A table is designed to store information in a database.

4.1.2 Product Details

This module collects information about the clothes. The information like cloth product name, product description, quantity, and price of the product is maintained.

4.1.3 Order Details

The order details module is designed in such a way as to collect all information about the products being ordered.

4.1.4 Administrator Module

The Administrator is provided with the username and password with which he/she can access the system. The Administrator has the right to maintain and modify the data in the database such as details, user details, etc.

4.1.5 Login Module

A login module is provided for both the administrator and the customer to access the system. The administrator has more rights than the customer in maintaining the system information.

4.2 USE CASE DIAGRAM

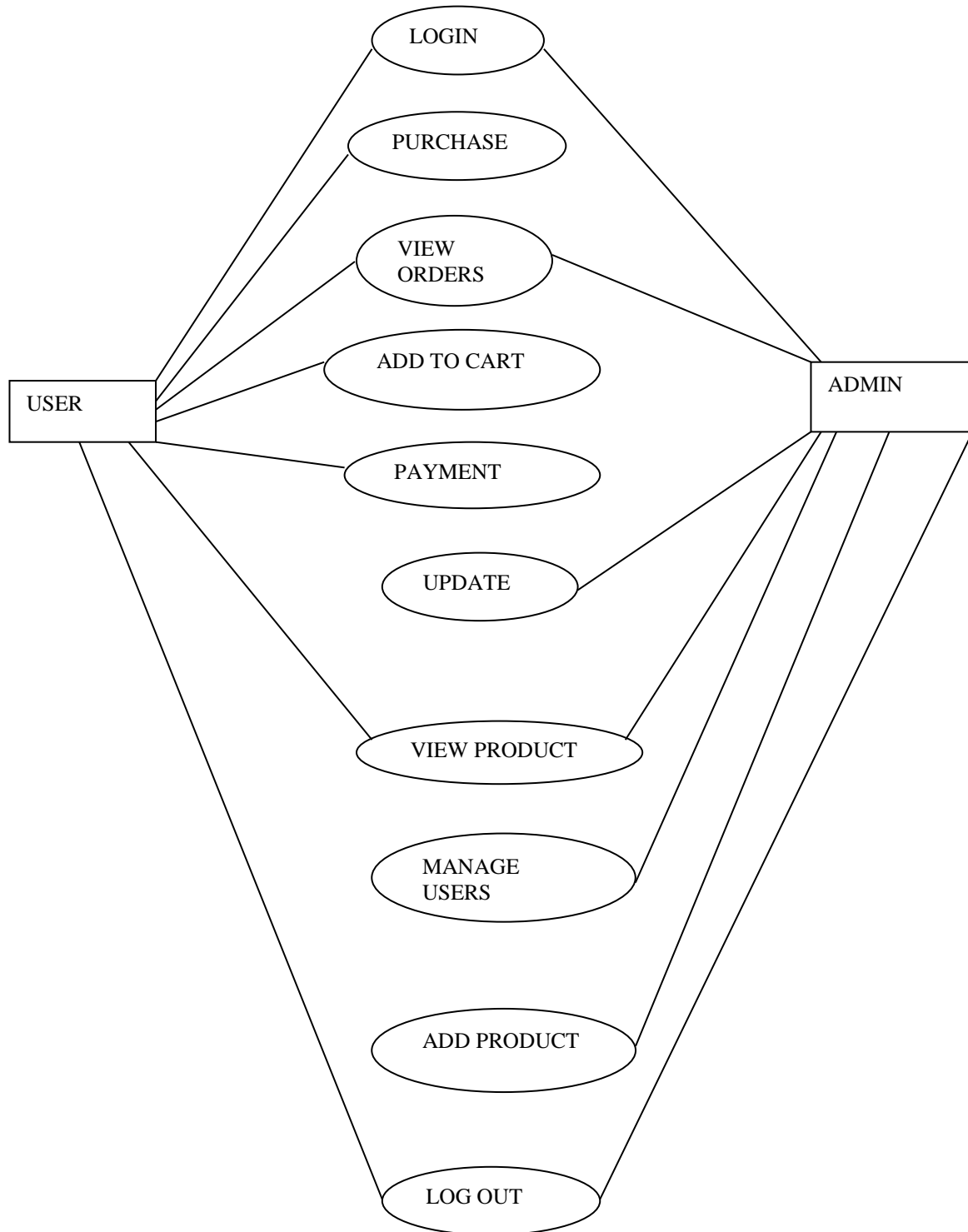


Figure 4.1 Use case Diagram

4.3 SYSTEM FLOW DIAGRAM

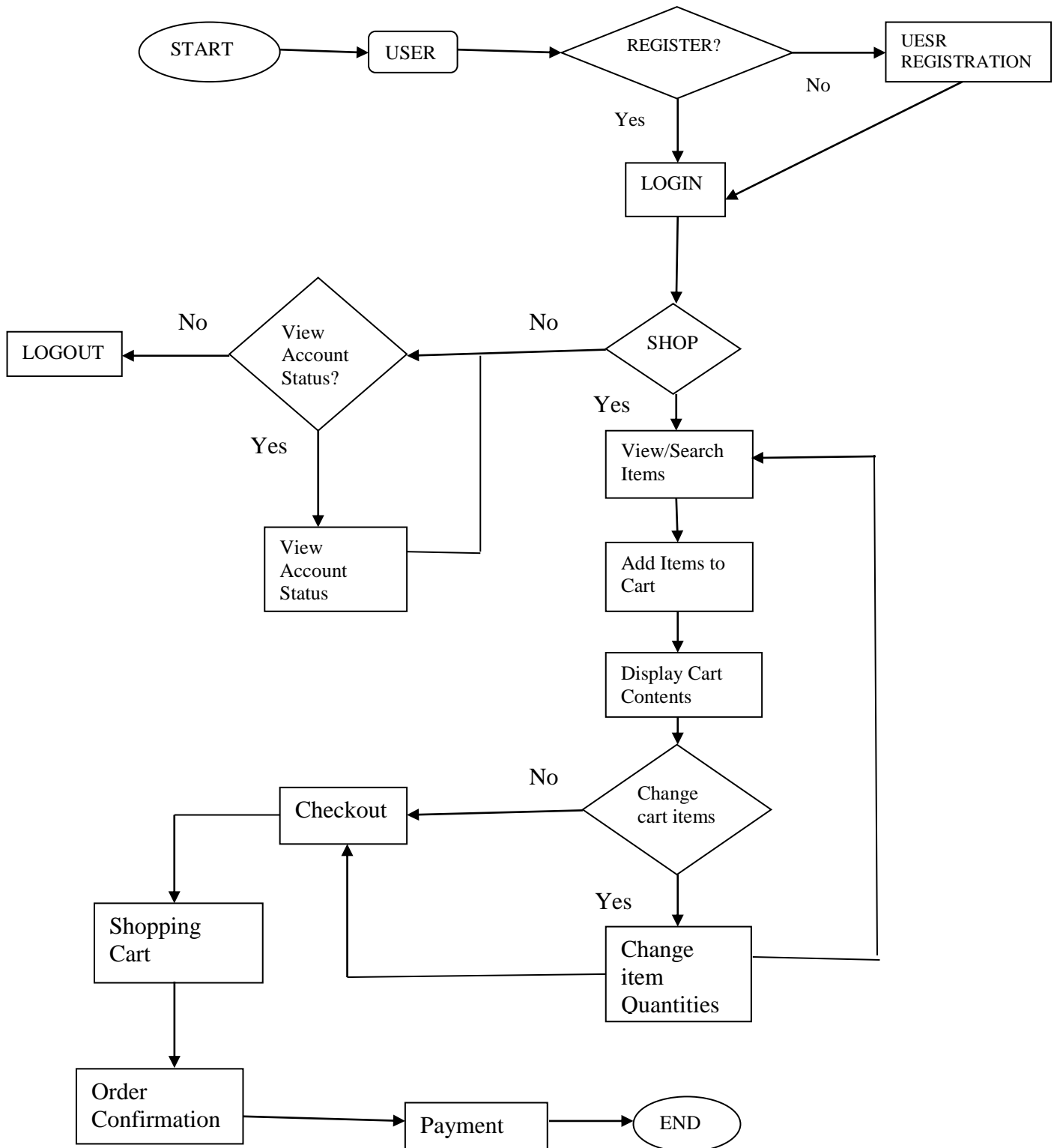


Figure 4.2 User System Flow Diagram

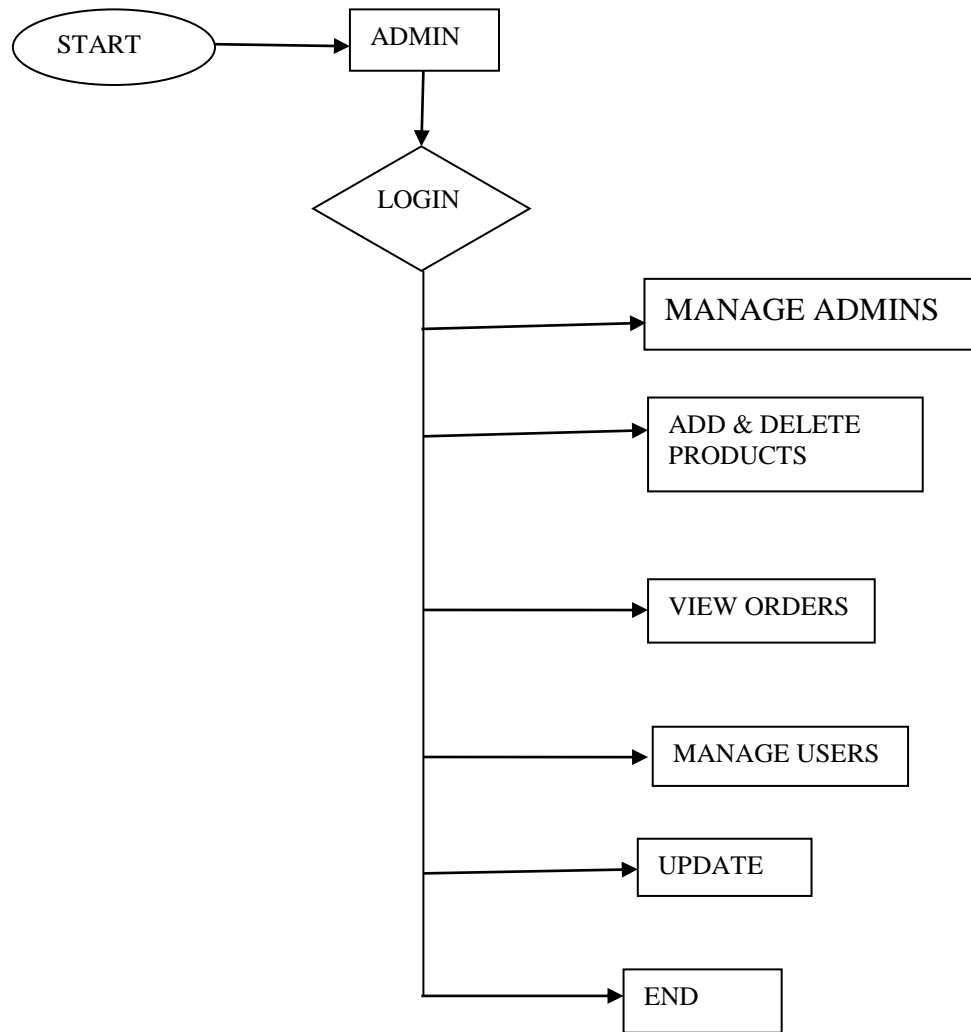


Figure 4.3 Admin System Flow Diagram

4.4 ER DIAGRAM

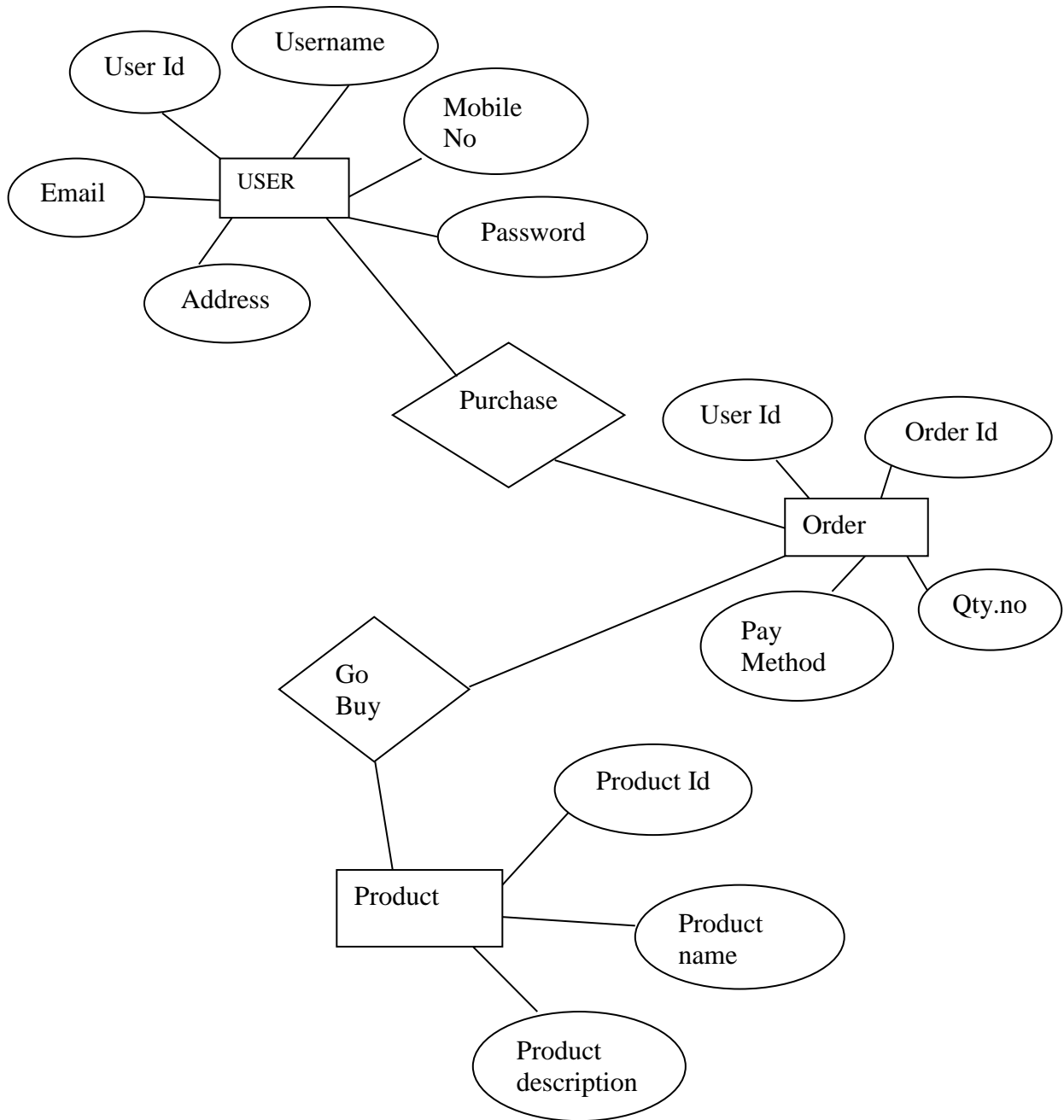


Figure 4.4 ER Diagram

4.5 DATA FLOW DIAGRAM

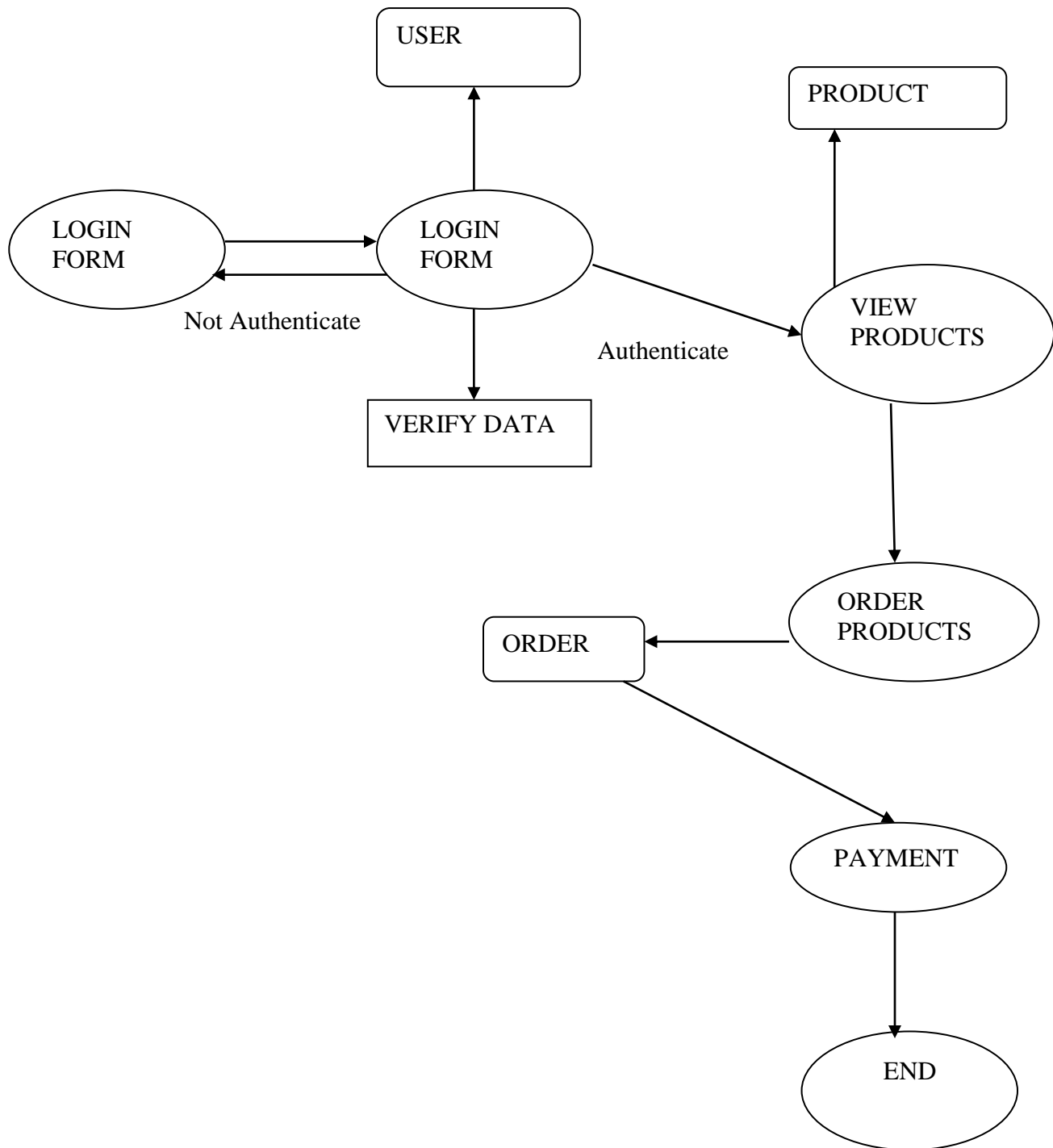


Figure 4.5 DFD for User Activities

4.6 DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information effectively and efficiently. All Firebase Realtime Database data is stored as JSON objects. You can think of the database as a cloud-hosted JSON tree. Unlike a SQL database, there are no tables or records. When you add data to the JSON tree, it becomes a node in the existing JSON structure with an associated key.

Avoid nesting data, because the Firebase Realtime Database allows nesting data up to 32 levels deep, you might be tempted to think that this should be the default structure. However, when you fetch data at a location in your database, you also retrieve all of its child nodes. In addition, when you grant someone read or write access at a node in your database, you also grant them access to all data under that node. Therefore, in practice, it's best to keep your data structure as flat as possible. With this nested design, iterating through the data becomes problematic. Flatten Data structure, To avoid nested data, try to flatten our data structure. Splitting the data into a separate path would be a better way. So, it can efficiently download separate calls as it's needed.

4.6.1 Data Security

The Firebase Realtime Database provides a full set of tools for managing the security of your app. These tools make it easy to authenticate your users, enforce user permissions, and validate inputs. Firebase-powered apps run more client-side code than those with many other technology stacks. Firebase services encrypt data in transit using HTTPS and logically isolate customer data. Firebase Security Rules work by matching a pattern against database paths and then applying custom conditions to allow access to data at those paths. All Rules across Firebase products have a path-matching component and a conditional statement allowing read or write access.

4.6.2 Data Validation

The Firebase Realtime Database is schema-less. This makes it easy to change things as you develop, but once your app is ready to distribute, data needs to stay consistent. The rules language includes a `.validate` rule which allows you to apply validation logic using the same expressions used for `.read` and `.write` rules. The only difference is that validation rules do not cascade, so all relevant validation rules must be evaluated to be true for the write to be allowed.

4.6.3 Data Protection

Firebase customers typically act as the "data controller" (GDPR) or "business" (CCPA/CPRA) for any personal data or information about their end-users they provide to Google in connection with their use of Firebase, and Google generally operates as a "data processor" (GDPR) or "service provider" (CCPA/CPRA). Google Analytics for Firebase uses cookies with unique identifiers, which are personal data under the GDPR. It also processes other personal data, including invasive identifiers for mobile devices. All these data are processed on Google's infrastructure because Google owns Firebase.

4.6.4 Scale with multiple database

The best way to optimize performance and scale your data in Firebase Realtime Database is to split your data across multiple Realtime Database instances, also known as database sharding. Sharding gives you the flexibility to scale beyond the limits that apply to individual database instances, in addition to load balancing and performance optimization.

When to share your data, to share your data, follow these steps:

1. Map your data to multiple databases according to your app's specific needs.
2. Create multiple database instances.
3. Configure your app so it connects to the Realtime Database instance necessary for each data set.

4.7 INPUT DESIGN

Input design is the process of converting user-originated inputs to a computer-understandable format. Input design is one of the most expensive phases of the operation of a computerized system and is often the major problem of a system. A large number of problems with a system can usually be tracked back to fault input design and method. Every moment of input design should be analyzed and designed with utmost construction.

The design of the input should be made the input as the over to the numerous networks in the reliable area that should be passed as the installation in the remote network. It has the following constraints in the input database.

- It is suitable for more available data clearance and is made available.
- The menu of designs should be understandable and it is in the right format.

The system takes input from the users, processes it, and produces an output. Input design is a link that ties the information system into the world of its users. The system should be user-friendly to gain appropriate information for the user. The decisions made during the input design are the project gives the low time consumption to make the sensitive application simple. When applying the project, it provides low manpower attrition with reasonable output.

The amount of funds that the company can spend on the research and development of the system is limited. The expenditures must be justified. Thus the developed system is well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased. Input data of a system may not necessarily be raw data captured in the system from scratch.

These can also be the output of another system or subsystem. The design of input covers all the phases of input from the creation of initial data to the actual entering of the data into the system for processing. The design of inputs involves identifying the data needed, specifying the characteristics of each data item, capturing and preparing data from computer processing, and ensuring the correctness of data. Input design is the process of converting user-originated inputs to a computer-based format. In the project, the forms are designed with easy-to-use options such as selecting the master records through a dropdown list in transaction forms. The coding is being done such that proper validations are made to get the perfect input. No error inputs are accepted. The end users need not give the ID themselves.

4.8 OUTPUT DESIGN

Output design generally refers to the results and information that are generated by the system for many end-users; it should be understandable with the enhanced format. The Output of the software is used to make the remote installation of the new software in the system and it awakens the immediate alert to the system that should be enhanced as the input to the system. Output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application.

Computer output is the most important direct source of information to the user output design deals with form design. Efficient output design should improve the interfacing with the user. The term output applies to any information produced by an information system in terms of display. When analysts design system output, they Identify the specific output that is needed to meet the requirements of the end user. Previewing the output reports by the user is extremely important because the user is the ultimate judge of the quality of the output and, in turn, the success of the system

CHAPTER 5

SYSTEM TESTING

After the supply code has been completed, documented as associated statistics structures. Completed the assignment has to go through trying out and validation where in there's subtitle and exact try to get mistakes.

The assignment developer treads lightly, designing and executing a look to demonstrate that this system works in preference to uncovering mistakes, lamentably mistakes might be a gift and if the assignment developer doesn't discover them, the person will discover them.

The assignment developer is constantly answerable for trying out the character devices i.e. modules of this system. In many instances developer additionally conducts integration trying out i.e. the trying out step that ends in the development of the whole application structure.

5.1 UNIT TESTING

We must inspect the programmers that make up the device while checking out the unit. As a result, Unit check-out is occasionally referred to as Program check-out. The software program devices in a device are the modules and workouts that may be combined and incorporated to perform a certain function, with the Unit checking out the modules separately to discover errors first. This allows for the discovery of code and common sense errors that would otherwise be missed if the module were used alone. The checking out was done at some point during the programming level.

TESTCASE 1

Module	:	Admin Login
Test Type	:	Loading of the appropriate form for the administrator
Input	:	Username and Password
Expected Output	:	Display admin products

Sample Test

Output	:	Redirect to the Main Page and display the admin products
Analysis	:	In this form, the username and password have been tested in the correct format. If the username and password are mismatched with data in the database, here a data mismatch error can occur.

5.2 INTEGRATION TESTING

Integration testing is done to test if the individual modules work together as one single unit. In integration testing, the individual modules that are to be integrated are available for testing. Thus, the manual test data that was used to test the interfaces was replaced by that which is generated automatically from the various modules. It can be used for testing how the modules would interact with the proposed system. The modules are integrated and tested to reveal the problem interfaces.

TESTCASE 2

Module	:	Admin products
Test Type	:	Order Management and Product Management
Input	:	Navigation between Admin options Expected
Output	:	Navigation between modules is completed

Sample Test

Input	:	By login page, add Product items.
Output	:	Respective products open correctly and display the type of product.
Analysis	:	Each category of the product will be open.

5.3 VALIDATION TESTING

Verification and validation checking out are critical tests, which might be achieved earlier than the product has been surpassed over to the customer. This makes sure that the software program checking out lifestyles cycle begins and evolves early. Each verification and validation intends to make certain that the product is made in step with the necessities of the customer and does certainly fulfill the supposed purpose.

TESTCASE 1

Module	:	Register
Test Type	:	Register new user
Input	:	Input to all fields Expected
Output	:	No Required field should not be empty

Sample Test

Input	:	Input for a required field is not provided
Output	:	Provide all the required fields
Analysis	:	The expected output is same, so the form passed the validation test

5.4 WHITE BOX TESTING

In white box testing knowing the internal workings of the product, tests can be conducted to ensure that internal operations are performed according to specifications and all internal components have been adequately exercised. In white-box testing, logical paths through the software are tested by providing test cases that exercise specific sets of conditions and loops.

Using white-box testing software developers can derive test cases that are

- Guarantee that all independent paths within a module have been exercised at least once
- Exercised all logical decisions on their true and false side
- Exercise all loops at their boundaries and within their operational boundExercise internal data structure to ensure their validity

CHAPTER 6

SYSTEM IMPLEMENTATION

When the preliminary layout was performed for the machine, the purchaser became consulted for the popularity of the layout so that similarly court cases of the machine improvement may be carried on. After the improvement of the machine, an illustration was given to them approximately the operation of the machine. The intention of the machine example became to discover any malfunction of the machine.

After the control of the machine became authorized the machine was carried out inside the concern, to start with the machine ran parallel with the current guide machine. The machine has been examined with stay records and has proved to be unfastened and consumer-friendly.

Implementation is the method of changing a brand new or revised machine layout into an operational one whilst the preliminary layout is performed through the machine, an illustration is given to the stop consumer approximately the operating machine.

This method is used to confirm and discover any logical mess operating the machine by feeding diverse combos or taking a look at records. After the approval of the machine through each stop consumer control of the machine became carried out. System implementation is made from many sports. The six essential sports are as follows.

CODING

Coding is the manner in wherein the bodily layout specs are created means the evaluation group becomes running PC code via the means of the programming group.

TESTING

As each software module can be tested, the coding procedure begins and advances in parallel.

INSTALLATION

The process of replacing an existing device with a new device is known as installation. This entails converting existing data, software, documentation, and painting techniques to those compatible with the new device.

DOCUMENTATION

Individual publications provide statistics on how to utilize the machine and its flow as a result of the setup procedure.

TRAINING AND SUPPORT

A training plan is a method for teaching consumers how to utilize a new system fast and the education approach was most likely upgraded before the start of the project.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION

The project entitled “**FESTIVE CRACKERS BOOKING**” was completed successfully. It is designed to provide a website that would make searching, viewing, and selecting a product easier. The customer can then view the complete specifications of each product. The system has been developed with much care and free of errors and at the same time it is efficient and less time-consuming. The entire system is secured. It can be implemented in any nearby shops or branded shops selling various kinds of products with simple modifications. Providing moderators more control over products so that each moderator can maintain their products. Online customers must have access to the Internet and a valid method of payment to complete the transaction.

7.2 FUTURE ENHANCEMENTS

Enhancement to be made, modifying or redeveloping the code to support change in the specification. It is necessary to keep up with the changing user requirements and operational environment.

Following are some of the future that can be considered as a future enhancement.

1. The system can be extended to allow the users to create accounts and save products into a wish list.
2. Updating the information online.
3. Support for event management.

APPENDIX 1

SAMPLE CODING

Home.html

```
import React from 'react';
import { Carousel } from 'react-bootstrap';
import WiresFrontPage from './Frontpage/WiresFrontPage';
import PipesFrontPage from './Frontpage/PipesFrontPage';
import FittingsFrontPage from './Frontpage/FittingsFrontPage';
import HitachiFrontPage from './Frontpage/HitachiDrillFrontPage';
import BoschDrillFrontPage from './Frontpage/BoschDrillFrontPage';
import BeltsFrontPage from './Frontpage/BeltsFrontPage';
import ChainsFrontPage from './Frontpage/ChainsFrontPage';
const Home = ({ fitcomponent }) => {
  const GoToWires = () => {
    fitcomponent("Products", "Wires");
  };
  return (
    <div>
      <section id="home">
        <div class="container">
          <h2> <span>Welcome to FCB</span> <br /> Your Festive Crackers Booking
Store</h2>
          <p>Get your product delivered to your doorstep.</p>
          <button onClick={GoToWires}>Explore Now</button>
        </div>
      </section>
    </div>
  );
}
```

```

</section>
<section id="Featured" className="my-5 pb-2">
  <div className="container text-center pt-5">
    <h3>Featured DayTime crackers</h3>
    <hr className="mx-auto" />
    <h4 className='pb-2'>Best Selling Confetti Crackers</h4>
    <WiresFrontPage />
    <h4>Best Selling Colorful Smoke Bombs</h4>
    <hr className="mx-auto" />
    <PipesFrontPage />
    <h4>Best Selling Parachute Rockets</h4>
    <hr className="mx-auto" />
    <FittingsFrontPage />
  </div>
  <div className="container text-center py-2">
    <h3>Featured Combination Boxes</h3>
    <hr className="mx-auto" />
    <p>Best Selling Family Packs</p>
    <HitachiFrontPage />
  </div>
  <div className="container text-center py-3">
    <h3>Featured Fancy Showpieces </h3>
    <hr className="mx-auto" />
    <p>Best Selling Sky Show Packs</p>
    <BoschDrillFrontPage />
  </div>
  <div className="container text-center py-3">
    <h3>Featured Kids's Special Crackers </h3>
    <hr className="mx-auto" />
    <p>Best Selling Pop-pops</p>
    <BeltsFrontPage />
  </div>

```

```

</div>
<div className="container text-center py-3">
  <h3>Featured Sound Crackers</h3>
  <hr className="mx-auto" />
  <p>Best Selling Double/Triple Sound Bomb</p>
  <ChainsFrontPage />
</div>
</section>
<section id="about" className="my-5 py-5" >
  <div className="container">
    <div className="row">
      <div className="col-lg-6">
        <Carousel>
          <Carousel.Item>
            
          </Carousel.Item>
          <Carousel.Item>
            
          </Carousel.Item>

```

```

</Carousel>
</div>
<div className="col-lg-6 d-flex align-items-center">
  <div>
    <h2 className="mb-4">About Us</h2>
    <p className="text-muted">
      FCB is your one-stop destination for all cracker needs. With a wide range of
      products and top-notch customer service, we strive to make your shopping experience
      seamless and convenient.
    </p>
    <p className="text-muted">
      Our mission is to provide high-quality products at competitive prices, delivered
      right to your doorstep. Whether you're a professional contractor or a DIY enthusiast, we have
      everything you need to get the job done.
    </p>
    <p className="text-muted">
      Shop with confidence at FCB and join our growing community of satisfied
      customers.
    </p>
    { /* <h3 className="mt-4">Why Choose Us?</h3>
    <ul className="text-muted">
      <li>Wide selection of products</li>
      <li>Competitive prices</li>
      <li>Top-notch customer service</li>
      <li>Convenient online shopping experience</li>
      <li>Fast and reliable delivery</li>
      <li>Customer satisfaction guarantee</li>
    </ul> */ }
  </div>
</div>
</div>

```

```

</div>
</section>
<div>
  {/* <section id="Category" className="my-5 pb-5">
    <div className="container text-center mt-5">
      <h3>Shop By Categories</h3>
      <hr className="mx-auto" />
      <div className="row mt-5">
        <div className="one col-lg-3 col-md-6 col-12">
          <div className="image-container position-relative">
            
            <div className="bottom mt-2">
              <h2>Wires & Cables</h2>
            </div>
          </div>
        </div>
        <div className="one col-lg-3 col-md-6 col-12">
          <div className="image-container position-relative">
            
            <div className="bottom mt-2">
              <h2>Pipes</h2>
            </div>
          </div>
        </div>
        <div className="one col-lg-3 col-md-6 col-12">
          <div className="image-container position-relative">
            
            <div className="bottom mt-2">

```

```

        <h2>PVC Fittings</h2>
      </div>
    </div>
  </div>
<div className="one col-lg-3 col-md-6 col-12">
  <div className="image-container position-relative">
    
    <div className="bottom mt-2">
      <h2>Lightings</h2>
    </div>
  </div>
</div>
<div className="row mt-5">
  <div className="one col-lg-3 col-md-6 col-12">
    <div className="image-container position-relative">
      
      <div className="bottom mt-2">
        <h2>Drill Machine</h2>
      </div>
    </div>
  </div>
  <div className="one col-lg-3 col-md-6 col-12">
    <div className="image-container position-relative">
      
      <div className="bottom mt-2">
        <h2>Drill Machine</h2>
      </div>
    </div>
  </div>
</div>

```

```

    </div>
  </div>
  <div className="one col-lg-3 col-md-6 col-12">
    <div className="image-container position-relative">
      
    <div className="bottom mt-2">
      <h2>Heat Gun And Air Bowlers</h2>
    </div>
  </div>
</div>
<div className="one col-lg-3 col-md-6 col-12">
  <div className="image-container position-relative">
    
    <div className="bottom mt-2">
      <h2>Belts</h2>
    </div>
  </div>
</div>
<div className="one col-lg-3 col-md-6 col-12">
  <div className="image-container position-relative">
    
    <div className="bottom mt-2">
      <h2>Chains</h2>
    </div>
  </div>
</div>
<div className="one col-lg-3 col-md-6 col-12">
  <div className="image-container position-relative">

```

```

        
        <div className="bottom mt-2">
            <h2>Bearings</h2>
        </div>
    </div>
</div>
</div>
</div>
</div>
</div>
</section> */}
</div>
</div>
)
}
export default Home

```


Navbar.html

```

import React, { useEffect, useState } from 'react'
import Swal from 'sweetalert2';
import { auth } from '../firebase';

const Navbar = ({ componentrender, component, userData }) => {
  const [loginstate, setloginstate] = useState(false)
  const [username, setUsername] = useState("")
  const logout = async () => {
    Swal.fire({
      html: `
        <div className="p-5">
          <div className="spinner-border text-dark" role="status">
            <span className="visually-hidden">Loading...</span>
          </div>
        </div>
      `,
      showConfirmButton: false,
      background: 'transparent',
    });
    try {
      await auth.signOut();
      Swal.fire({
        icon: 'success',
        title: 'Logout Successful',
        showConfirmButton: true,
        confirmButtonColor: '#212529',
      });
    } catch (error) {
    }
  }
}

```

```

    }
    useEffect(() => {
      if (userData) {
        setloginstate(true)
        setUsername(userData.profile.firstname);
      }
      else {
        setloginstate(false)
      }
    }, [userData]);
    const scrollToAboutSection = () => {
      const aboutSection = document.getElementById('about');
      if (aboutSection) {
        aboutSection.scrollIntoView({ behavior: 'smooth' });
      }
    };
    const [navtoggleicon, setNavToggleIcon] = useState(false);
    return (
      <div>
        <nav class="navbar navbar-expand-lg bg-white shadow fixed-top d-lg-none d-block">
          <div class="container " >
            
            <a className='text-dark navbar-toggler border-0' data-bs-toggle="collapse" aria-
expanded="false" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
              {navtoggleicon ? (
                <i className='bi bi-x-lg'></i>
              ) : (

```

```

        <i className='bi bi-justify'></i>
      )}
    </a>
    <div class="collapse navbar-collapse" id="navbar">

      <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
          <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Home" ? "text-coral" : ""}`} onClick={() => componentrender("Home")}>Home</a>
        </li>
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
          <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Products" ? "text-coral" : ""}`} onClick={() => componentrender("Products")}
>Products</a>
        </li>
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
          <a className={`navigation p-2 fw-regular mx-2 ${component ===
"About" ? "text-coral" : ""}`} onClick={() => CompositionEvent(scrollToAboutSection)}
>About</a>
        </li>
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>

```

```

        <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Contact" ? "text-coral" : ""}`} onClick={() => componentrender("Contact")} >Contact
Us</a>
    </li>

    {!loginstate && (
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
            <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Login" ? "text-coral" : ""}`} onClick={() => componentrender("Login")} >Login</a>
        </li>
    )}

    {!loginstate && (
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
            <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Register" ? "text-coral" : ""}`} onClick={() => componentrender("Register")} >Register</a>
        </li>
    )}

    {loginstate && (
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
            <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Account" ? "text-coral" : ""}`} onClick={() => componentrender("Account")} >Account</a>
        </li>
    )}

    {loginstate && (
        <li class="nav-item my-3 ps-2 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar" >

```

```

        <a className="navigation fw-regular mx-2 d-flex">Hi
{username}</a>
    </li>
  )}
  {loginstate && (
    <li class="nav-item my-3 pt-1 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
      <a className={`navigation fw-regular mx-2 d-flex ${component ===
"Cart" ? "text-coral" : ""}`} onClick={() => componentrender("Cart")}><i class="fa fa-
shopping-cart ps-1 pb-1" aria-hidden="true"></i><span className='d-lg-none d-
block'>Cart</span></a>
    </li>
  )}
  {loginstate && (
    <li class="nav-item my-3 pt-1 my-lg-0" aria-expanded="false" data-bs-
toggle="collapse" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
      <a className="navigation fw-regular mx-2 d-flex" onClick={() =>
logout()}><i class="fa fa-sign-out" aria-hidden="true"></i><span className='d-lg-none d-
block ps-1 pb-1'>Logout</span></a>
    </li>
  )}
</ul>
</div>
</div>
</nav>

<nav class="navbar navbar-expand-lg bg-white shadow fixed-top d-lg-block d-none">
  <div class="container" >

```

```



<a className='text-dark navbar-toggler border-0' data-bs-toggle="collapse" aria-
expanded="false" data-bs-target="#navbar" aria-controls="navbar" onClick={() =>
setNavToggleIcon(!navtoggleicon)}>
    {navtoggleicon ? (
        <i className='bi bi-x-lg'></i>
    ) : (
        <i className='bi bi-list'></i>
    )}
</a>

<div class="collapse navbar-collapse" id="navbar">
    <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar">
            <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Home" ? "text-coral" : ""}`} onClick={() => componentrender("Home")}>Home</a>
        </li>
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar">
            <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Products" ? "text-coral" : ""}`} onClick={() => componentrender("Products")}
>Products</a>
        </li>
        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar">
            <a className={`navigation p-2 fw-regular mx-2 ${component ===
"About" ? "text-coral" : ""}`} onClick={scrollToAboutSection} >About</a>
        </li>

```

```

        <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar">
            <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Contact" ? "text-coral" : ""}`} onClick={() => componentrender("Contact")} >Contact
Us</a>

        </li>
        {!loginstate && (
            <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar">
                <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Login" ? "text-coral" : ""}`} onClick={() => componentrender("Login")} >Login</a>
            </li>
        )}
        {!loginstate && (
            <li class="nav-item my-3 my-lg-0 " aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar">
                <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Register" ? "text-coral" : ""}`} onClick={() => componentrender("Register")} >Register</a>
            </li>
        )}

        {loginstate && (
            <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar" >
                <a className={`navigation p-2 fw-regular mx-2 ${component ===
"Account" ? "text-coral" : ""}`} onClick={() => componentrender("Account")} >Account</a>
            </li>
        )}
        {loginstate && (
            <li class="nav-item my-3 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar" >

```

```

        <a className="navigation fw-regular mx-2 d-flex">Hi
{username}</a>
    </li>
  )}
  {loginstate && (
    <li class="nav-item my-3 pt-1 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar">
      <a className={`navigation fw-regular mx-2 d-flex ${component ===
"Cart" ? "text-coral" : ""}`} onClick={() => componentrender("Cart")}><i class="fa fa-
shopping-cart ps-1 pb-1" aria-hidden="true"></i><span className='d-lg-none d-
block'>Cart</span></a>
    </li>
  )}
  {loginstate && (
    <li class="nav-item my-3 pt-1 my-lg-0" aria-expanded="false" data-bs-
target="#navbar" aria-controls="navbar" >
      <a className="navigation fw-regular mx-2 d-flex" onClick={() =>
logout()}><i class="fa fa-sign-out" aria-hidden="true"></i><span className='d-lg-none d-
block ps-1 pb-1'>Logout</span></a>
    </li>
  )}
</ul>
</div>
</div>
</nav>
</div>
  )}
export default Navbar

```


APPENDIX-2

SCREENSHOTS

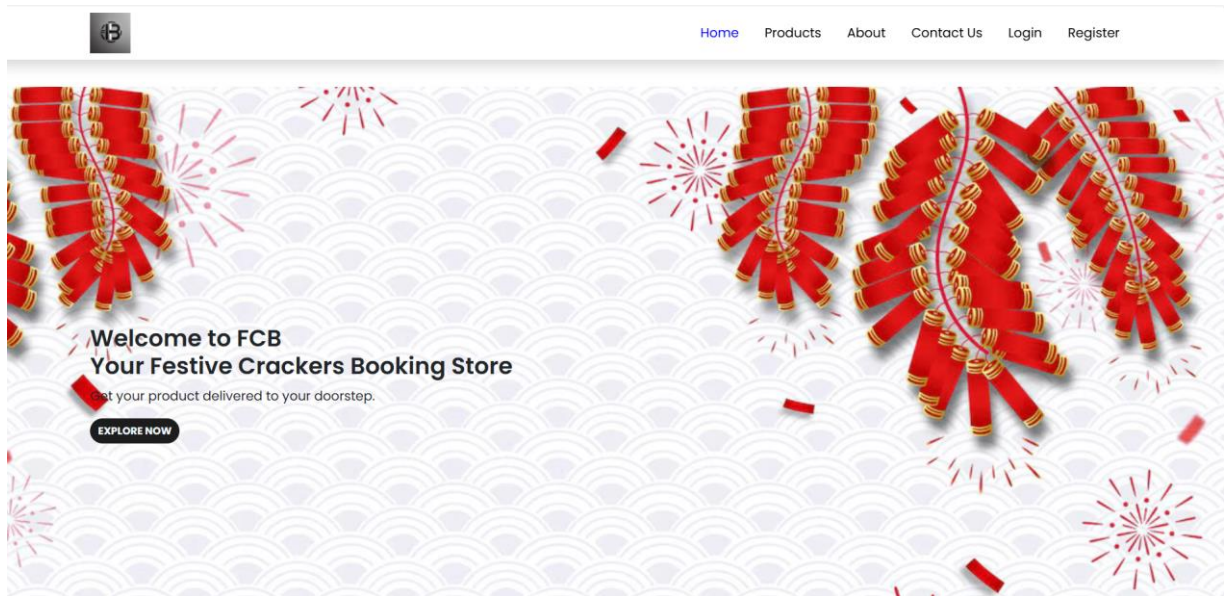


Fig A 2.1 Home Page

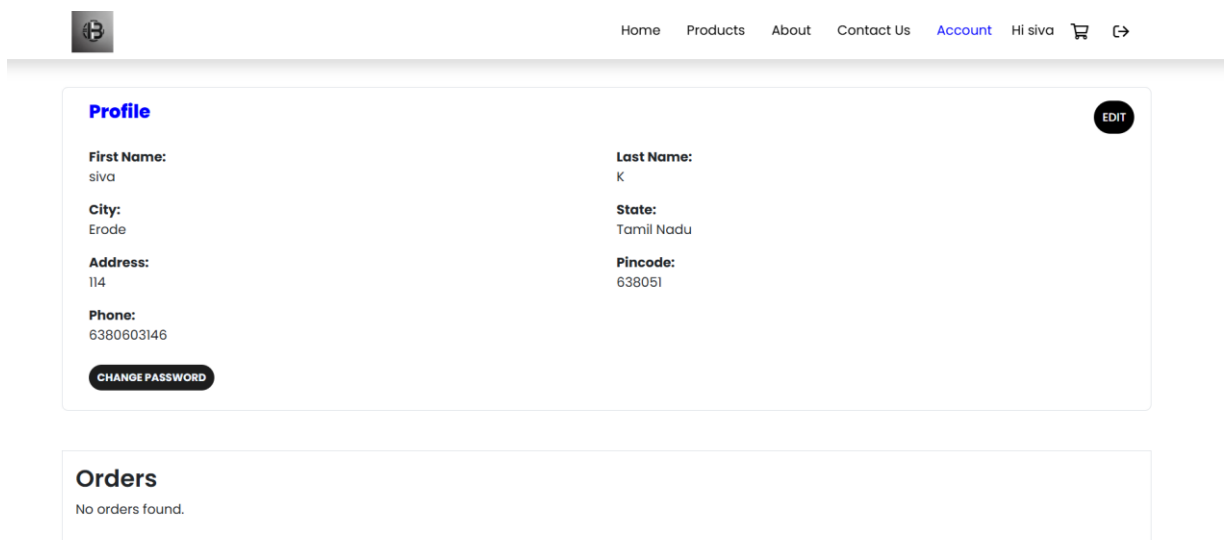


Fig A 2.2 Account Page

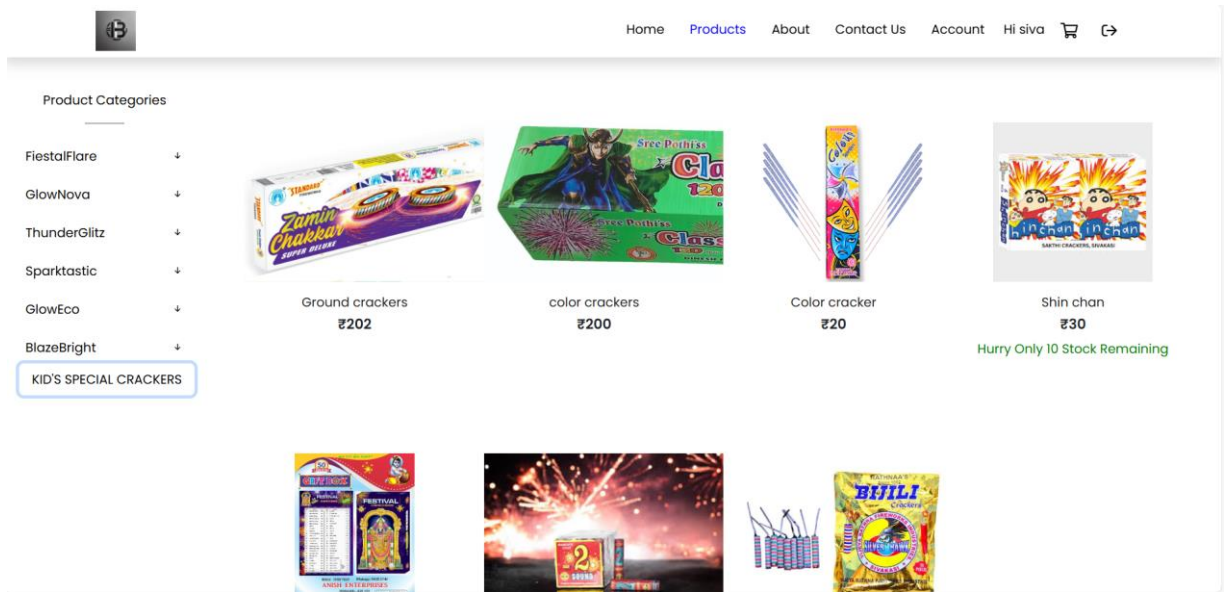


Fig A 2.3 Product Page

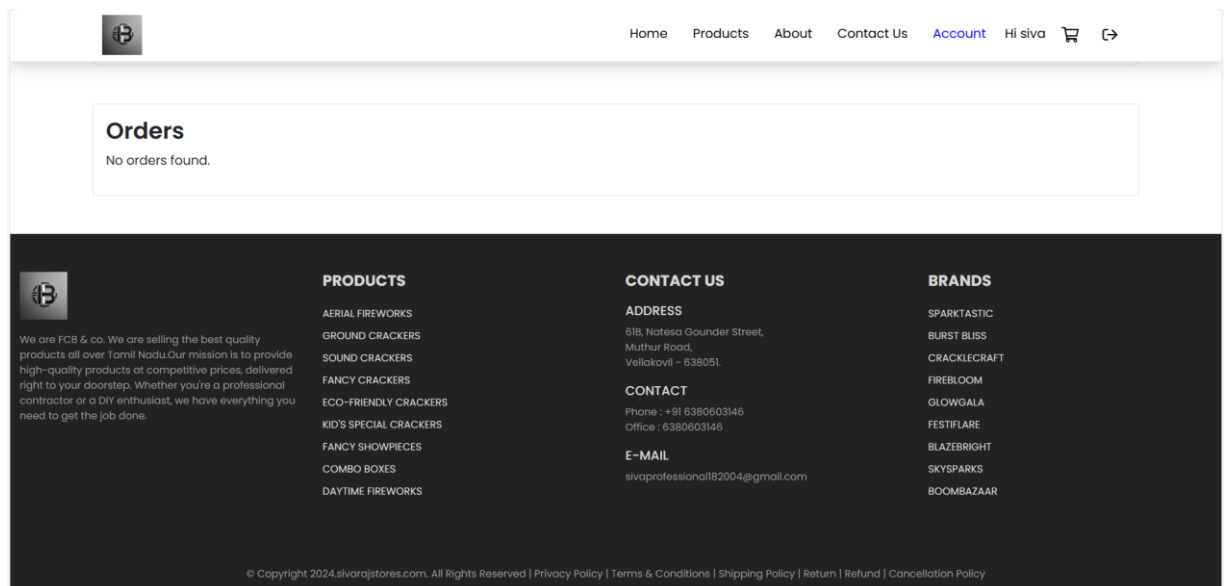


Fig A 2.4 Order Page

Order Confirmation

First Name

Last Name

City

State


Address


Pincode

Phone

Your Order

Cart Items

 7 shots
Quantity: 1

 Bigili
Quantity: 4

Subtotal ₹ 700

GST (18%) ₹ 126

Shipping Cost Free

Total ₹ 826.00

Payment Method:

Fig A 2.5 Order Confirm Page








 Home Products About Contact Us Account Hi siva  					
Remove	Product Image	Product Name	Price	Quantity	Total
		7 shots	₹ 500	- <input type="text" value="1"/> +	₹ 500
		Bigili	₹ 50	- <input type="text" value="4"/> +	₹ 200
Total Cart Price:					₹ 700
Cart Total					
Subtotal					₹ 700
GST (18%)					₹ 126
Shipping Cost					Free
Total					₹ 826.00
Grand Total					₹ 826.00
<input type="button" value="PROCEED TO CHECKOUT"/>					

Fig A 2.6 Cart Page

REFERENCES

1. Alex Banks & Eve Porcello (2020), 'Learning React' Second Edition
2. David Herron (2020), 'Node.js Web Development' Fifth Edition
3. Jonathan Wexler (2019), 'Programming with Node.js' First Edition, Design Academy
4. Robin Nixon, "Learning CSS & HTML5"
5. www.w3school.com and www.geeksforgeeks.org, "Learn JS and Bootstrap".