SPORTYKIT APPLICATION

Submitted by

ABISH MOHAMED R

1P23MC002

In partial fulfillment of the requirements for the award of the Degree of

MASTER OF COMPUTER APPLICATIONS

From Bharathiar University, Coimbatore.

Under the internal supervision of

Ms. K. NARMATHA MCA.,

Assistant Professor

SCHOOL OF COMPUTER STUDIES

MASTER OF COMPUTER APPLICATIONS

RVS COLLEGE OF ARTS AND SCIENCE(AUTONOMOUS)

Sulur, Coimbatore – 641 402.

Nov 2024.

RVS COLLEGE OF ARTS AND SCIENCE
(AUTONOMOUS)
Sulur, Coimbatore – 641 402.

School of Computer Studies
Department of Computer Applications (MCA)

NOV 2024

Register Number: 1P23MC002

Certified bona fide original record work done by **ABISH MOHAMED R**

Guide **Director**

Submitted for the project Evaluation and *Viva voce* held on

Internal Examiner **External Examiner**

DECLARATION

I am, **ABISH MOHAMED R** ,hereby declare that the project entitled **SPORTKIT APPLICATION,** submitted to the School of Computer Studies , RVS College of Arts and Science , in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a record of original project work done by me during the period June to Nov 2024 under the internal supervision of **Ms.K.NARMATHA MCA., ASSISTANT PROFESSOR, RVS COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS)** From Bharathiar University, Coimbatore.

ABISH MOHAMED R
Signature of the Candidate

ACKNOWLEDGEMENTS

I express my sincere thanks to our Managing trustee **Dr.K.Senthil Ganesh MBA (USA).,MS (UK)., Ph.D.,** for providing us with the adequate faculty and laboratory resources for completing my project successfully.

I take this as a fine opportunity to express my sincere thanks to **Dr. T. Sivakumar M.Sc., M. Phil., Ph.D., Principal,** Rathnavel Subramaniam College of Arts And Science (Autonomous) for giving me the opportunity to undertake this project.

I express my sincere thanks to **Dr. P. Navaneetham M.Sc., M.Phil., Ph.D., Director (Administration), Department of Computer Science** for the help and advice throughoutthe project.

I express my sincere thanks to **Dr. S. Yamini M.Sc.,(CC)., M. Phil., Ph.D., Director (Academic), Department of Computer Science** for her valuable guidance and prompt correspondence throughout the curriculum to complete the project.

I express my gratitude to **Ms. K. NARMATHA MCA., Assistant Professor** for his valuableguidance, support, encouragement and motivation rendered by her throughout this project.

Finally, I express my sincere thanks to all other staff members and my dear friends, and alldear and near for helping me to complete this project.

ABISH MOHAMED R

ABSTRACT

This project is dedicated to developing a cutting-edge mobile application for our e-commerce platform focused on sports kit. Our primary goal is to enhance the online shopping experience by offering a user-friendly and highly engaging mobile solution. The application will feature a modern design that ensures effortless navigation and a seamless journey from product browsing to checkout.

Key features of the app will include personalized product recommendations powered by advanced algorithms, tailored exclusively to user preferences and browsing history. We will also introduce app-only promotions to drive user engagement and increase sales. A streamlined checkout process will be implemented to reduce cart abandonment and enhance the overall efficiency of purchasing.

Additionally, the app will offer real-time notifications to keep users informed about their orders, new product arrivals, and special promotions. The app will be built using React Native for cross-platform mobile development, Node.js for backend logic, MongoDB for efficient data management, and expo for testing and debugging on Android devices.
Advanced analytics will be utilized to continually refine the user experience and app performance based on user data. This project aims to significantly boost sports kit sales, foster customer loyalty, and position our e- commerce platform as a leader in the competitive market, driving sustainable growth and expanding our reach.

CONTENTS

Declaration

Acknowledgements

Abstract

CHAPTER 1

1.Introduction 1

1.1 An overview of the project 1

1.2 Mission of the project 1

1.3 Background study 2

1.3.1 A study of the existing system 3

CHAPTER 2

2. System Analysis 4

2.1 A study of the proposed system 4

2.2 User requirement specification 4

2.2.1 Major modules 5

2.2.2 Sub modules 5

2.3 Software requirement specification 6

2.4 System specification 7

2.4.1 ReactNative.js 7

2.4.2 Node.js 7

2.4.3 Mongodb 7

2.4.4 Express. js 7

2.4.5 Hardware configuration 7

2.4.6 Software configuration 8

CHAPTER 3

3. System design and development 9

3.1 Fundamentals of design concepts 20
3.1.1 Abstraction 21
3.1.2 Refinement 21
3.1.3 Modularity 21
3.2 Design Notations 22
3.2.1 System structure chart 23
3.2.2 System flow diagram 23
3.2.3 Data flow diagram / UML 23
3.2.4 Software Engineering Model 24
3.3 Design process 25
3.3.1 Software Architecture 25
3.3.2 Control Hierarchy 25
3.3.3 Structural Partitioning 26
3.3.4 Data structure 26
3.3.5 Software procedure 27
3.4 Database design 28
3.5 Input design 29
3.6 Output design 30
3.7 Development approach 30

CHAPTER 4

4. Testing and implementation 32
4.1 Testing 32
4.1.1 System testing 33
4.1.2 White box testing 34
4.1.3 Unit testing 35
4.2 System implementation 37
4.2.1 System maintenance 37

CHAPTER 5

5. Conclusion
5.1 Directions for future enhancements references 40

5.2 BIBLIOGRAPHY
5.2.1 Book reference 41
5.2.2 website reference 42

ANNEXURE

ANNEXURE - A - Output design 43
ANNEXURE - B - Source code 47

CHAPTER -1

1. Introduction

The e-commerce sporty kit Application is revolutionizing the way businesses and consumers interact, offering unprecedented access to global markets and products. In this new era, companies can sell goods and services online, reaching customers across geographical boundaries with just a few clicks. As digital technologies evolve, so do the possibilities for personalized shopping experiences, seamlesstransactions, and innovative marketing strategies.

Whether you're a startup looking to tap into the global market or a consumer searching for convenience, the e-commerce space provides limitless opportunities for growth and efficiency.

1.1 An overview of the project

The e-commerce sporty kit application project aims to create a seamless digital marketplace where users can browse, compare, and purchase sports kit based on various preferences such as model, new arrival, and price range. The application will feature a comprehensive product listing, complete with detailed descriptions, images, and customer reviews, enabling users to make informed decisions. It will include powerful search and filter functionality to refine results, along with secure user authentication for personalized profiles. Users will be able to add products to a shopping cart, proceed to a secure checkout with multiple payment options, and track their orders in real-time. Additionally, a reviews and ratings system will allow customers to provide feedback, helping others with their purchase choices. Built with React Native and using Android studio for cross-platform compatibility, the app is designed to offer a responsive and user-friendly shopping experience, ensuring accessibility across both iOS and Android devices. Mission of the project

1.2 Mission of the project

The mission of the e-commerce sporty kit application project is to provide a seamless, user-centricplatform that simplifies the sports items purchasing experience by offering a wide range of products, personalized recommendations, and secure, efficient transactions. The project aims to enhance the convenience of online shopping by integrating advanced search and filtering tools, real-time order tracking, and a responsive design that ensures accessibility across all devices. Ultimately, the goal is to create a trustworthy and engaging marketplace where users can easily find, compare, and purchase jewels tailored to their needs.

Problem:

- Handling user data manually can lead to errors and privacy concerns.
- Mistakes in user data management might result in lost messages and compromised privacy. Accurately tracking and calculating overtime can be complex, especially for organizations with varying work hours.
- Maintaining optimal performance during high user activity is a development challenge.

Solution:

- Utilize a robust database system: Implement a secure database system to store user data, ensuring data integrity and quick retrieval of information
- Use authentication protocols: Implement strong user authentication mechanisms, such as OAuth or JWT, to verify user identities securely.
- Caching mechanisms: Implement caching for frequently accessed data to reduce database queries and improve response times.

1.3 Background study

- Analyze how users interact with sporty kit Application, including preferences for text, voice, and details sharing. Understand user expectations for real-time helping application.
- Understand the server architecture of large-scale helping platforms. Focus on handling concurrent connections, real-time message delivery, and data synchronization.
- Analyze user reviews of existing e-commerce Apps.

1.3.1 A study of the existing system

1. Create a comprehensive document outlining the current sporty kit e-commerce application processes. Include details on user registration, message transmission, products details sharing, notifications, and any additional features. Diagram the flow of messages from customer to admin .

2. Conduct interviews or surveys with sporty kit e-commerce to collect feedback. Focus on the app's user-friendliness, responsiveness, and any challenges faced during interactions.Understand user preferences for message formats.

3. **Customer Reviews and Social Proof:** Customer feedback, ratings, and reviews play a critical role in influencing buying decisions. E-commerce platforms are integrating systems that allow for easy interaction between customers and products, using reviews and social proof to build trust.

4. **Sustainability and Ethical Shopping:** Consumers are increasingly mindful of sustainability, and platforms that promote eco-friendly products or ethical business practices tend to attract a conscientious demographic.

CHAPTER-2

2. System Analysis

System analysis involves studying the various components and functionalities of the e-commerce sportykit application, defining its structure, and understanding the requirements to ensure seamless operation. In this project, the system analysis focuses on identifying the users' needs, analyzing the system components, and determining how the application will interact with external systems, like payment gateways and shipping services.

2.1 A study of the proposed system

In the context of the sportykit project, our primary objectives revolve around creating a digital space that promotes meaningful human connections while ensuring user safety and satisfaction. This involves developing features aimed at fostering interactive conversations, customer details sharing. The proposed system's key tasks include designing an intuitive user interface, implementing real-time helping functionalities, and integrating multi details sharing capabilities.

2.2. User requirement specification

In the context of the sportykit app, customer require a platform that facilitates seamless and enriching communication experiences. The app should helping a user-friendly interface, allowing individuals to engage in real-time conversations, share customer details content effortlessly customer expect features like instant product detail, address, whishlist , product details sharing functionalities, and the ability to search and connect with others based on preferences.

In terms of management, the app should enable customer to search for others and add them as details. To maintain a safe environment, customer should also be able to images other products if they encounter inappropriate behavior.

2.2.1 Major modules

- **Authentication Module**

This module manages user authentication for the sporty kit app, handling processes such as user registration, login, and secure access control to ensure a safe environment for customers.

- **Product calalog Module**

The core of the e-commerce application, this module manages the entire sports item inventory,allowing users to browse and search products efficiently**.**

- **Shopping cart and checkout Module**

This module ensures that users can easily add items to a shopping cart and complete their purchases through a secure checkout process.Customer Experience Enhancement Module

- **Order management and Tracking Module**

The Customer Experience Enhancement Module concentrates on optimizing the app's customer interface, ensuring it is intuitive and user-friendly. This includes aspects of UI/UX design, responsive design techniques, and user testing to create a seamless and enjoyable customer experience for users.

2.2.2 Sub modules

- **Signup Sub-Module**

The Signup Module facilitates the customer registration process within the sportykit app. Customer can create their accounts by providing essential information like a username, password, email address, and image upload details.

- **Login Sub-Module**

Customer enter their credentials (username/email and password) to securely access the sportykit application, verifying their identity. This submodule ensures user privacy and grants authenticated access for real-time helping within the platform.

- **Customer Listings Sub-Module**

The Customer Listings Module presents digital world with a curated catalog of active customer conversations, showcasing relevant details, customer, and conversation.

- **Customer Details Sub-Module**

The Customer Details Module provides customer with comprehensive insights into specific chat conversations within the app. This submodule displays essential information, including customers, timestamps, and customer booking histories, offering customer a detailed overview of the conversation context.

- **Profile Page Sub-Module**

The Profile Page Module in the sportykit app allows customer to personalize their digital presence. This submodule enables users to view their profile information like email and password, login and logout.

2.3 Software requirement specification

In the context of the sportykit app, software specifications pertain to the detailed documentation outlining functional and non-functional requirements. This includes features, security protocols, and real-time donating mechanisms necessary for the app's operation. Meanwhile, hardware specifications refer to the specific details regarding the physical components of devices where the digital world app will be installed. This encompasses information about processors, RAM, storage, and network capabilities, ensuring compatibility and seamless performance on various devices. Both sets of specifications are crucial in defining the app's functionalities and ensuring optimal user experience across different platforms and devices.

2.4 System specification

The sportykit app, built using React Native, ensures cross-platform compatibility on iOS and Android. MongoDB serves as the backend database, storing user profiles, messages, and multimedia content, while Node.js and Express.js handle server-side logic and real-time helping. Security measures include JWT authentication for user data privacy. The app is optimized for diverse hardware, accommodating different processors and RAM capacities. Its network compatibility spans Wi-Fi and mobile data, ensuring uninterrupted chat services across various devices and network environments.

2.4.1 ReactNative

In a MERN stack project to React Native involves transitioning from a web-based application to a mobile app. React Native allows you to build mobile apps for iOS and Android using JavaScript and React Native.

2.4.2 Node.js

In the context of the sportykit app, Node.js serves as the JavaScript runtime environment responsible for constructing and operating the server-side functionalities. It enables real-time communication, data processing, and user interactions within the app. The choice of Node.js version is crucial, aligning with the project's specific dependencies and libraries

2.4.3 MongoDB

In the context of the sportykit app, specifying the version of MongoDB is vital to maintain compatibility with the database driver and Node.js. MongoDB, a NoSQL database, isemployed to store and manage user profiles, customer details, and multimedia content in a document-oriented structure. This ensures seamless organization and retrieval of customer data, enhancing the app's performance and user experience.

2.4.4 Express js

In Express.js is a widely-used framework for Node.js that provides a streamlined way to build APIs and manage HTTP requests in web applications. It provides a set of features for building web and mobile applications, including robust routing, middleware support, and easy integration with various template engines.

2.4.5 Hardware configuration

The minimum hardware requirements for a MERN stack project will vary based on thescope and intricacy of the application.

- Processor AMD RYZEN
- RAM 8GB
- Dual-core CPU with a minimum of 3.0 GHz
- High-speed internet connection

2.4.6 Software configuration

Environment

Win11Platform

FrameworkVS

code Database

MongoDB 3.6

CHAPTER-3

3. System design and development

The system design and development of the e-commerce sportykit application is structured using a three-tier architecture, comprising the presentation layer (frontend), the application layer (business logic), and the data layer (database). This architecture ensures separation of concerns, scalability, and maintainability, making the system robust and adaptable to future growth.

The presentation layer is responsible for the user interface (UI) and overall user experience (UX), enabling users to interact seamlessly with the platform. The web interface will be developed using React, providing a dynamic and responsive interface that adjusts to various screen sizes. The mobile application will be built using React Native, ensuring cross-platform compatibility for both iOS and Android devices. For managing the application's state across various components, Redux will be used to ensure a consistent data flow. The combination of HTML5, CSS3, and JavaScript/TypeScript will allow for interactive, responsive, and visually appealing pages. The design will prioritize intuitive navigation, fast loading times, and accessibility features to cater to a diverse user base, including those with disabilities.

3.1 Fundamentals of design concepts

The design concept of the e-commerce sports kit application is grounded in fundamental principles that prioritize **user experience (UX)**, **scalability**, **security**, and **maintainability**. At the core of the designis **user-centricity**, ensuring that the application is intuitive and easy to navigate for both novice and experienced users. A **responsive design** approach is employed to provide a seamless experience acrossdevices, including desktops, tablets, and smartphones, ensuring the interface adjusts fluidly to differentscreen sizes.

Consistency in visual and interaction design is another key principle. The application will follow a uniform layout and design language, including typography, color schemes, and component behavior, enhancing usability and reducing user friction. **Accessibility** is integrated into the design, adhering to WCAG standards to make the platform usable by individuals with disabilities.

3.1.1 Abstraction

In the context of developing a sporty kit application, abstraction simplifies intricate system components byemphasizing their core features while concealing unnecessary intricacies. This approach enables developers to create a high-level representation of the app, focusing on essential functionalities such as real-time helping, multimedia sharing, and customer interactions.

3.1.2 Refinement

The refinement of the design concept for the e-commerce sporty kit application involves a continuous process of **improving usability**, **enhancing performance**, and **optimizing functionality** based on user feedback and evolving technological trends. Refinement focuses on perfecting the balance between aesthetics and functionality, ensuring that the user interface (UI) is not only visually engaging but also intuitively designed for effortless navigation.

3.1.3 Modularity

Modularity is a fundamental principle in the design of the e-commerce sporty kit application, emphasizing the breakdown of the system into distinct, independent components or modules that can function individually or in combination. Each module is designed to handle a specific functionality, such as user authentication, product catalog management, or payment processing,making the system more **scalable**, **maintainable**, and **flexible**.

3.2 Design Notations

3.2.1 System structure chart

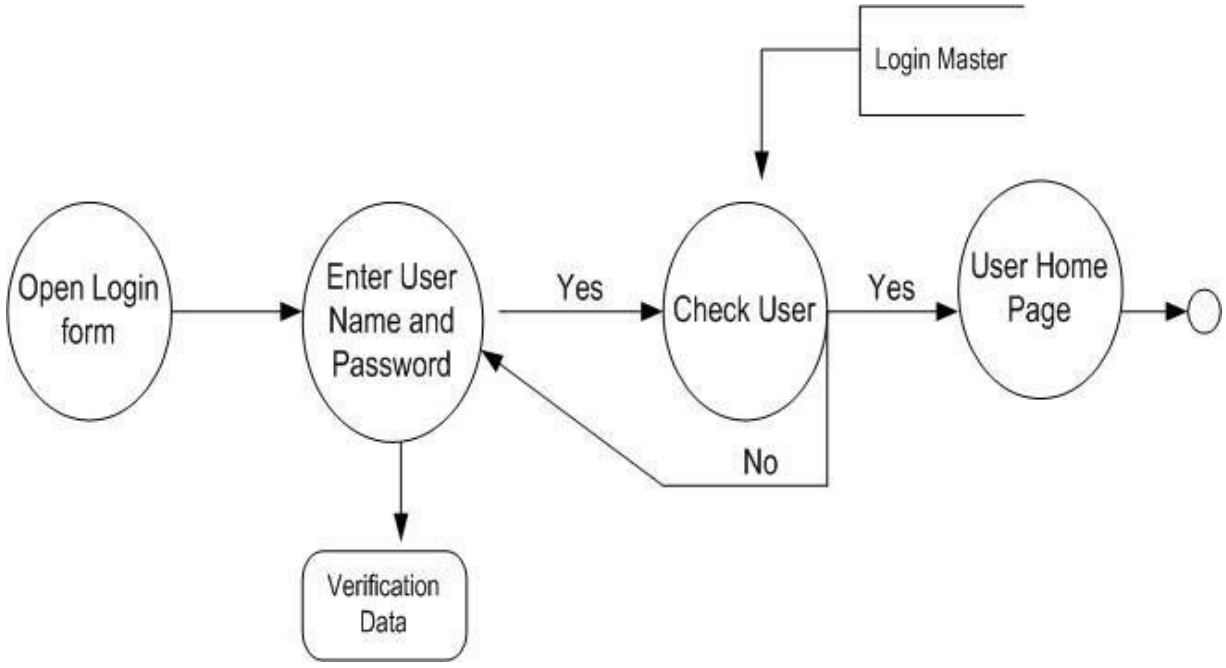
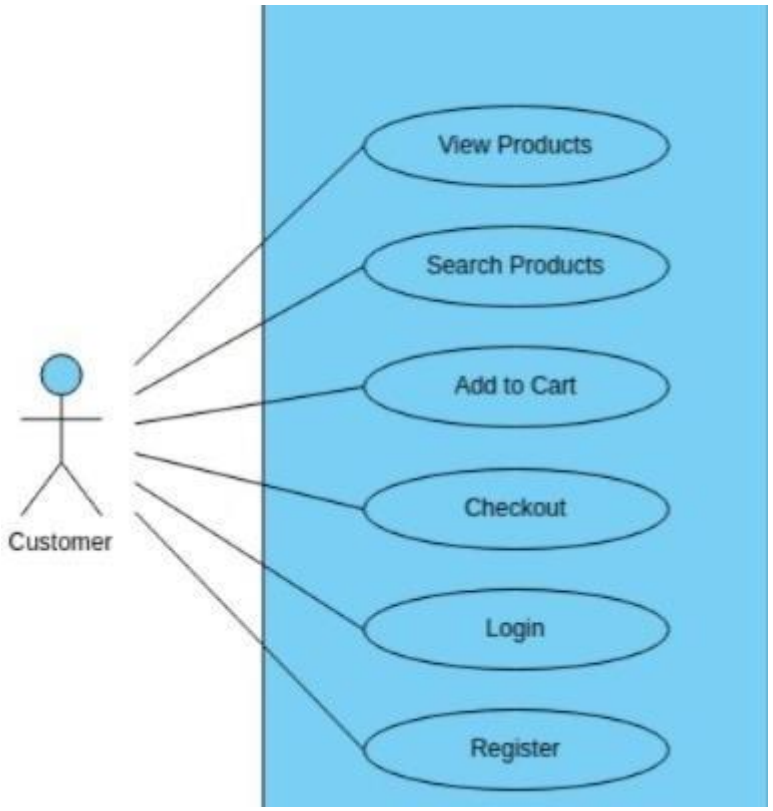


Figure 3.1 - System structure chart

3.2.2 System flow diagram



3.2.2 Data flow diagram / UML

A Data Flow Diagram (DFD) is a structured analysis and design tool that can be used for flowcharting. A DFD is a network that describes the flow of data and the processes that change or transform the data throughout a system. This network is constructed by using a set of symbols that do not imply any physical implementation. It has the purpose of clarifying system and identifying major transformations. So it is the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. DFD can be considered to an abstraction of the logic of an information-oriented or a process-oriented system flow-chart. For these reasons DFD's are often referred to as logical data flow diagrams.

EXTERNAL ENTITY

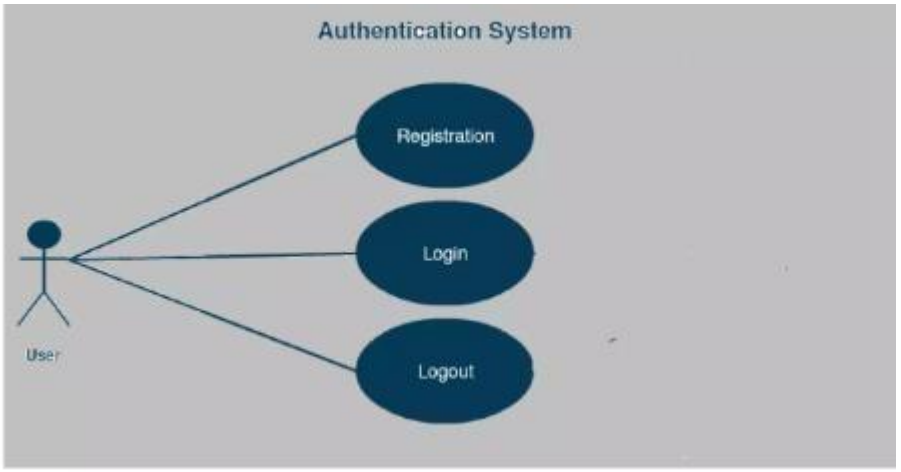
An external entity is a source or destination of a data flow. Only those entities which originate or receive data are represented on a data flow diagram. The symbol used is a rectangular box.

PROCESS

A process shows a transformation or manipulation of data flow within the system. The symbol used is an oval shape.

DATAFLOW

The data flow shows the flow of information from a source to its destination. Dataflow is representedby a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal or electronic. Each data flow may be referenced by the processesor data storeset its head and tail, or by a description of its contents.



3.2.3 Software Engineering Model

A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the order of activities of theprocess and the sequence in which they are performed.

In the system, we can observe that the user interacts with the application through a graphical user interface. The inputs to the system are the Search and Filter criteria provided by the user and a new review written by the user. Also, the output is in the form of Repeater and grid views which present the customer with list . The users can view complete specification, view Images.

3.3 Design process

3.3.1 Software Architecture

- Software architecture is the set of structures needed to reason about a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations.
- The architecture of a software system is a metaphor, analogous to the architecture of a building. It functions as the blueprints for the system and the development project, which project management can later use to extrapolate the tasks necessary to be executed by the teams and people involved.

- Software architecture is about making fundamental structural choices that are costly to change once implemented. Software architecture choices include specific structural options from possibilities in the design of the software.

3.3.2 Control Hierarchy

- Control hierarchy, also called program structure, represents the organization of program components (modules) and implies a hierarchy of control. It does not represent procedural aspects of software such as sequence of processes, occurrence or order of decisions, or repetition of operations; nor is it necessarily applicable to all architectural styles.
- The control hierarchy also represents two subtly different characteristics of the software architecture: visibility and connectivity. Visibility indicates the set of program components that may be invoked or used as data by a given component,even when this is accomplished indirectly.
- A control system manages, commands, directs, or regulates the behavior of other devices or systems using control loops. It can range from a single home heatingcontroller using a thermos state controlling a domestic boiler to large industrial control systems which are used for controlling processes or machines. The control systems are designed via control engineering process.

3.3.3 Structural Partitioning

- The program structure should be partitioned both vertically and horizontally. As describe in horizontal partitioning describe separate branches of the modular hierarchy of reach major program function. The Control modules, represented in a darker shade are used to coordinate communication among and execution of program functions. The easiest approach to horizontal partitioning describes 3 partitions - input, data transformation often called processing and output.

- The nature of modification in program architectures justifies the requirement for vertical partitioning. The change in a control module high in the architecture will have a higher possibility of propagating side effects to modules which are subordinate to it. The change to a worker module given its low level in the structure is less likely to cause the propagation of side effects. In common changes to computer programs revolve around changes to input.

3.3.4 Data structure

- Data structure is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.
- Depending on your requirement and project, it is important to choose the right data structure for your project. For example, if you want to store data sequentially in the memory, then you can go for the Array data structure.
- Data structures are generally based on the ability of a computer to fetch and store data at any place in its memory, specified by a pointer—a bit string, representing a memory address,that can be itself stored in memory and manipulated by the program. Thus, the array and record data structures are based on computing the addresses of data items with arithmetic operations, while the linked data structures are based on storing addresses of data items within the structure itself.

3.3.5 Software procedure

The development process starts with crafting a high-level design that outlines the application's architecture and selecting an appropriate technology stack. User authentication and authorization mechanisms are implemented securely, incorporating methods like email/phone verification and robust password hashing. User profiles are created, allowing users to personalize their experience with profile details and pictures.

A key aspect of the digital world app is real-time help, requiring the integration of instant details delivery and support for various booking types, including text, images, videos , audio.Additionally, the app includes booking details functionality, enabling customers to create both return and buying options. Customer administration features, such as member management and details settings, are implemented for a smooth customer experience.

3.4 DataBase Design

Databases are the storehouses of data used in the software systems. The data is stored in collection inside the database. Several collections are created for the manipulation of the data for the system. MongoDB is a NoSQL database, which means it doesn't use the traditional relational table-based structure. Instead, it stores data in flexible, JSON-like documents.

- USER DETAIL

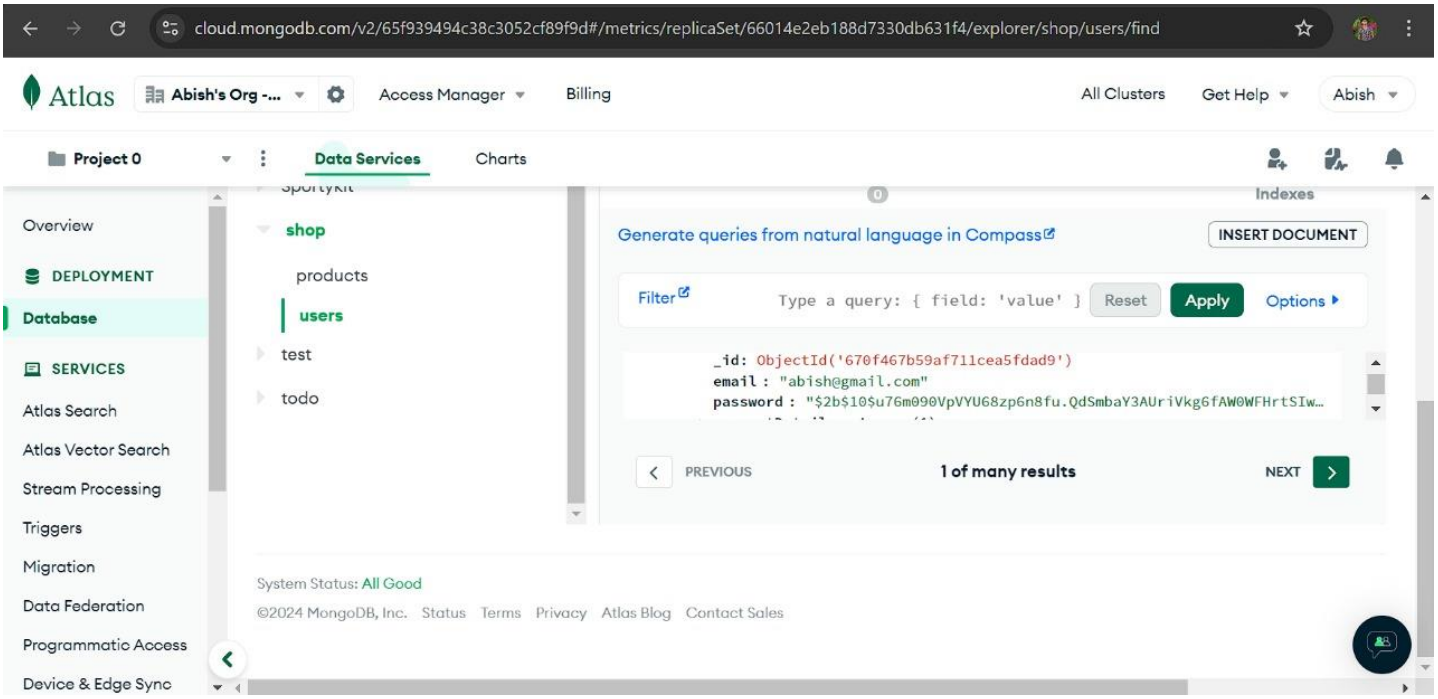


Fig 3.7: User Detail

➢ PRODUCT DETAILS

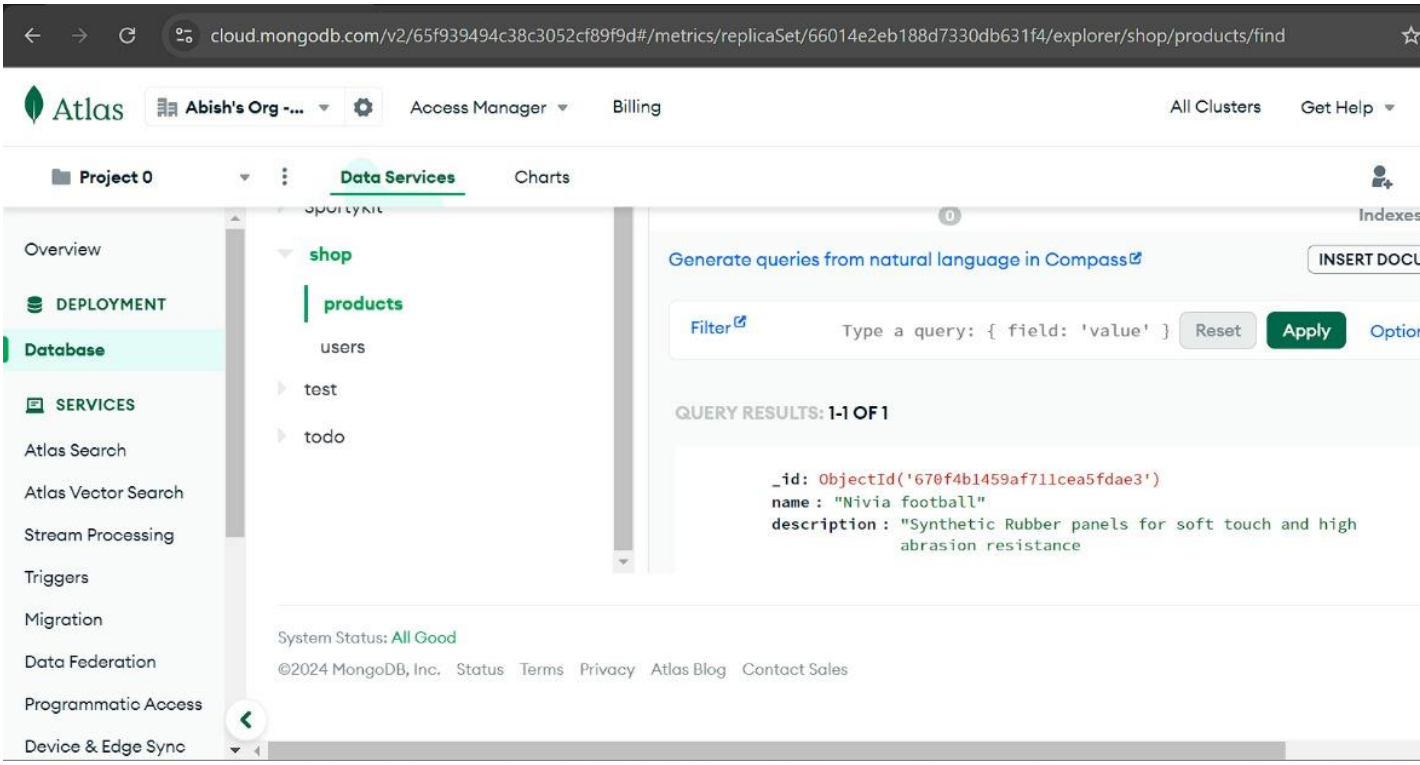


Fig 3.8: Product detail

3.5 Input design

In the context of a sporty kit Application, input design plays a pivotal role in ensuring a seamless user experience. It serves as the bridge connecting the customers with the digital platform, allowing themto input information effectively and interact with the application effortlessly. The input design process involves meticulous planning, determining the types of inputs required, validatingthe data entered, and optimizing the data entry process for efficiency.

Inaccurate inputs are a common source of errors in digital systems, and input design acts as a control mechanism to mitigate such errors. It encompasses the conversion of user-originated inputs into a format compatible with the digital world interface. Customer inputs, such as messages, appointment files, and commands, are collected and organized logically.

For instance, when booking a appointment, the input design ensures that the message format is validand within the customer limit. If a customer attempts to send an detail message or a message containing prohibited content, the system issues a warning message, guiding the customer to correct the input.

3.6 Output design

Sporty kit applications, computer output takes center stage as the primary means of relaying information to users. Output design holds paramount importance as it directly influences how information is presented efficiently and intelligibly. Well-designed output enhances the system's interaction with users, aiding in effective decision-making processes.

Efficient output design ensures that customer receive details and messages in a clear, organized, and timely manner. These notifications are critical for real-time communication, ensuring that customer are promptly informed about new appointments, or any other relevant detailswithin the blood donation application.

3.7 Development approach

Given the intricate nature of sporty kit applications, it is vital to prioritize customer experience and fluid communication. Agile development methodologies, such as Scrum or Kanban, prove invaluable.

Through testing remains a cornerstone, particularly in blood donation applications where user engagement and satisfaction are directly tied to system performance. Rigorous testing of features like instant messaging, multimedia sharing, and real-time details is imperative to guarantee a glitch-free user experience.

In the context of a sporty kit application, the development approach should be adaptable, allowing for quick adjustments based on customer feedback and emerging trends. The choice of technology stack, the expertise of the development team, and the preferences of the organization heavily influence the approach. By embracing an agile mindset, maintaining open lines of communication with stakeholders, and prioritizing user-centric design, developers can create a sporty kit application that meets but exceeds user expectations, fostering a vibrant and engaging online community.

Chapter-4

4. Testing and implementation

4.1 Testing

- Software testing is the act of examining the artifacts and the behavior of the software under test by validation and verification. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

- Testing comprises all activities to accomplish a satisfactory level of confidence that the system under development fulfills it intended purpose. Objects of testing can be documents such as specifications or software such as a module or a complete system. The goal of testing is to find errors and remove the causes of the errors.

- Testing and implementation are obviously closely connected. If a concept is to be executed on a computer, as the definition of “implementation” states, then the implementation must run without errors, otherwise the concept has not been implemented or at least not correctly implemented. In practice, implementation and testing go hand-in-hand.

- Implementation refers to a formulated plan for a designated process. It is important for any process to have a completed plan and clear objectives. The plan usually contains several actions that need to be carried out. Each of these actions needs to be tested, which is called implementation testing.

4.1.1 System testing

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS). System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. System Testing is a black-box testing. System Testing is performed after the integration testing and before the acceptance testing.

System testing verifies that an application performs tasks as designed. It's a type of black box testing that focuses on the functionality of an application rather than the inner workings of a system, which white box testing is concerned with.

4.1.2 White box testing

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

Using white-box testing software developer can derive test case that

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

At every stage of project development, I have tested the logics of the program by supplying the invalid inputs and generating the respective error messages. All the loops and conditional statements are tested to the boundary conditions and validated properly.

4.1.3 Unit testing

Unit testing emphasizes the verification effort on the smallest unit of software design i.e.a softwarecomponent or module. Unit testing is a dynamic method for verification, where program is actually compiled and executed. Unit testing is performed in parallel with the coding phase. Unit testing testsunits or modules not the whole software.

I have tested each view/module of the application individually. As the modules were built up testing was carried out simultaneously, tracking out each and every kind of input and checking the corresponding output until module is working correctly.

The functionality of the modules was also tested as separate units. Each of the three modules was tested as separate units. In each module all the functionalities were tested in isolation.

Unit testing for a digital world app management system involves breaking down the system into small, testable units or modules. Develop comprehensive test cases covering normal, boundary, and error scenarios. Automate testing where possible, isolating dependencies and focusing on accuracy in calculations.

Visual Studio has in built support for testing the application. The unit testing can be done using visual studio without the need of any external application. Various methods have been created for the purpose of unit testing. Test cases are automatically generated for these methods. The tests run under context which means settings from file are automatically picked up once the test case starts running. Methods were written to retrieve all the manufacturers from the database, strings that match a certain search term, products that match certain filter criteria, all details that belong to a particular customer etc. Unit test cases were automatically generated for these methods and it can be seen that the tests have passed.

Unit Testing Techniques:

1. Black Box Testing**:** This testing technique is used in covering the unit tests for input, user interface, and output parts.

2. White Box Testing**:** This technique is used in testing the functional behavior of the system by giving the input and checking the functionality output including the internal design structure and code of the modules.

3. Gray Box Testing**:** This technique is used in executing the relevant test cases, test methods, test functions, and analyzing the code performance for the modules.

4.2 System implementation

Implementation refers to a formulated plan for a designated process. It is important for any process to have a completed plan and clear objectives. The plan usually contains several actionsthat need to be carried out. Each of these actions needs to be tested, which is `called implementation testing.

1. It is vital to test implementations of technological specifications.
2. Implementation testing improves certain interfaces so that developers can easily understand them.
3. It confirms that we can implement a specific identified action.
4. Any ambiguous, contradictory, or unimplementable actions or activities are easily identified using implementation testing.
5. Implementation testing provides a kind of quality test on our implementations.

During test implementation, test manager has also to liaise with the IT infrastructure team and figure out the test environment availability and configuration.

Implementation is the realization of a design so that it can be executed on a computer. This includes the realization of: the system's classes; the user interface; and the database structures.

4.2.1 System maintenance

Maintenance means restoring something to its original conditions. Enhancement means adding, modifying the code to support the changes in the user specification. System maintenance conforms the system to its original requirements and enhancement adds to system capability by incorporating new requirements.

Thus, maintenance changes the existing system, enhancement adds features to the existing system, and development replaces the existing system. It is an important part of system development that includes the activities which corrects errors in system design and implementation, updates the documents, and tests the data.

System maintenance is a catchall term used to describe various forms of computer or server maintenance required to keep a computer system running properly. It can describe network maintenance, which could mean that servers are being physically repaired, replaced, or moved. Network maintenance can also mean that the software for a server is being updated, changed, or repaired. This sort of maintenance is typically performed on a regular or semi-regular schedule, often during non-peak usage hours, and keeps servers running smoothly.

The systems maintenance is an activity that is becoming more and more important. Many companies, which increasingly depend on Information Technologies, have realized that a correct investment in maintenance can mean savings in the medium and long term, and that is why they increasingly need this type of professional services.

CHAPTER-5

5. Conclusion

In summary, the creation and implementation of a well-crafted sportykit application are paramount in the digital age, serving as a cornerstone for seamless communication and connection among customer.A meticulously designed sporty kit application helping a platform for real-time donating, multimedia sharing, and instant details communities.

Comprehensive testing, spanning unit tests, integration tests, and performance evaluations, is indispensable. Rigorous testing procedures identify and rectify potential glitches, ensuring a smooth and error-free user experience. Regular updates and maintenance are essential to keep the app aligned with emerging technologies and evolving user expectations.

Documentation plays a pivotal role, serving as a guide for system architecture, processes, and testing scenarios. Thorough documentation not only aids in ongoing support but also facilitates training for new users and developers.

In essence, a well-developed sportykit application not only enables seamless communication but also cultivates online communities, fostering connections and collaborations. By prioritizing user security, experience, and on going maintenance, a sportykit app becomes a catalyst for meaningful interactions, enhancing the digital landscape and promoting positive customer engagement.

5.1 Directions for future enhancements references

The following things can be done in future.

- The integration of self-service portals stands as a promising feature. By allowing users to manage their profiles, update personal details, and access essential chat-related documents, the application empowers users, granting them greater control over their experience. This not only enhances user satisfaction but also reduces the administrative burden on the system.

- A forward-thinking addition would be the incorporation of real-time help alerts. Users could opt-in for notifications that trigger when customer fall below specified thresholds. This feature not only provides users with valuable information but also enrichestheir customer experience, enabling them to make well-informed decisions.

- The current system, primarily focused on the digital world, could be extended to include a user-friendly checkout process. Allowing users to store multiple customer and addresses and facilitating selection through intuitive drag-and-drop functionality can significantly improve the efficiency of the checkout process. This streamlined approach ensures a seamless experience, enhancing user satisfaction and encouraging help usage.

5.2 Bibliography

5.2.1 Book References

1."Learning React: A Hands-On Guide to Building Web Applications Using React and Redux", Author: Kirupa Chinnathambi, Publisher: Addison-Wesley Professional, Year: 2020, Edition: 2nd Edition

2."Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node", Author: Vasan Subramanian, Publisher: Apress, Year: 2019, Edition: 2nd Edition

3."Full-Stack React Projects: Learn MERN Stack Development by Building Modern Web Apps Using MongoDB, Express, React, and Node.js", Author: Shama Hoque
,Publisher: Packt Publishing, Year: 2020, Edition: 2nd Edition

4."The Road to React: Your Journey to Master React.js in JavaScript",Author: Robin Wieruch, Publisher: Independently Published, Year: 2020, Edition: 4th Edition

5."MongoDB: The Definitive Guide: Powerful and Scalable Data Storage", Author: Shannon Bradshaw, Eoin Brazil, Kristina Chodorow, Publisher: O'Reilly Media, Year: 2019, Edition: 3rd Edition

These books provide in-depth knowledge on MERN stack development, from React front-end programming to server-side scripting with Node.js and managing databases using MongoDB.

5.2.2 website reference

- All about ReactNative
  https://reactnative.dev/

- Wikipedia for various diagrams & testing methods
  http://www.wikipedia.org/

- All about NodeJs
  http://nodejs.org/

- All about npm
  http://www.npmjs.com

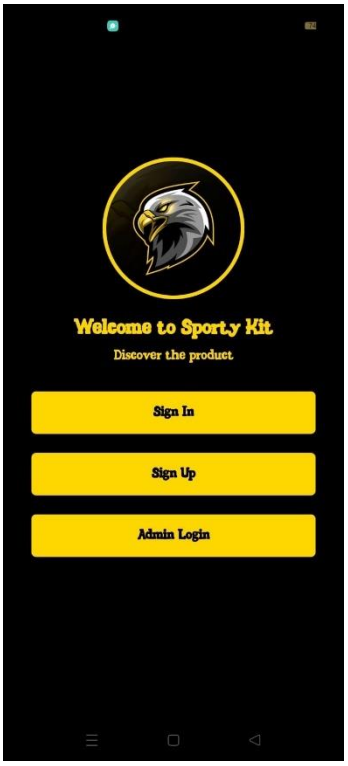
- Smart Draw for drawing all the Diagrams used in this report.
  http://www.smartdraw.com/

- Start Developing with MongoDB
  http://www.mongodb.com/

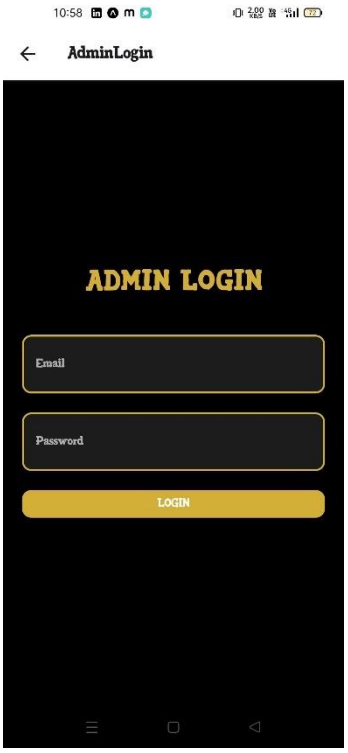
ANNEXURE

ANNEXURE –A-OUTPUT DESIGN

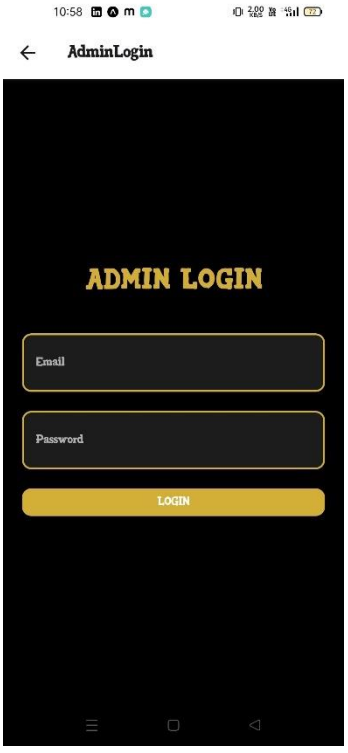
Welcome page:



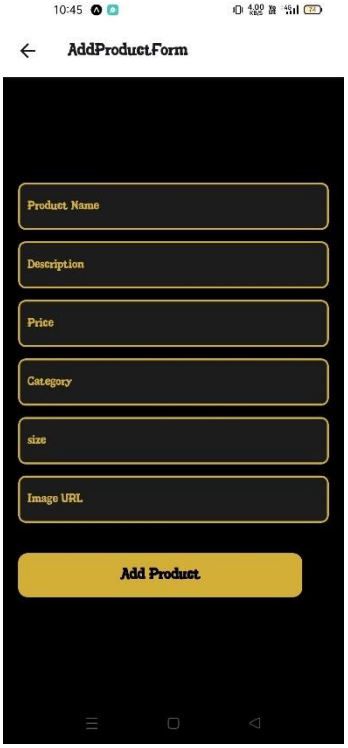
ADMIN LOGIN



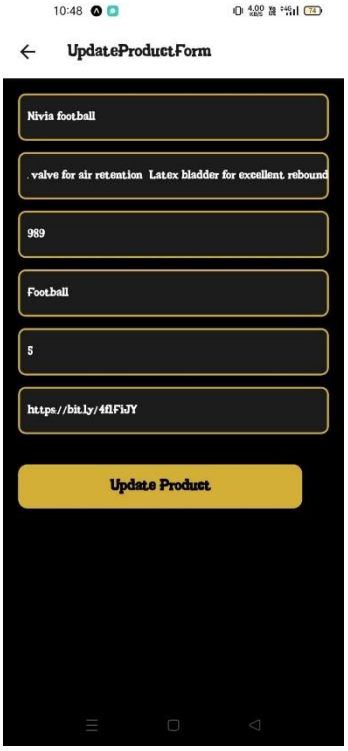
ADMIN PRODUCT MANAGEMENT

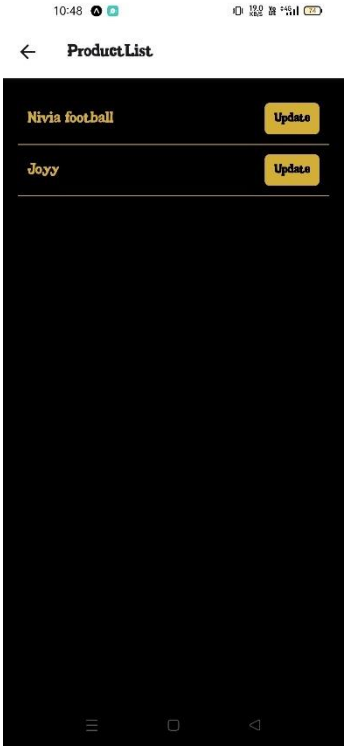


ADD PRODUCT FORM

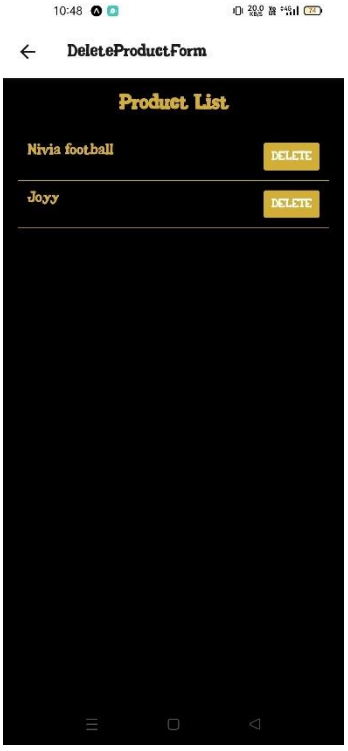


Update page :

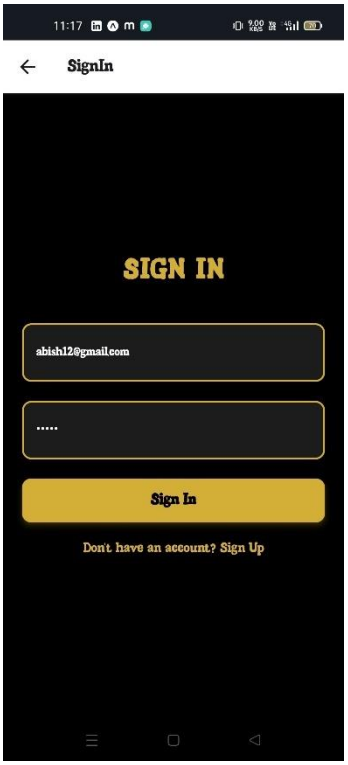




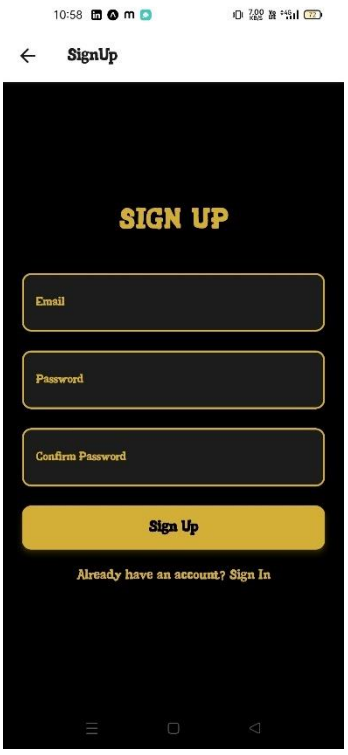
DELETE PRODUCT FORM PAGE



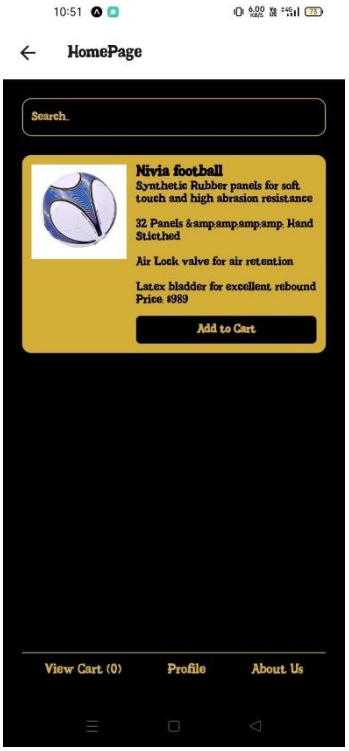
SIGN-IN PAGE:



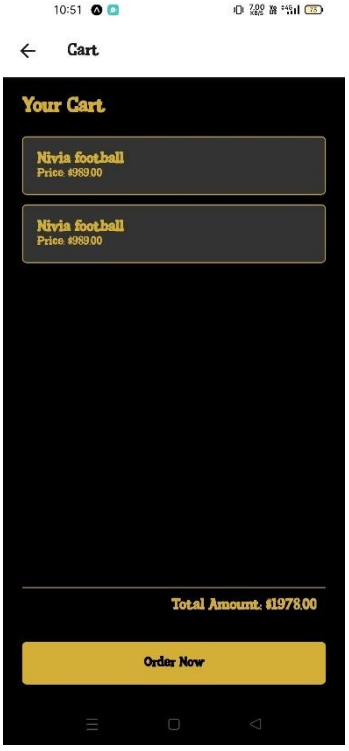
SIGN-UP :



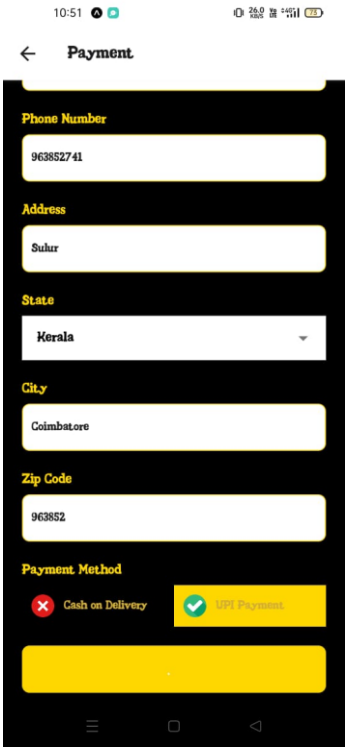
HOME PAGE:



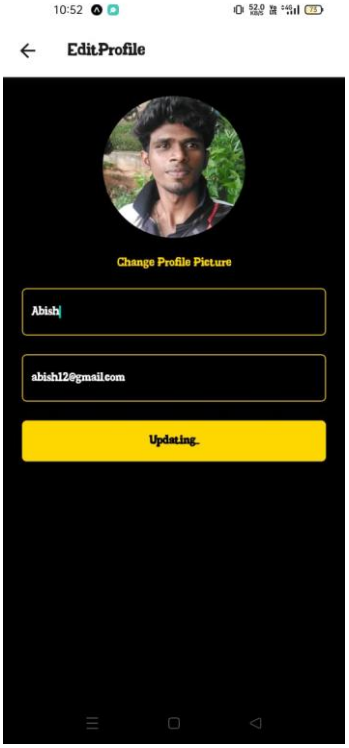
CART PAGE

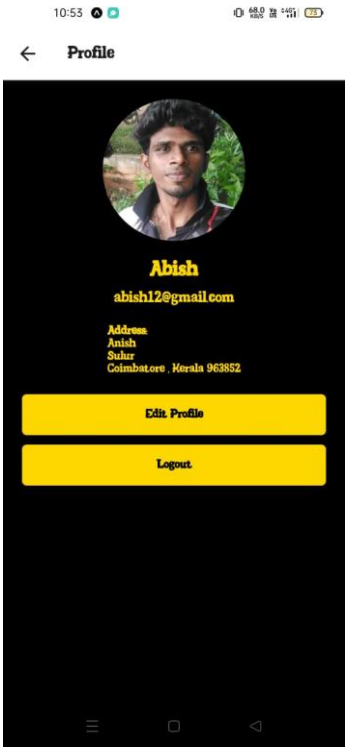


PAYMENT PAGE:

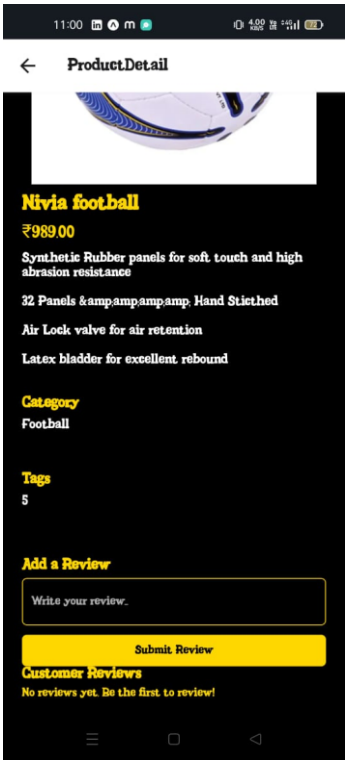


PROFILE PAGE:





PRODUCT DETAILS:



ABOUT US:

10:52
About Us
Sporty Kit
SPORT
INSPIRE
Click For More: www.yourwebsitename.com
Be Part Of The Year's Biggest Cycling Event!
Welcome to Elite Sports Club
At Elite Sports Club, we are committed to providing a world-class sports experience. From soccer to basketball, we offer top-tier facilities, training programs, and competitions for athletes of all ages and skill levels.
Our Sports Programs
Our club offers a wide range of sports programs including soccer, basketball, tennis, and swimming. Whether you're aiming for professional excellence or just want to stay active, our expert coaches and

Logo.js

```
import React, { useEffect, useRef } from 'react';
import { View, Text, Image, StyleSheet, Animated, TouchableOpacity } from 'react-native';
import { useNavigation } from '@react-navigation/native';

export default function LogoPage() {
  const fadeAnim = useRef(new Animated.Value(0)).current;
  const navigation = useNavigation();

  useEffect(() => {
    Animated.timing(fadeAnim, {
      toValue: 1,
      duration: 2000,
      useNativeDriver: true,
    }).start();
  }, [fadeAnim]);

  const handleSignin = () => {
    navigation.navigate('SignIn');
  };

  const handleSignup = () => {
    navigation.navigate('SignUp');
  };

  const handleAdminLogin = () => {
    navigation.navigate('AdminLogin');
  };

  return (
    <View style={styles.container}>
      <Animated.View style={{ ...styles.logoContainer, opacity: fadeAnim }}>
        <Image
          source={require('./assets/images/snack.jpeg')}
          style={styles.logo}
          resizeMode="contain"
        />
      </Animated.View>
      <Text style={styles.title}>Welcome to Sporty Kit</Text>
      <Text style={styles.subtitle}>Discover the product</Text>
      <View style={styles.buttonContainer}>
        <TouchableOpacity style={styles.button} onPress={handleSignin}>
          <Text style={styles.buttonText}>Sign In</Text>
        </TouchableOpacity>
        <TouchableOpacity style={styles.button} onPress={handleSignup}>
          <Text style={styles.buttonText}>Sign Up</Text>
        </TouchableOpacity>
        <TouchableOpacity style={styles.adminButton} onPress={handleAdminLogin}>
          <Text style={styles.buttonText}>Admin Login</Text>
        </TouchableOpacity>
      </View>
```

```
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: 'black', // Background set to black
    alignItems: 'center',
    justifyContent: 'center',
    padding: 20,
  },
  logoContainer: {
    marginBottom: 20,
    alignItems: 'center',
    justifyContent: 'center',
  },
  logo: {
    width: 150,
    height: 150,
    borderWidth: 4,
    borderColor: '#FFD700', // Golden yellow border for the logo
    borderRadius: 100,
    padding: 10,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    color: '#FFD700', // Golden yellow for the title text
    marginBottom: 10,
  },
  subtitle: {
    fontSize: 16,
    color: '#FFD700', // Golden yellow for the subtitle text
    textAlign: 'center',
    marginBottom: 20,
  },
  buttonContainer: {
    flexDirection: 'column',
    justifyContent: 'center',
    width: '100%',
    maxWidth: 300,
  },
  button: {
    backgroundColor: '#FFD700', // Golden yellow background for buttons
    padding: 15,
    borderRadius: 5,
    marginVertical: 10,
    alignItems: 'center',
  },
  buttonText: {
```

```
    color: 'black', // Black text for the buttons
    fontSize: 16,
    fontWeight: 'bold',
  },
  adminButton: {
    backgroundColor: '#FFD700', // Golden yellow background for the admin login button
    padding: 15,
    borderRadius: 5,
    marginVertical: 10,
    alignItems: 'center',
  },
});
```

App.js

```
import React from 'react';
import { createStackNavigator } from '@react-navigation/stack';
import { NavigationContainer } from '@react-navigation/native';
import LogoPage from './Logo';
import SignIn from './signin';
import SignUp from './signup';
import HomePage from './home';
import PaymentPage from './PaymentPage'
import Profile from './profile';
import AdminLogin from './AdminLogin';
import AdminProductManagement from './AdminProductManagement';
import AddProductForm from './AddProductForm';
import DeleteProductForm from './DeleteProductForm';
import UpdateProductForm from './UpdateProductForm';
import CartPage from './CartPage';
import ProductList from './ProductList';
import { UserProvider } from './UserContext';
import EditProfile from './EditProfile';
import AboutUs from './AboutUs'
import ProductDetail from './ProductDetail';

const Stack = createStackNavigator();

export default function App() {
  return (
    // <CartProvider>
    <UserProvider>
      <NavigationContainer>
        <Stack.Navigator initialRouteName="LogoPage">
          <Stack.Screen
            name="LogoPage"
            component={LogoPage}
            options={{ headerShown: false }}
          />
```

```
        <Stack.Screen
          name="SignIn"
          component={SignIn}
        />
        <Stack.Screen
          name="SignUp"
          component={SignUp}
        />

        <Stack.Screen
          name="HomePage"
          component={HomePage}
        />

        <Stack.Screen name="Cart" component={CartPage}  />
        <Stack.Screen name="Payment" component={PaymentPage}  />

        <Stack.Screen
          name="Profile"
          component={Profile}
        />
        <Stack.Screen
          name="EditProfile"
          component={EditProfile}
        />
         <Stack.Screen
          name="ProductList"
          component={ProductList}
        />

        <Stack.Screen
          name="AdminLogin"
          component={AdminLogin}
        />

        <Stack.Screen name="AdminProductManagement" component={AdminProductManagement} />
        <Stack.Screen name="AddProductForm" component={AddProductForm} />
        <Stack.Screen name="DeleteProductForm" component={DeleteProductForm} />
        <Stack.Screen name="UpdateProductForm" component={UpdateProductForm} />
        <Stack.Screen name="AboutUs" component={AboutUs} />
        <Stack.Screen name="ProductDetail" component={ProductDetail} />
      </Stack.Navigator>
    </NavigationContainer>
    </UserProvider>
  );
}
```

signin.js

```
import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, StyleSheet, Alert } from 'react-native';
```

```
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { useNavigation } from '@react-navigation/native';
import config from './config';
import { useUser } from './UserContext';

export default function SignIn() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const navigation = useNavigation();
  const { setUser } = useUser();
  const [loading, setLoading] = useState(false);

  const handleSignIn = async () => {
    if (!email || !password) {
      Alert.alert('Validation Error', 'Please fill in all fields.');
      return;
    }

    try {
      const response = await axios.post(${config.BASE_URL}/signin, { email, password });
      const { token, userId } = response.data;
      console.log(token, userId);
      await AsyncStorage.setItem('token', token);
      setUser({ id: userId, name: response.data.name, email: response.data.email });
      Alert.alert('Success', 'Sign-in successful!');
      navigation.navigate('HomePage');
    } catch (err) {
      console.error(err);
      const errorMessage = err.response?.data?.error || 'Something went wrong';
      Alert.alert('Error', errorMessage);
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Sign In</Text>
      <TextInput
        style={styles.input}
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
        keyboardType="email-address"
        autoCapitalize="none"
        placeholderTextColor="#D4AF37"
      />
      <TextInput
        style={styles.input}
        placeholder="Password"
        value={password}
        onChangeText={setPassword}
```

```
        secureTextEntry
        placeholderTextColor="#D4AF37"
      />
      <TouchableOpacity style={styles.button} onPress={handleSignIn} disabled={loading}>
        <Text style={styles.buttonText}>{loading ? 'Signing In...' : 'Sign In'}</Text>
      </TouchableOpacity>

      <TouchableOpacity onPress={() => navigation.navigate('SignUp')}>
        <Text style={styles.linkText}>Don't have an account? Sign Up</Text>
      </TouchableOpacity>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 20,
    backgroundColor: 'black', // Black background for bold contrast
  },
  title: {
    fontSize: 32, // Slightly larger title for better emphasis
    fontWeight: 'bold',
    marginBottom: 40,
    color: '#D4AF37', // Golden yellow for the title
    textTransform: 'uppercase', // Uppercase for a striking look
    letterSpacing: 2, // Spacing for a sleek, modern style
  },
  input: {
    width: '100%',
    maxWidth: 350,
    padding: 15,
    borderRadius: 10,
    borderWidth: 2, // Thicker border for better visibility
    borderColor: '#D4AF37', // Golden yellow border for inputs
    marginBottom: 20,
    backgroundColor: '#1C1C1C', // Darker input background for elegance
    color: '#FFF', // White text for the input fields to contrast the background
  },
  button: {
    backgroundColor: '#D4AF37', // Golden yellow button background
    padding: 15,
    borderRadius: 10,
    width: '100%',
    maxWidth: 350,
    alignItems: 'center',
    shadowColor: '#FFD700', // Subtle golden shadow for depth
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.5,
```

```
    shadowRadius: 3,
    elevation: 5, // Shadow elevation for a material design effect
  },
  buttonText: {
    color: '#000', // Black text on the golden button for strong contrast
    fontSize: 18,
    fontWeight: 'bold',
  },
  linkText: {
    color: '#D4AF37', // Golden yellow link text for the link
    marginTop: 20,
    fontSize: 16,
  },
});
```

signup.js

```
import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, StyleSheet, Alert } from 'react-native';
import axios from 'axios';
import { useNavigation } from '@react-navigation/native';
import config from './config';

export default function SignUp() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [confirmPassword, setConfirmPassword] = useState('');
  const navigation = useNavigation();

  const handleSignUp = async () => {
    if (!email || !password || !confirmPassword) {
      Alert.alert('Validation Error', 'Please fill in all fields.');
      return;
    }

    if (password !== confirmPassword) {
      Alert.alert('Validation Error', 'Passwords do not match.');
      return;
    }

    try {
      await axios.post(${config.BASE_URL}/signup, { email, password });
      Alert.alert('Sign Up', 'Sign up successful! Please sign in.');
      navigation.navigate('SignIn'); // Corrected 'signin' to 'SignIn'
    } catch (err) {
      console.error(err);
      const errorMessage = err.response?.data?.error || 'Something went wrong';
      Alert.alert('Error', errorMessage);
    }
  };
```

```
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Sign Up</Text>
      <TextInput
        style={styles.input}
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
        keyboardType="email-address"
        autoCapitalize="none"
        placeholderTextColor="#D4AF37"
      />
      <TextInput
        style={styles.input}
        placeholder="Password"
        value={password}
        onChangeText={setPassword}
        secureTextEntry
        placeholderTextColor="#D4AF37"
      />
      <TextInput
        style={styles.input}
        placeholder="Confirm Password"
        value={confirmPassword}
        onChangeText={setConfirmPassword}
        secureTextEntry
        placeholderTextColor="#D4AF37"
      />
      <TouchableOpacity style={styles.button} onPress={handleSignUp}>
        <Text style={styles.buttonText}>Sign Up</Text>
      </TouchableOpacity>
      <TouchableOpacity onPress={() => navigation.navigate('SignIn')}>
        <Text style={styles.linkText}>Already have an account? Sign In</Text>
      </TouchableOpacity>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 20,
    backgroundColor: 'black', // Golden yellow background
  },
  title: {
    fontSize: 32, // Slightly larger title for better emphasis
    fontWeight: 'bold',
    marginBottom: 40,
    color: '#D4AF37', // Golden yellow for the title
```

```
    textTransform: 'uppercase', // Uppercase for a striking look
    letterSpacing: 2, // Spacing for a sleek, modern style
  },
  input: {
    width: '100%',
    maxWidth: 350,
    padding: 15,
    borderRadius: 10,
    borderWidth: 2, // Thicker border for better visibility
    borderColor: '#D4AF37', // Golden yellow border for inputs
    marginBottom: 20,
    backgroundColor: '#1C1C1C', // Darker input background for elegance
    color: '#FFF', // White text for the input fields to contrast the background
  },
  button: {
    backgroundColor: '#D4AF37', // Golden yellow button background
    padding: 15,
    borderRadius: 10,
    width: '100%',
    maxWidth: 350,
    alignItems: 'center',
    shadowColor: '#FFD700', // Subtle golden shadow for depth
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.5,
    shadowRadius: 3,
    elevation: 5, // Shadow elevation for a material design effect
  },
  buttonText: {
    color: '#000', // Black text on the golden button for strong contrast
    fontSize: 18,
    fontWeight: 'bold',
  },
  linkText: {
    color: '#D4AF37', // Golden yellow link text for the link
    marginTop: 20,
    fontSize: 16,
  },
});
```

Home.js

```
import React, { useEffect, useState } from 'react';
import {
  View,
  Text,
  TextInput,
  TouchableOpacity,
  StyleSheet,
  FlatList,
  Image,
  Alert,
  ActivityIndicator,
```

```
} from 'react-native';
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { useNavigation } from '@react-navigation/native';
import config from './config';
import { Image as RNImage } from 'react-native';

export default function HomePage() {
  const [data, setData] = useState([]);
  const [filteredData, setFilteredData] = useState([]);
  const [searchQuery, setSearchQuery] = useState('');
  const [loading, setLoading] = useState(false);
  const [cart, setCart] = useState([]);
  const navigation = useNavigation();

  useEffect(() => {
    const fetchData = async () => {
      setLoading(true); // Start loading
      try {
        const token = await AsyncStorage.getItem('token');
        const response = await axios.get(${config.BASE_URL}/data, {
          headers: {
            Authorization: Bearer ${token},
          },
        });
        console.log(response.data);
        setData(response.data);
        setFilteredData(response.data);
      } catch (error) {
        console.error(error);
        Alert.alert('Error', 'Failed to fetch data');
      } finally {
        setLoading(false); // Stop loading
      }
    };

    fetchData();
  }, []);

  const handleSearchInputChange = (text) => {
    setSearchQuery(text);

    if (text === '') {
      setFilteredData(data);
    } else {
      const filtered = data.filter(item =>
        item.name.toLowerCase().includes(text.toLowerCase())
      );
      setFilteredData(filtered);
    }
  };
```

```
const addToCart = (item) => {
  setCart([...cart, item]);
  Alert.alert('Success', ${item.name} added to cart!);
};

return (
  <View style={styles.container}>
    <View style={styles.header}>
      <TextInput
        style={styles.searchBar}
        placeholder="Search..."
        value={searchQuery}
        onChangeText={handleSearchInputChange}
        returnKeyType="search"
        placeholderTextColor="#D4AF37"
      />

    </View>

    {loading ? (
      <ActivityIndicator size="large" color="#D4AF37" />
    ) : (
      <FlatList
        data={filteredData}
        renderItem={({ item }) => (
          <View style={styles.itemContainer}>
            <TouchableOpacity onPress={() => navigation.navigate('ProductDetail', { id: item._id })}>
              {item.image ? (
                <Image source={{ uri: item.image }} style={{ width: 100, height: 100 }} />
              ) : (
                <Text>No Image Available</Text>
              )}
            </TouchableOpacity>
            <View style={styles.itemDetails}>
              <Text style={styles.itemTitle}>{item.name}</Text>
              <Text>{item.description}</Text>
              <Text>Price: ${item.price}</Text>
              <TouchableOpacity
                style={styles.addToCartButton}
                onPress={() => addToCart(item)}
              >
                <Text style={styles.addToCartText}>Add to Cart</Text>
              </TouchableOpacity>
            </View>
          </View>
        )}
        keyExtractor={(item) => item._id.toString()}
      />
    )}
```

```
      <View style={styles.menuBar}>
        <TouchableOpacity onPress={() => navigation.navigate('Cart', { cart })}>

          <Text style={styles.menuItem}>View Cart ({cart.length})</Text>
        </TouchableOpacity>

        <TouchableOpacity onPress={() => navigation.navigate('Profile')}>

          <Text style={styles.menuItem}>Profile</Text>
        </TouchableOpacity>

        <TouchableOpacity onPress={() => navigation.navigate('AboutUs')}>

          <Text style={styles.menuItem}>About Us</Text>
        </TouchableOpacity>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: 'black',
    padding: 20,
  },
  header: {
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    marginBottom: 20,
  },
  searchBar: {
    flex: 1,
    height: 40,
    borderWidth: 1,
    borderColor: '#D4AF37',
    borderRadius: 10,
    paddingHorizontal: 10,
    color: '#D4AF37',
  },
  logo: {
    width: 30,
    height: 30,
    marginLeft: 10,
  },
  itemContainer: {
    backgroundColor: '#D4AF37',
    padding: 10,
    borderRadius: 10,
    marginBottom: 10,
```

```
    flexDirection: 'row',
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.3,
    shadowRadius: 3,
    elevation: 5,
  },
  itemImage: {
    width: 100,
    height: 100,
    borderRadius: 10,
  },
  itemDetails: {
    flex: 1,
    marginLeft: 10,
  },
  itemTitle: {
    fontSize: 18,
    fontWeight: 'bold',
    color: 'black',
  },
  addToCartButton: {
    marginTop: 10,
    backgroundColor: 'black',
    paddingVertical: 8,
    paddingHorizontal: 15,
    borderRadius: 5,
  },
  addToCartText: {
    color: '#D4AF37',
    fontWeight: 'bold',
    textAlign: 'center',
  },
  menuBar: {
    flexDirection: 'row',
    justifyContent: 'space-around',
    paddingVertical: 10,
    borderTopWidth: 1,
    borderColor: '#D4AF37',
  },
  menuItem: {
    fontSize: 16,
    color: '#D4AF37',
    textAlign: 'center',
  },
  menuLogo: {
    width: 20,
    height: 20,
    marginBottom: 5,
  },
});
```