



MOHAMMAD SHAHEER KHAN

01-135201-042

SHABBIR ALAM

01-135201-092

OFF Brand Awareness App

Bachelors of Science in Information Technology

Supervisor: Engr. Lala Rukh Ahsan

Department of Computer Science
Bahria University, Islamabad

June 2023

Certificate

We accept the work contained in the report titled "OFF Brand Awareness App", written by Mohammad Shaher & Shabbir Alam as a confirmation to the required standard for the partial fulfillment of the degree of Bachelors of Science in Information Technology.

Approved by...:

Supervisor: Engr. Lala Rukh Ahsan

Internal Examiner:

External Examiner(Title):

Project Coordinator: Dr Usman Hashmi

Head of the Department: Dr. Arif Ur Rehman(Head of Department)

June 23rd,2023

Abstract

This project is a mobile application-based shopping system. The project objective is to deliver an online shopping application onto Android and iOS platforms. This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buy the products in the shop anywhere through the internet by using a mobile device. Thus, the customer will get the service of online shopping and home delivery from his favorite shop. This system can be implemented in any shop in the locality or to multinational branded shops having retail outlet chains. If shops provide an online portal where their customers can enjoy easy shopping from anywhere, the shops won't be losing any more customers to the trending online shops such as Daraz or Olx. Since the application is available on the Smartphone it is easily accessible and always available..

Acknowledgments

In the name of Allah, the most beneficent and the most merciful. We are highly grateful to the One who created us and blessed us with a privileged life. We would like to thank our parents who have always been a pillar of strength and support. We are thankful to our parents who taught us that how hard work, devotion, and working towards your goal with pure intention can take you to places.

Furthermore, we would like to thank and appreciate our supervisor Sir, Jawwad Ijaz who has given us a chance to work on a project that can help in creating value for our beloved country. His professional guidance, time, and effort will always be remembered. Last but not the least, we would like to appreciate our friends who make studying and hard times less challenging. May ALLAH (SWT) guide us to the right path.

Mohammad Shaheer & Shabbir Alam
Islamabad, Pakistan

June, 2023

Table of Contents

Abstract	iii
1 Introduction	1
1.1 Overview	1
1.2 Problem Description	1
1.3 Objectives	2
1.4 Project Scope	2
1.5 Limitations	4
2 Literature Review	5
2.1 Introduction	5
2.2 Background	5
2.3 Purpose	6
2.4 Research question	6
2.5 Delimitation	6
2.6 Definitions	7
2.7 Behavioral Intention	9
2.8 Technology Acceptance Theories and Models	9
2.9 Technology Acceptance Models	9
2.10 Conclusion	10
3 Requirements Specifications	12
3.1 Product Functions	12
3.2 Functional Requirements	13
3.3 6. Non-Functional Requirements	14
3.3.1 Performance Requirements	14
3.3.2 Safety Requirements	15
3.3.3 6. Software Quality Attributes	16
3.4 Hardware Requirements	18

3.5	Software Requirements	18
3.6	Use Cases	19
4	System Design	22
4.1	System Architecture	22
4.2	System Architecture Diagram	24
4.3	Design Constraints	26
4.4	Design Methodology	27
4.5	High Level Design	27
4.5.1	Sequence Diagram	27
4.5.2	Activity Diagram	28
4.5.3	Activity Diagram User	28
4.5.4	Activity Diagram Admin	29
5	System Implementation	33
5.1	Introduction	33
5.2	System Architecture	33
5.2.1	User Interface (UI):	33
5.2.2	State Management	47
5.2.3	Navigation	47
5.2.4	Data Storage	47
5.2.5	Payment processing	47
5.2.6	Security layer	47
5.2.7	Caching and Image Loading	47
5.2.8	Backend Server	48
5.2.9	Load Balancing and Scaling	48
5.3	Tools And Technologies	48
5.3.1	Visual Studio Code	48
5.3.2	Android Studio	48
5.3.3	Java Development kit	48
5.3.4	Node.js	49
5.3.5	Software Development Kit	49
5.3.6	Node Package Manager	49
5.3.7	Expo	50
5.3.8	Yarn	50
5.3.9	Redux	50
5.3.10	Figma	51
5.3.11	QT Designer	51

5.4	Experimental Setup	51
5.4.1	Node Js and NPM	51
5.4.2	Java Development Kit (JDK)	52
5.4.3	Android Studio	52
5.4.4	Install React Native CLI	53
5.4.5	Create OFF Project	54
5.4.6	Debugging and Testing	55
5.4.7	Building and Publishing Your App	55
5.5	Methodology	55
5.5.1	Project Setup	55
5.5.2	UI/UX Design	55
5.5.3	Component Structure	55
5.5.4	State Management	56
5.5.5	Styling	56
5.5.6	Navigation	56
5.5.7	API Integration	56
5.5.8	Testing	56
5.5.9	Error Handling and Logging	56
5.5.10	Performance Optimization	57
5.5.11	Security	57
5.5.12	Deployment	57
5.5.13	App Store Submission	57
5.5.14	Monitoring and Maintenance	57
5.5.15	Documentation	57
5.5.16	Analytics and User Feedback	57
6	Testing	59
6.1	System Testing	59
6.2	Functional testing	59
6.3	Interface testing	60
6.4	Usability testing	61
6.5	Compatibility testing	61
6.6	Performance testing	61
6.7	Testing Strategies	61
6.7.1	Black Box Testing	61
6.7.2	Specification Testing	63
6.7.3	White Box Testing	63

6.8	Testing Performance Test Cases	63
6.9	Testing Usability Test Cases	64
6.10	Test Cases	64
6.10.1	Test Case 1 : Registration	64
6.10.2	Test Case 2 : Log In	65
6.10.3	Test Case 3 : Verification Code	65
6.10.4	Test Case 4 : Saving Snapshot	66
6.10.5	Test Case 6 : Notification	66
6.11	Limitations	68
7	Conclusion	69
7.0.1	Cross-Platform Compatibility	69
7.0.2	User-Centric Design	69
7.0.3	Functionality	69
7.0.4	Security	70
7.0.5	Performance	70
7.0.6	Scalability	70
7.0.7	Maintenance	70
7.0.8	Community and Support	70
7.0.9	Future Development	70
	References	72

List of Figures

2.1	TA MODEL	10
3.1	Full Use Case	19
3.2	Registration	20
3.3	Log In Process	21
4.1	Client-Server Architecture	23
4.2	System Architecture	25
4.3	Sequence Diagram	28
4.4	Activity Diagram User	30
4.5	Activity Diagram Admin	32
5.1	Splash Screen	34
5.2	Registration Screen	35
5.3	Login Screen	36
5.4	Forgot Password Screen	37
5.5	New Password Screen	38
5.6	Home Screen	39
5.7	Menu Screen	40
5.8	Payment Method Screen	41
5.9	Add Card Screen	42
5.10	Oder Confirmation Screen	43
5.11	Orders Screen	44
5.12	Cart Screen	45
5.13	Reviews Screen	46
5.14	Node version	52
5.15	Java version	52
5.16	Android Studio	53
5.17	React Native CLI	53

5.18 Project Creation	54
5.19 Project Creation	54
6.1 Interface Testing	60

List of Tables

3.1	Registration Process	20
3.2	Log In	21
6.1	Test Case: Register	65
6.2	Test Case: Login	65
6.3	Test Case: Gun Detection	66
6.4	Test Case: Saving Snapshot	66
6.5	Test Case: Notification	67

Chapter 1

Introduction

1.1 Overview

The purpose of an e-commerce app named "OFF" would be to provide a platform for customers to browse, purchase, and sell clothing items online. Here are some key purposes and features that an e-commerce app like "OFF" could serve:

1. Online Clothing Shopping.
2. Product Listings.
3. Secure Payments.
4. Personalization and Recommendations.
5. User Reviews and Ratings.

By combining these features, an e-commerce app like "OFF" would aim to provide a seamless and enjoyable online shopping experience for customers, while also offering a platform for clothing brands and sellers to reach a wider audience and grow their businesses.

1.2 Problem Description

Shopping can be a time-consuming and daunting task, with users often having to spend hours browsing through different stores and websites to find the best deals and discounts.

1. Furthermore, keeping track of when brand products are on sale can be challenging, leading to missed opportunities to save money.

2. This can be time-consuming and overwhelming, particularly for those with busy lifestyles.
3. Furthermore, many shoppers are not aware of the latest sales and promotions offered by their favorite brand, resulting in missed opportunities.
4. The OFF app aims to address these problems by providing users with a comprehensive list of brands and products on sale, along with the convenience of placing orders directly from the app.

1.3 Objectives

The objectives of the OFF app are to:

1. Provide users with a comprehensive list of brands and products that are on sale, allowing them to save money and shop smarter.
2. Providing a user-friendly interface that allows users to easily browse and filter sale items based on their preferences, such as size, color, style, and price range.
3. Ensure that users never miss out on great deals by keeping them informed about the latest sales and promotions.
4. Offer a seamless and user-friendly shopping experience by providing a feature to place orders directly from the app.
5. Help users discover new products and brands by curating a diverse selection of items on sale.
6. Build trust and loyalty with users by providing a reliable and secure platform for online shopping.
7. Enhance the convenience of shopping by offering features like easy payment options and fast delivery.
8. Continuously improve the app's functionality and user experience to meet the evolving needs of shoppers.

1.4 Project Scope

OFF's scope as a clothing e-commerce app is comprehensive and promising. By offering a vast selection of fashion products, delivering personalized recommendations, fostering a sense of community, embracing a global approach, and prioritizing user

security and satisfaction, OFF aspires to become a leading player in the competitive world of online fashion retail. As the e-commerce industry continues to flourish, OFF is poised to tap into the growing demand for convenience, variety, and personalized experiences, making it a compelling proposition for both fashion-conscious consumers and clothing brands alike. The product scope of an e-commerce clothing React Native app would typically include the following features:

- 1. User Registration and Authentication:** Allow users to create accounts, log in, and manage their profiles. This feature ensures secure access to the app's features and personalized experiences.
- 2. Product Catalog:** Display a comprehensive catalog of clothing items available for purchase. This includes various categories such as men's, women's, and children's clothing, along with filters and search options for easy browsing.
- 3. Product Details:** Provide detailed information about each clothing item, including product images, descriptions, sizes, colors, prices, and availability. It should also allow users to view customer reviews and ratings.
- 4. Shopping Cart:** Enable users to add selected items to their shopping carts for a seamless shopping experience. The cart should display the selected items, quantities, prices, and allow users to modify or remove items as needed.
- 5. Secure Checkout:** Implement a secure payment gateway that supports various payment methods such as credit/debit cards, mobile wallets, or other digital payment systems. Ensure that user payment information is encrypted and protected.
- 6. Reviews and Ratings:** Allow users to provide feedback on purchased items by leaving reviews and ratings. This helps other users make informed decisions and builds trust in the platform.
- 7. Social Sharing:** Integrate social media sharing options to allow users to share their favorite products or purchases with their friends and followers. This feature can help increase brand visibility and attract new customers.
- 8. Notifications:** Send push notifications or emails to users regarding order updates, promotions, discounts, or other relevant information to keep them engaged and informed.
- 9. Customer Support:** Provide a customer support system, such as a chatbot or messaging feature, to assist users with their queries, complaints, or order-related issues.

10. Admin Panel: Develop an admin panel to manage inventory, track orders, update product details, manage user accounts, and perform other administrative tasks required to run the e-commerce app effectively.

The specific features and functionalities may vary depending on the target audience, business requirements, and any unique selling points or value propositions the "Off Clothing" app intends to offer.

1.5 Limitations

The clothing e-commerce app named "OFF" may offer numerous benefits and convenience to users, but it is not without its limitations. Here are some of the key drawbacks that users and the company itself may face:

1. Limited Physical Interaction.
2. Sizing and Fit Issues
3. Color and Texture Discrepancies
4. Return and Refund Process
5. Security and Privacy Concerns
6. Competition and Pricing
7. Dependence on Technology
8. Delivery Challenges
9. Lack of Personalized Experience

Chapter 2

Literature Review

A literature review of the clothing e-commerce app named OFF would involve an in-depth analysis of existing research and academic literature related to similar apps, the e-commerce industry, and consumer behavior in the context of fashion and clothing shopping. The objective would be to gain insights into the various aspects that influence the success and effectiveness of such an app in the market.

2.1 Introduction

This chapter specifies the background of the research and presents the problem realized, and motivations behind the study. Subsequently, the purpose of the work, research question, definitions of some key concepts, and delimitations will be presented.

2.2 Background

Nowadays, accepting and using modern technologies is common practice, and people are increasingly willing to adopt new technology in their daily lives, making technology, now more than ever a part of our everyday activities (Islam, Low Hasan, 2013). Over the last two decades, mobile devices have brought a deep impact on the human's daily life. The adoption of mobile devices grows quickly all over the world, and Europe has the highest mobile device adoption rate in the world (Ecommerce-news. EU, 2015).

Moreover, mobile applications are third-party software that can be installed on mobile devices (Grotnes, 2009). Users can install different kinds of mobile applications, such as games, music, shopping, bank payment applications, and so forth, which are

delivered by third-party providers (Islam, Islam Mazumder, 2010). The installation of these applications, the functions of the mobile devices are expanded. The number of mobile apps has been rising, and this rise contributed to the increasing range of consumer needs that are being served by mobile apps (Kim, Yoon Han, 2014).

Because of the disadvantages of websites due to their limited functionality, many companies, especially fashion retail companies provide mobile shopping apps to the customers (Magrath McCormick, 2013). In order to take advantages of mobile shopping, fashion companies gradually invest into creating their mobile shopping apps because these apps foster discussion regarding the products; enable the consumers to recommend products to friends via social networks; enable the users to receive instant push notifications regarding special offers; obtain personalized information, in other words further enhance their shopping experiences (Magrath McCormick, 2013). Compared to mobile websites, mobile applications are preferred by consumers primarily because they are perceived as more convenient, faster and easier to browse (Mobile Apps: What Consumers Really Need and Want, 2016). Mobile shopping apps usage is growing faster comparing to most other categories of mobile apps (Khalaf, 2015).

2.3 Purpose

This paper aims to identify the factors that affect the behavioral intention to use m-shopping fashion apps. More explicitly, the external factors that directly determine the behavioral intention to use m-shopping fashion apps in Sweden from a consumer perspective.

2.4 Research question

What are the factors that affect the user's behavioral intention to use m-shopping fashion apps in the World?

2.5 Delimitation

In order to limit the research, this study will not focus on the factors of individual difference variables, such as age, gender, and experience which moderate the effects of factors that affect behavioral intention to use a technology. Moreover, as the study will investigate the predictors of behavioral intention from a consumer point of view, the study will not focus on business-to-business m-shopping and the non-

users of shopping fashion apps. The consumers in our context are the users of mobile shopping fashion apps.

2.6 Definitions

The terms that are used throughout this study are presented, in order to clarify for the readers the concepts and reduce the risk of misunderstandings.

Term	Definition
Electronic commerce (e-commerce)	E-commerce refers to digitally enabled commercial transaction between and among organizations and individuals.
Mobile commerce (m-commerce)	Mobile commerce refers to the use of mobile devices (smartphones, tablets) to enable online transactions
Mobile shopping (m-shopping)	M-shopping is defined as the use of the wireless Internet service for shopping activities via a mobile device
Mobile applications (mobile apps)	Mobile applications are software programs that can perform certain tasks for the users, and can be installed on the m-device
Mobile shopping fashion apps (m-shopping fashion apps)	The term m-shopping fashion apps will be used, by that that we mean mobile shopping applications that can be installed on the mobile device and which enable the user of the app to purchase and browse fashion goods on mobile devices.
Technology acceptance	Technology acceptance is defined as an individual's psychological state with his or her voluntary use of a technology
Consumer Acceptance and Use of Information Technology (UTAUT2)	Consumer Acceptance and Use of Information Technology (UTAUT2) is a model and theory that integrates nine theories about user acceptance and user behavior.
Behavioral Intention	Behavioral intention is the individual willingness to use and continue to use the technology.

2.7 Behavioral Intention

Behavioral intention has been defined in previous technology acceptance studies as the individual willingness to use a technology system (Venkatesh et al., 2012; Venkatesh et al., 2003; Davis et al., 1989). In accordance with Venkatesh et al. (2012) in our study we define behavioral intention as the individual willingness to use and continue to use a technology system, where the individuals are the users of technology, and the context is m-shopping fashion apps.

2.8 Technology Acceptance Theories and Models

The understanding of user's acceptance of technology, more explicitly why users of technology tend to accept or reject a certain technology, and what drives people to accept a technology has been a challenging issue (Davis, Bagozzi, Warshaw, 1989). Today because of the rapid and constant increase in new technology systems, there is a call for research to understand user's acceptance of the latest technology such as the recent sales technology, the mobile platform with respect to a certain type of goods, and include factors related to the vendors (Chang, 2012). Various theories and models have been developed to explain and measure the intention to use technology, the technology acceptance theories and models have in common that they are based on the assumption that an intention to use technology will result in actual usage of the technology, however different theories state different factors that affect the behavioral intention to use a certain technology (Venkatesh et al., 2003). Moreover, this study will use the Consumer Acceptance and Use of Information Technology (UTAUT 2) to investigate the factors that affect the behavioral intention to use m-shopping fashion apps as the UTAUT 2 was mainly developed from four previous technology acceptance models. These models are briefly introduced in the following sections in order to enrich the readers understanding of the UTAUT 2 model. Likewise, the UTAUT 2 was developed specially for the consumer context of technology acceptance and empirically validated to outperform the previous technology acceptance models (Venkatesh et al. 2012).

2.9 Technology Acceptance Models

The Technology Acceptance Model (TAM), first introduced by Davis (1989) is an information systems theory and models the users' use and acceptance of technology. Davis, Bagozzi, and Warshaw, in 1989 revised the TPB and TRA, where the factor Perceived Usefulness (PU) and Perceived Ease of Use (PEU) were found the two

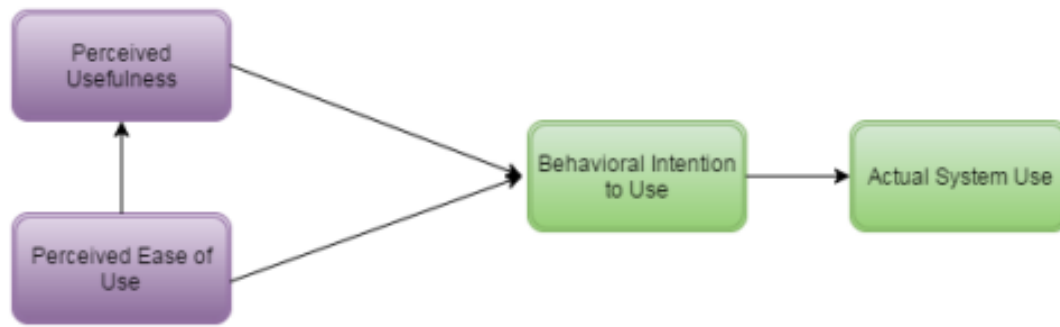


Figure 2.1: TA MODEL

most important factors that predict the intention to use technology. PU is defined as “the prospective user’s subjective probability that using a specific application system will increase his or her jobs performance” and PEOU is defined as “the degree to which the the prospective user expects the target system to be free of effort” (Davis et al., 1989, p. 985). Furthermore one of their major conclusions is that the use of technology can be soundly predicted from the users’ intentions, which is consistent with the TRA and TBP, where users’ behavioral intention to perform a certain action is the main determinant of actual behavior.

2.10 Conclusion

Most of the research work done on the acceptance of mobile commerce and mobile shopping was done by applying the TAM model, and expanding it by two or more factors, that affect the intention to use mobile shopping. However being a recent development, no previous research was using the UTAUT 2 model. The UTAUT 2 model addresses the consumer technology acceptance and as m-fashion shopping apps are in this category this makes UTAUT 2 model especially beneficial for our study. The UTAUT 2 model consists of seven factors that determine the user intention to use technology. Moreover, the UTAUT 2 model was empirically validated and proven to outperform the TAM, TPB, and other technology acceptance models (Venkatesh et al., 2003), while due to the three additional factors added to the UTAUT 2, “substantial improvement in the variance explained in the behavioral intention” was achieved (Venkatesh et al., 2012, p. 157). This indicates that UTAUT 2 being a more complete technology acceptance model is demanded for the successful investigation of the factors that affect the intention to use mshopping fashion apps. Therefore, the UTAUT 2 is valuable for investigating the factors that affect the behavioral intention to use m- m-shopping fashion apps in Sweden and will be

used as the foundation for the proposed research model of this study.

Chapter 3

Requirements Specifications

3.1 Product Functions

OFF is an innovative clothing e-commerce app that aims to revolutionize the way people shop for apparel online. The proposed system is designed to provide users with a seamless and enjoyable shopping experience, combining cutting-edge technology, user-friendly features, and a vast selection of trendy fashion items. The app's user interface is sleek and intuitive, allowing users to navigate effortlessly through various categories and collections.

One of the key features of OFF is its personalized recommendation engine. Leveraging advanced machine learning algorithms, the app analyzes users' browsing and purchase history, as well as their preferences and style choices, to offer highly tailored product suggestions. This feature not only enhances the user's shopping experience but also encourages them to explore new styles they might have never considered before.

Moreover, OFF is dedicated to providing users with the most accurate sizing information possible. It includes a virtual fitting room, which utilizes augmented reality technology to allow users to virtually try on clothes. This helps eliminate sizing concerns and increases the confidence of customers in making purchases, resulting in reduced returns and higher customer satisfaction.

The app also integrates a social networking component, allowing users to share their favorite outfits and styles with friends and followers. They can also follow fashion influencers and celebrities, gaining inspiration from their looks and accessing exclusive collections curated by these influencers.

To further enhance customer engagement, OFF implements gamification elements within the app. Users can earn loyalty points, badges, and rewards by participating in various activities such as leaving reviews, referring friends, or participating in style challenges. This not only fosters a sense of community but also incentivizes users to remain active on the platform.

Security and privacy are top priorities for OFF. The platform implements robust encryption protocols to safeguard users' personal and financial information. It also provides a secure payment gateway, supporting multiple payment options to ensure convenient and safe transactions.

Additionally, OFF embraces sustainability and ethical fashion practices. The app features a designated section for eco-friendly and ethically produced clothing, promoting brands that prioritize social and environmental

3.2 Functional Requirements

1. User Registration and Authentication:

- User registration with email, social media accounts, or phone number
- Secure authentication mechanisms (password, biometrics, etc.)
- Profile management and customization options

2. Product Catalog:

- Categorized listing of clothing products (e.g., Men, Women, Kids, Accessories)
- Detailed product pages with images, descriptions, pricing, and specifications
- Filtering and sorting options to refine search results.

3. Shopping Cart and Checkout:

- Shopping cart functionality to add/remove items
- Multiple payment options (credit/debit cards, digital wallets, etc.)
- Shipping address management and order tracking
- Order history and invoice generation.

4. Admin Panel:

- Backend system for managing products, inventory, and orders
- User management, including permissions and roles

- Content management system for updating app content and promotions

5. Social Integration:

- Social media sharing options for products
- Integration with social media platforms to enhance user engagement
- User comments and discussions on product pages

3.3 6. Non-Functional Requirements

3.3.1 Performance Requirements

To develop a high-performance e-commerce app named "Off Clothing," you need to consider several performance requirements to ensure a smooth and satisfying user experience. Here are some key performance requirements for your e-commerce app:

1. **Responsiveness:** The app should respond quickly to user interactions, such as browsing product listings, adding items to the cart, or completing purchases. Aim for near-instantaneous response times to prevent user frustration and cart abandonment.
2. **Page Load Speed:** Minimize the loading time for product pages, category listings, and search results. Optimizing image sizes, leveraging caching mechanisms, and implementing efficient database queries can help reduce page load times.
3. **Scalability:** Design the app to handle high traffic loads, especially during peak shopping seasons or promotional campaigns. Consider using cloud-based infrastructure and auto-scaling techniques to ensure the app can handle increased user demand without performance degradation.
4. **Search Performance:** Implement a fast and accurate search functionality to enable users to find products quickly. Utilize indexing, caching, and efficient search algorithms to provide relevant results in real-time.
5. **Checkout Process:** Optimize the checkout process to minimize the number of steps required and simplify the form filling. Ensure that payment processing is secure and transactions are completed swiftly to reduce cart abandonment.
6. **Database Performance:** Optimize database queries and ensure efficient data retrieval and storage. Use proper indexing, caching, and database tuning techniques to minimize response times for fetching product information and user data.

7. Mobile Performance: Optimize the app for mobile devices to ensure smooth performance across various screen sizes and resolutions. Consider mobile-specific performance factors, such as battery usage, network constraints, and limited processing power.

8. Error Handling: Implement robust error handling and provide helpful error messages to guide users when they encounter issues. Handle errors gracefully and provide feedback to users, so they understand the problem and can take appropriate actions.

9. Analytics and Monitoring: Incorporate analytics and monitoring tools to track app performance, identify bottlenecks, and proactively address any performance issues. Monitor key metrics like response times, conversion rates, and user engagement to continuously improve performance.

10. Security: Implement proper security measures to protect user data and prevent unauthorized access. Use secure protocols for data transmission, encrypt sensitive information, and follow industry best practices for authentication and authorization

3.3.2 Safety Requirements

To ensure the safety and security of an e-commerce app named "Off Clothing," several safety requirements should be considered. Here are some important safety requirements for an e-commerce app:

1. Secure Authentication:

- Implement a robust authentication mechanism, such as username/password or two-factor authentication, to prevent unauthorized access.
- Enforce strong password policies and provide password reset options.

2. Secure Connection:

- Use SSL/TLS encryption to establish a secure connection between the app and the server. This ensures that all data transmitted between the user and the app is encrypted and protected from interception.

3. Payment Security:

- Comply with the Payment Card Industry Data Security Standard (PCI DSS) if you process credit card payments. This standard ensures the secure handling of payment card information.

- Implement secure payment gateways that support encryption and tokenization to protect sensitive payment data.

4. Secure Data Storage:

- Encrypt sensitive user data, such as passwords and payment information, when stored on the server or in databases.
- Follow best practices for secure data storage, including regular backups and access controls to prevent unauthorized access to the stored data.

5. User Privacy:

- Clearly state the app's privacy policy and obtain user consent for collecting and processing personal information.
- Implement data protection measures, such as anonymization or pseudonymization, when handling user data.

3.3.3 6. Software Quality Attributes

Software quality attributes refer to the characteristics and properties of a software system that define its overall quality and performance. Here are some important software quality attributes for an e-commerce app named "Off Clothing" along with their corresponding requirements.

7. Reliability:

- The app should be available and accessible to users at all times.
- It should handle high-traffic loads and concurrent user sessions without crashing or experiencing significant slowdowns.
- Transactions and order processing should be accurate and consistent.

8. Performance:

- The app should have fast response times for user interactions, such as browsing products, adding items to the cart, and checking out.
- Pages and images should load quickly to provide a smooth user experience.
- The app should be able to handle a large number of concurrent users during peak hours.

9. Security:

- User authentication and authorization mechanisms should be in place to protect user accounts and personal information.
- Payment processing should be secure and comply with industry standards (e.g., PCI DSS).
- Secure connections (HTTPS) should be used for all communications involving sensitive data.

10. Usability:

- The app should have an intuitive and user-friendly interface that is easy to navigate.
- Product search and filtering should be provided to help users find desired items quickly.
- The checkout process should be simple and streamlined, minimizing steps and reducing user effort.

11. Scalability:

- The app should be designed to scale horizontally to accommodate increased user demand and growth.
- The underlying infrastructure should support the addition of more servers and resources as needed.
- Database and caching mechanisms should be scalable to handle increasing amounts of data.

12. Maintainability:

- The app's codebase should be modular and well-structured, allowing for easy maintenance and future enhancements.
- Proper documentation should be provided to aid in understanding the system's architecture, components, and dependencies.
- Logging and error handling mechanisms should be in place to facilitate debugging and troubleshooting.

13. Compatibility:

- The app should be compatible with multiple platforms and devices, such as web browsers, smartphones, and tablets.

- It should support various operating systems and versions to reach a wider user base.
- Responsiveness and adaptability to different screen sizes and resolutions are essential.

14. Testability:

- The app should be designed with testability in mind, allowing for effective unit testing, integration testing, and end-to-end testing.
- Test cases should be created to cover different functionalities and scenarios, ensuring proper functionality and minimizing the introduction of bugs.
- These are just some of the software quality attributes and corresponding requirements for an e-commerce app like "Off Clothing." It's important to tailor these attributes based on the specific needs and priorities of the application and its users

3.4 Hardware Requirements

Mobile Devices: E-commerce apps are often designed to be compatible with smartphones and tablets, utilizing touchscreens as the primary input method. The app can be accessed through mobile platforms such as iOS or Android.

3.5 Software Requirements

Login/Registration Screen: This is the first screen users encounter when they open the app. It allows new users to register for an account or existing users to log in using their credentials or social media accounts.

Home Screen: After logging in, users are presented with the home screen, which showcases featured products, promotions, and personalized recommendations based on their browsing history or preferences. It may also include categories or a search bar for easy navigation.

Product Listing Screen: When users tap on a category or perform a search, they are taken to the product listing screen. Here, they can view a grid or list of products with their images, names, prices, and basic details. Users can scroll through the list and tap on a product to view more details.

Product Detail Screen: This screen displays detailed information about a selected product, including high-quality images, descriptions, available sizes, colors, customer

reviews, and the option to add the item to the cart or Wishlist. Users can also see related products or recommended items on this screen.

Visual Studio Code: We used VS Code for the development of our web application, it is a code editor with support for development operations like debugging, task running, and version control.

Adobe XD: We used Adobe XD for designing the user interface of our web application.

QT Designer: Qt Designer is a tool that provides us to create GUI of client side of our web application productively and efficiently.

3.6 Use Cases

Note: The use cases may change during SDLC

The use case of the whole system will be explained below:

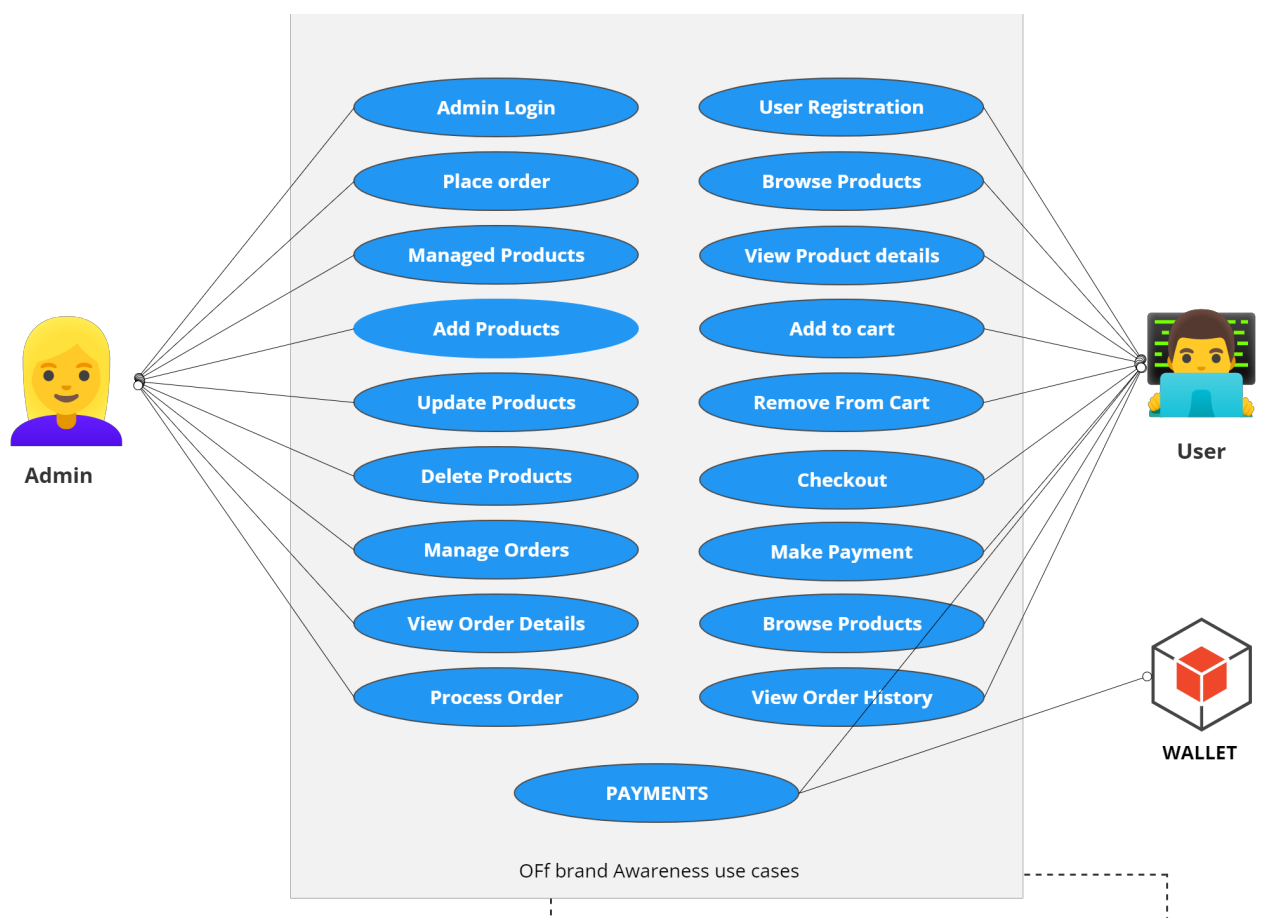


Figure 3.1: Full Use Case

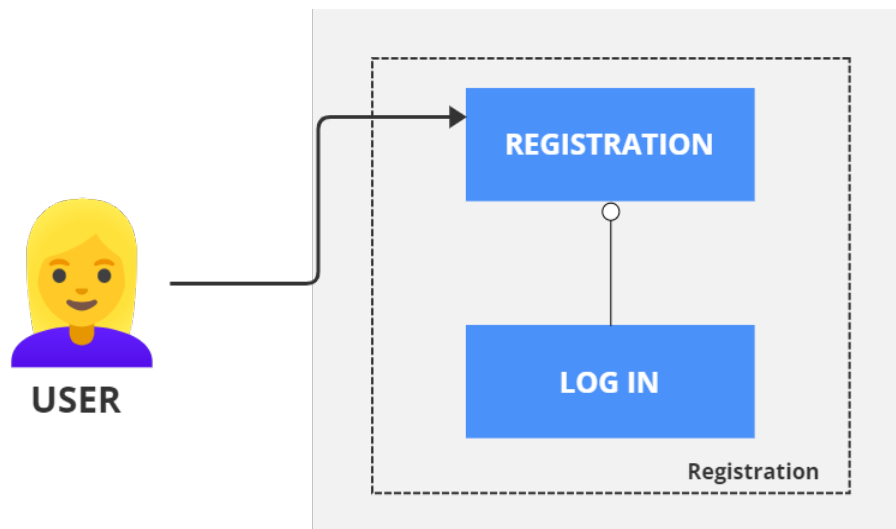


Figure 3.2: Registration

Registration Process	
Actor	User
Description	Use case specifies registration process
Main Success factor	User must be able to register and add in login details
Pre-Condition	The application is in running state
Post-Condition	User has been successfully registered

Table 3.1: Registration Process

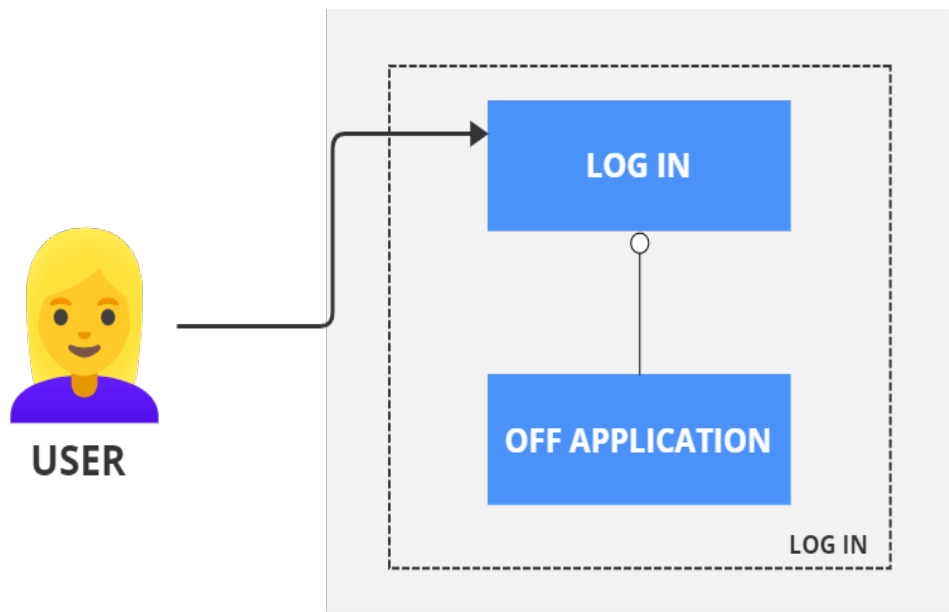


Figure 3.3: Log In Process

Log In	
Actor	User
Description	This use case specifies Log In procedure
Main Success factor	The user must be able to Log In and enter specified details
Pre-Condition	The User should be registered
Post-Condition	User has been successfully Logged In

Table 3.2: Log In

Chapter 4

System Design

In this chapter of System Design, which is all about defining component, interfaces and data to reach to our specified requirements, we have a deeper look into the development phases of our gun detection system. This chapter will discuss the system architecture and provide details of the design methodology, constraints, models, interaction between the system and the user.

4.1 System Architecture

OFF's system architecture is designed to provide a reliable, secure, and enjoyable shopping experience for customers while empowering sellers to manage their products efficiently. By incorporating cutting-edge technologies and best practices, OFF aims to stay competitive in the ever-evolving landscape of clothing e-commerce. Here's a brief overview of the system architecture:

1. User Interface (UI) Layer:

- This is the front-end layer that users interact with.
- It includes the website or mobile app where customers can browse products, add items to their cart, and make purchases.
- The UI layer is responsible for presenting the user interface, handling user inputs, and displaying product information and images

2. Database Layer:

- Stores and manages data related to products, user profiles, orders, and more.
- Utilizes a relational database management system (RDBMS) or NoSQL databases,

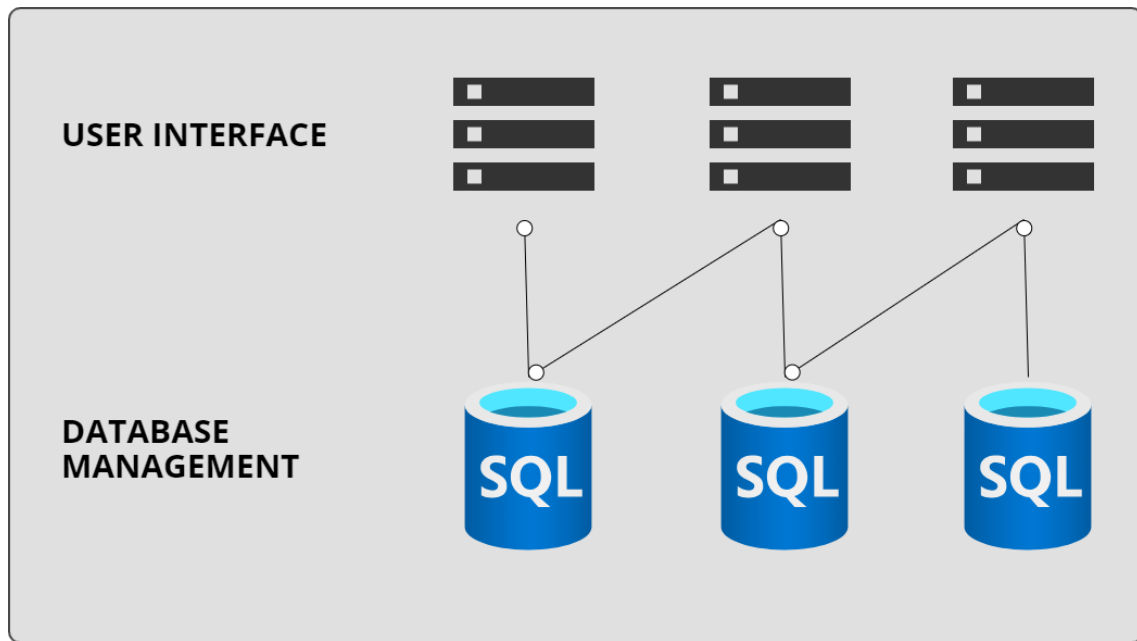


Figure 4.1: Client-Server Architecture

depending on scalability and data requirements.

- Ensures data integrity and supports various data retrieval and storage operations.

3. Authentication and Authorization Layer:

- Manages user authentication, identity verification, and access control.
- Controls which users have permission to perform certain actions (e.g., admin privileges for managing products).
- Often incorporates security measures like password hashing and token-based authentication.

4. Payment Gateway Integration:

- Interfaces with third-party payment service providers to facilitate secure payment processing.
- Handles transactions, payment verification, and order confirmation.
- Ensures the security of financial transactions and user payment information.

5. Security Layer:

- Enforces security measures to protect against common threats, such as SQL injection, cross-site scripting (XSS), and data breaches.

- Implements encryption, firewalls, and regular security audits.

6. Scalability and Load Balancing:

- Utilizes load balancers and auto-scaling mechanisms to handle increased traffic during peak periods.
- Ensures that the application can scale horizontally to accommodate growing user demands.

4.2 System Architecture Diagram

An architectural diagram is a diagram of a system that is used to deduce the overview of the software system. It is an important tool as it provides an overall view of the physical application of the software system. Our developed system's overall architectural overview is illustrated in the diagram below:

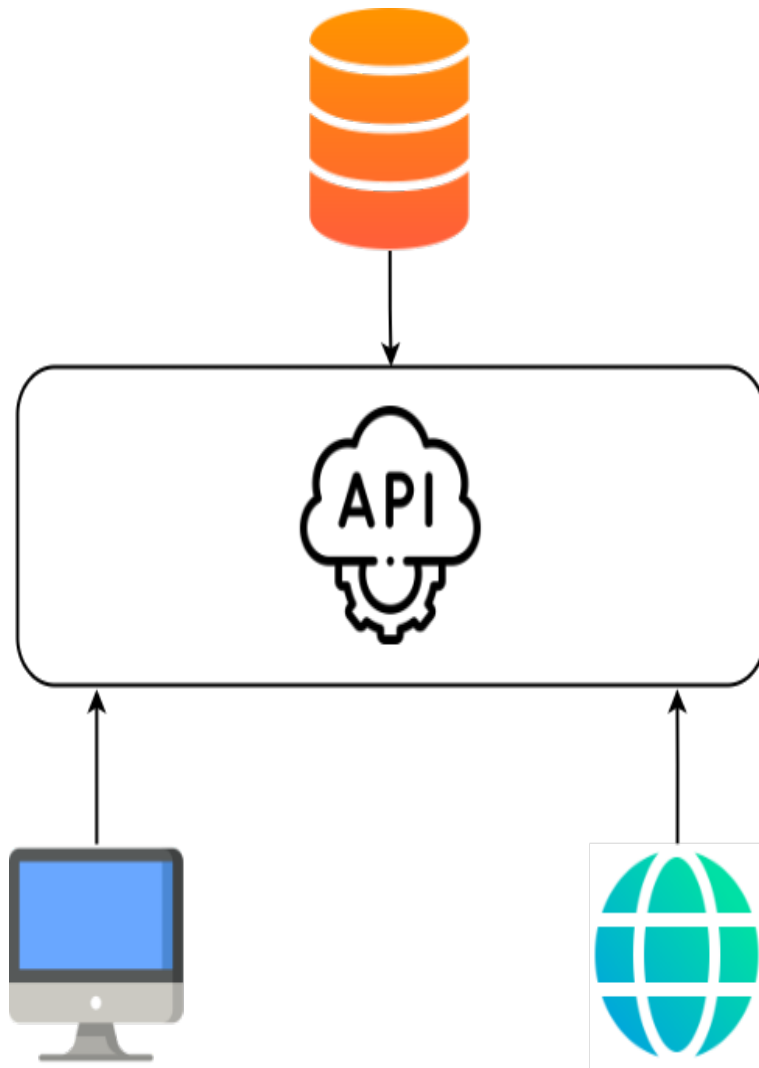


Figure 4.2: System Architecture

4.3 Design Constraints

Design constraints play a crucial role in shaping the functionality and user experience of a clothing e-commerce app like "OFF." These constraints are essential considerations that guide the development process to ensure the app meets specific requirements and limitations. Let's explore some key design constraints for the app:

1. User Interface (UI) and User Experience (UX):

One of the primary design constraints for the "OFF" app is to create an intuitive, user-friendly interface that caters to users of all ages and technical proficiency. The app should have a visually appealing and responsive design, enabling seamless navigation, product exploration, and an easy checkout process. Striking the right balance between aesthetics and usability is crucial to engage and retain users.

2. Mobile Responsiveness:

In the era of smartphones and tablets, it is imperative for "OFF" to be fully responsive across various devices and screen sizes. The app should adapt its layout and functionality to provide a consistent and optimized experience on both mobile and desktop platforms. Ensuring a mobile-friendly design is critical to reach a broader audience and enhance user accessibility.

3. Product Catalog Management:

A critical constraint is to efficiently manage the vast catalog of clothing items available on the app. The design must include robust product categorization, filters, and search functionalities to help users find their desired products quickly. Additionally, the app should support real-time inventory updates to prevent discrepancies and offer accurate product availability information.

4. Security and Privacy:

Given that users will be sharing personal information and payment details, stringent security measures must be implemented. The app should incorporate robust encryption protocols, secure payment gateways, and strict privacy policies to safeguard user data from potential breaches and cyber threats.

5. Integration with Social Media:

To leverage the power of social media for marketing and user engagement, the app should seamlessly integrate with popular platforms like Facebook, Instagram, and Pinterest. This integration allows users to share their favorite products and purchases, promoting the app organically and expanding its user base.

In conclusion, the successful design of the "OFF" clothing e-commerce app heavily relies on effectively managing these constraints. Balancing functionality, security, scalability, and user experience will contribute to creating a compelling platform that attracts and retains customers in the highly competitive online clothing market.

4.4 Design Methodology

In design methodology, we define the procedures and techniques which will help us in designing our system. As we have to develop a Native application, this is why we needed some Javascript knowledge. The design process always requires iterations and improvements on each level in order to minimize errors in the final product due to which we brainstormed and encouraged new ideas and collaborative thinking to work through each idea and arrive at the best and optimal solution.

4.5 High Level Design

High level design provides us a detailed overview of how different functionalities and features coordinate and interact with each other. This section highlights the basic workflow along with alternative paths just in case a failure occurs. The following diagrams are used to demonstrate how different functions of this app will work.

4.5.1 Sequence Diagram

The "OFF" clothing e-commerce app allows users to browse and purchase clothing items online. Let's consider a simple sequence diagram illustrating the process of a user searching for a product, adding it to the cart, and completing the purchase:

User Interaction: The sequence begins with the user interacting with the "OFF" app. They open the app on their device and land on the home screen, which displays various clothing categories and featured products.

Searching for a Product: The user decides to search for a specific product. They enter the search query in the app's search bar and click the "Search" button. The app sends the search request to the server.

Server Processing: The server receives the search request and processes it. It queries the database to find matching products based on the search keywords and returns the search results to the app.

Displaying Search Results: The app receives the search results from the server

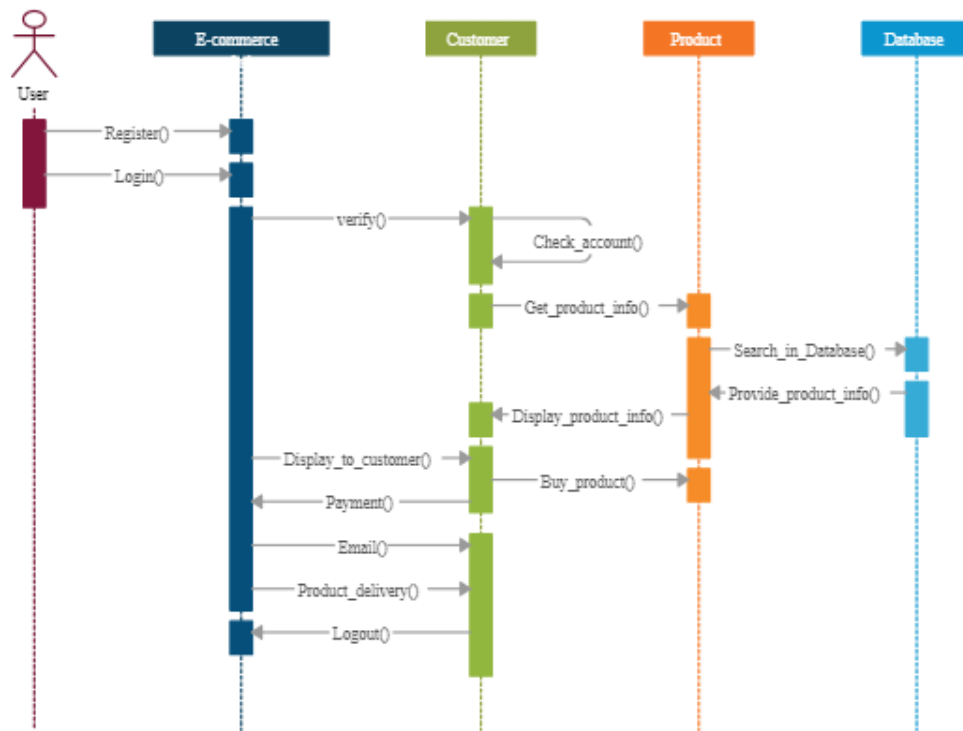


Figure 4.3: Sequence Diagram

and displays them on the user's screen. The user can now browse through the list of products that match their search criteria.

Adding a Product to Cart: The user selects a product from the search results and clicks on it to view more details. After reviewing the product information, the user decides to add it to their shopping cart. They click the "Add to Cart" button.

4.5.2 Activity Diagram

OFF is a clothing e-commerce app that allows users to browse and purchase a wide variety of clothing items conveniently from their mobile devices. Here, I'll describe an activity diagram of the main functionalities and processes involved in the OFF app.

4.5.3 Activity Diagram User

An activity diagram for an e-commerce store from the perspective of a user is a visual representation that illustrates the various activities and interactions a user engages in while using the online store. Here's a description that explains the activity

diagram step by step:

- 1. Start:** The diagram begins with a "Start" node, indicating the initiation of the user's interaction with the e-commerce store.
- 2. Login/Register:** The user is presented with options to either log in to their existing account or register as a new user if they don't have an account. This decision point is represented by a diamond-shaped "Decision" node.
 - If the user already has an account, they proceed to log in.
 - If the user chooses to register, they enter their registration details.
- 3. Browse Products:** After logging in or registering, the user starts browsing products. They can search for products, filter by categories, or simply explore the store's offerings.
- 4. View Product Details:** When the user clicks on a product, they can view its details, including price, description, reviews, and images.
- 5. Add to Cart:** If the user decides to purchase a product, they have the option to add it to their shopping cart. They can add multiple items to the cart.
- 6. Payment:** The user enters their payment details, such as credit card information or selects other payment options like PayPal. The payment process may involve additional steps, such as verification.
- 7. Order Confirmation:** The system displays an order confirmation message and provides an order summary.
- 8. End:** The diagram ends with an "End" node, signifying the completion of the user's shopping activities.

4.5.4 Activity Diagram Admin

An activity diagram for an e-commerce store from the perspective of an administrator (admin) provides a visual representation of the various activities and processes that an admin can perform within the system. Here's a description of the key activities depicted in such an activity diagram:

- 1. Start:** The diagram typically starts with a "Start" node, indicating the beginning of the admin's activities in the e-commerce system.
- 2. Login:** The admin begins by logging into the admin panel of the e-commerce store. This involves providing valid admin credentials (username and password) to

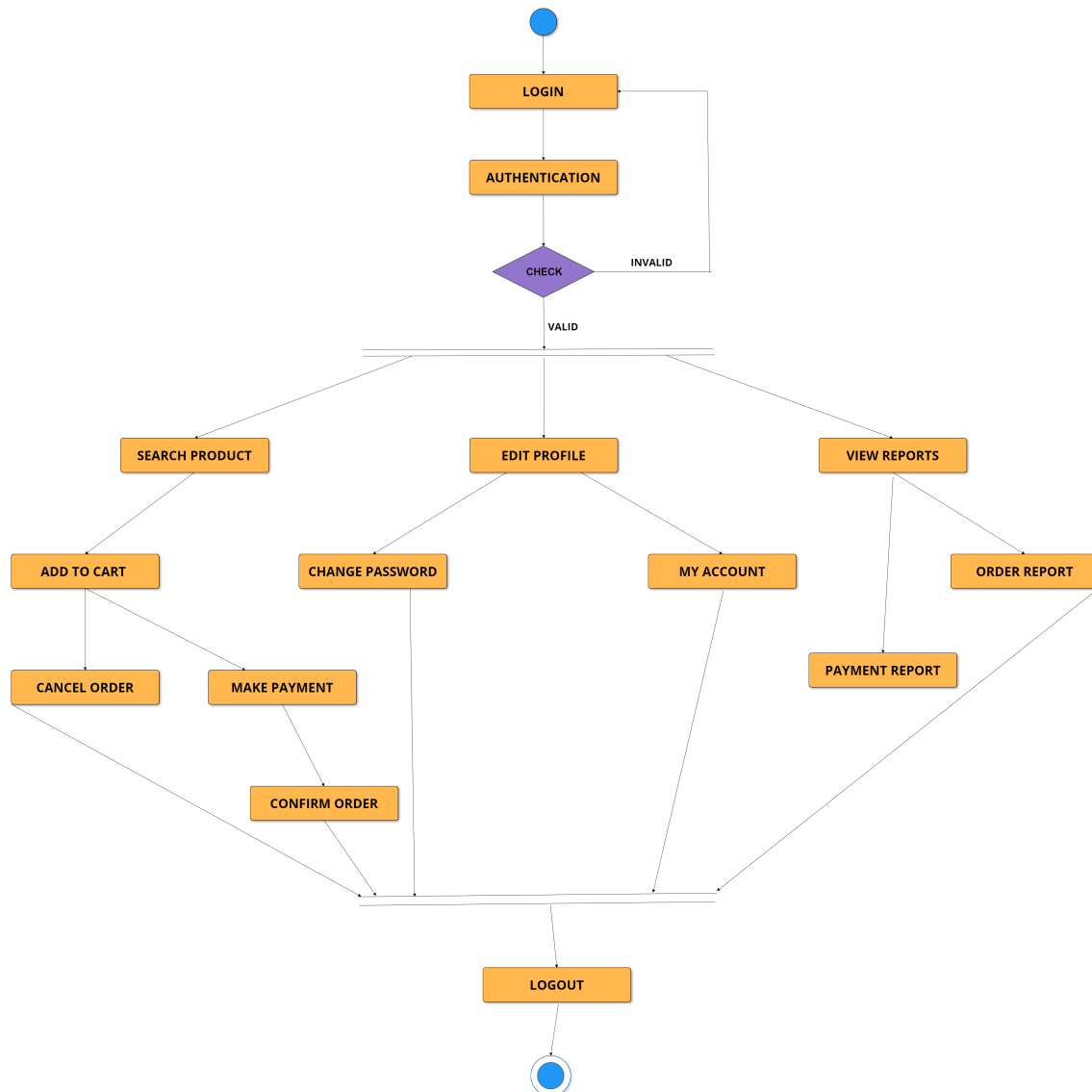


Figure 4.4: Activity Diagram User

access the administrative interface.

3. View Dashboard: After a successful login, the admin is directed to the admin dashboard. The dashboard provides an overview of important metrics and activities within the system, such as sales data, order processing status, and product management options.

4. Manage Products: Admins have the authority to manage products within the e-commerce store. This includes activities such as:

- **Add Product:** Admins can add new products to the catalog. This involves providing product details such as name, description, price, and images.
- **Update Product:** Admins can edit and update existing product information.

This might include modifying prices, descriptions, or availability.

- **Delete Product:** Admins can remove products from the catalog when necessary.

5. Manage Orders: Admins oversee the order processing and fulfillment. Activities related to managing orders include:

- **View Orders:** Admins can view a list of incoming orders, each with details like the order number, customer information, and order status.
- **Process Orders:** Admins can take actions to fulfill orders, which may include marking orders as shipped or canceled.

6. Manage Customers: Admins can interact with customer data, including:

- **View Customer Information:** Admins can access customer profiles and view their details, such as contact information and order history.
- **Manage Customer Accounts:** Admins can perform actions like updating customer account information or resetting passwords.

7. Generate Reports: Admins can generate various reports to gain insights into the e-commerce store's performance. These reports may include sales reports, inventory reports, and customer analytics.

8. Log Out: When the admin has completed their tasks, they can log out of the admin panel to secure their session.

9. End: The "End" node represents the conclusion of the admin's activities within the system.

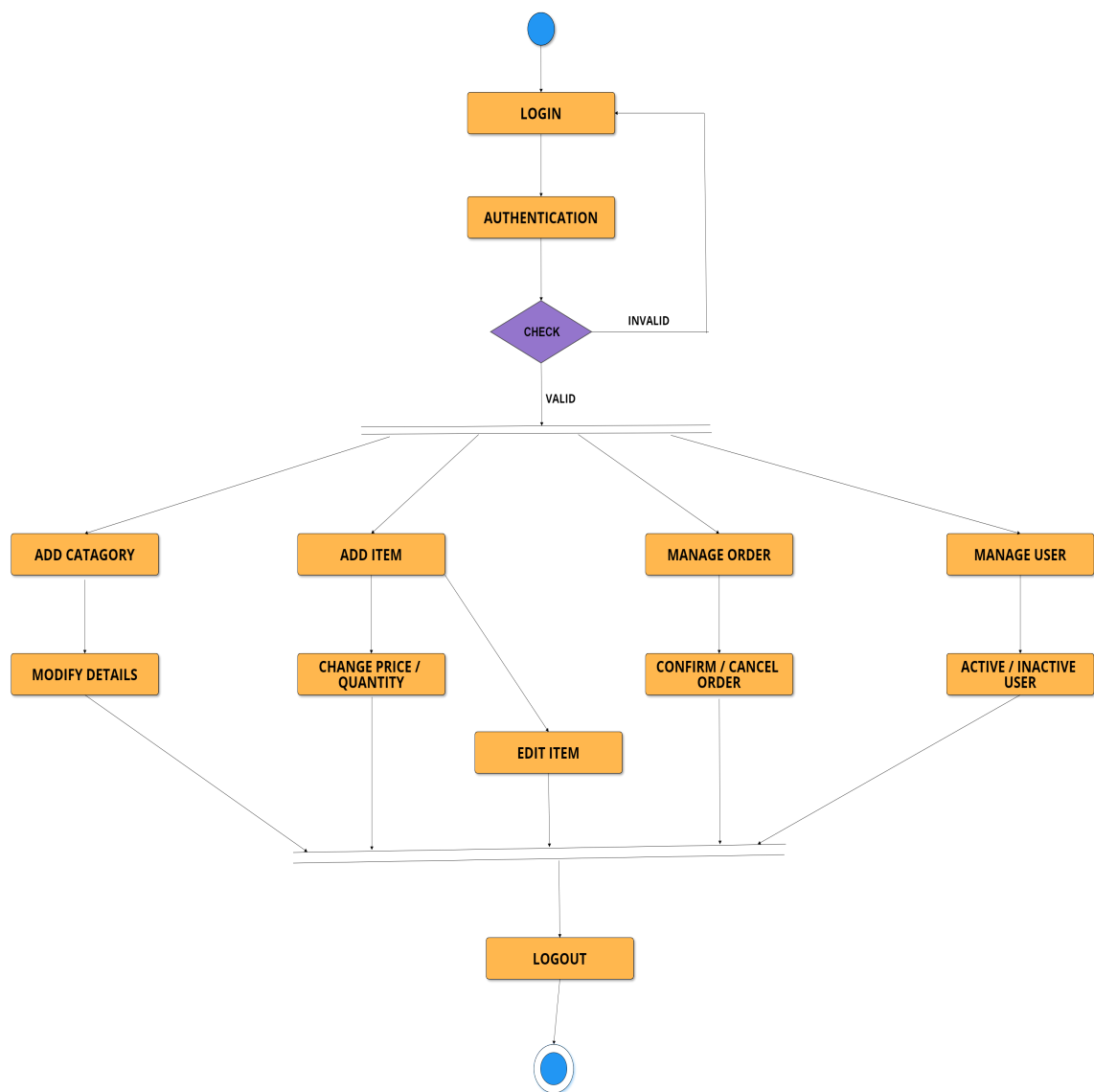


Figure 4.5: Activity Diagram Admin

Chapter 5

System Implementation

5.1 Introduction

The Implementation requires the translation of the design into programs that work successfully. In the implementation phase, the system is installed, all the processes are completed, and the documentation is provided to the user. Once this phase is completed, the application will be in static production, when the system enters static production, It will verify to ensure that all the requirements that we have planned are met and that we have obtained an acceptable result.

5.2 System Architecture

The system architecture of an "Off Clothing" online shopping React Native application involves various components and layers that work together to provide a functional and efficient mobile shopping experience. Below is a high-level description of the system architecture:

5.2.1 User Interface (UI):

- The UI layer represents the mobile app's user interface components and screens.
- It is developed using React Native, allowing for a cross-platform user experience on both iOS and Android devices.
- UI components include product listings, product details, shopping cart, user registration, login, and checkout screens.

The description of the User Interface (UI) is as follows:

1. Splash Screen

A Splash Screen is a screen or image that briefly appears when a mobile app is launched, providing a visually appealing and branded entry point for users. It serves as a loading screen while the app initializes, helping to improve user experience by giving the impression of a faster startup time. In React Native, developers can implement a splash screen by customizing the native splash screen for both Android and iOS platforms. This allows them to display an image, logo, or animation that represents the app's identity, engaging users while the app loads its initial content. Splash screens are commonly used to create a seamless and professional introduction to the mobile app.

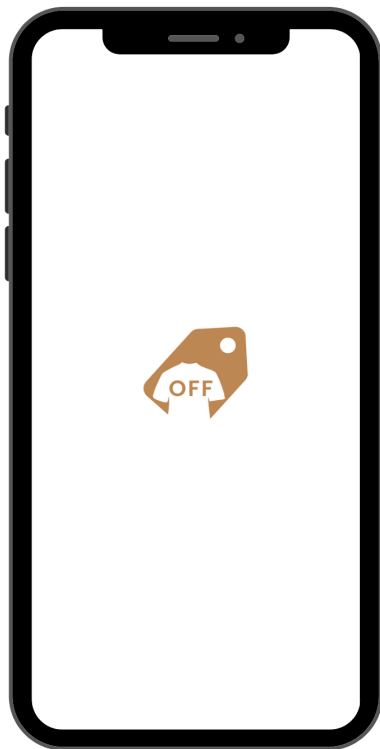


Figure 5.1: Splash Screen

2. Registration Screen

The Registration Screen is a fundamental component of a mobile application designed to onboard new users. It serves as the initial step in the user journey, allowing individuals to create accounts and gain access to the app's features and services. This screen typically includes input fields for essential user information, such as name, email address, password, and sometimes additional details like phone numbers or profile pictures. Users are required to input this information accurately and securely, following best practices for data validation and security.

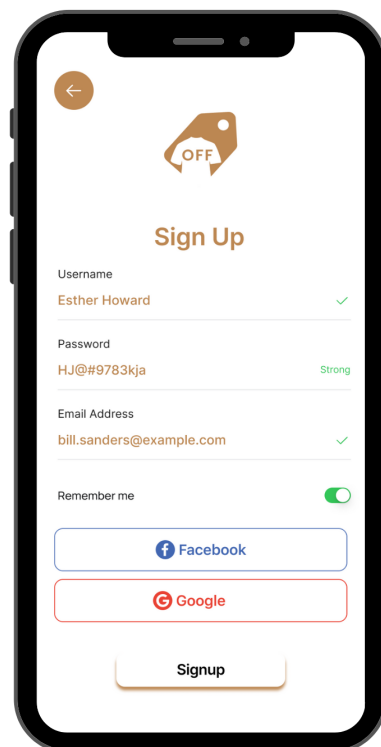


Figure 5.2: Registration Screen

3. Login Screen

The Login Screen is a user interface component designed for authentication and user access control in mobile applications built using the React Native framework. This screen typically includes input fields for users to enter their credentials, such as username/email and password, along with buttons for logging in or accessing password recovery options. Developers can customize and style this screen to align with the app's design and functionality requirements. It serves as a crucial entry point for users to access the app's features and data while ensuring security and user identity verification. Proper implementation and design of the Login Screen are essential for a seamless and secure user experience in React Native applications.

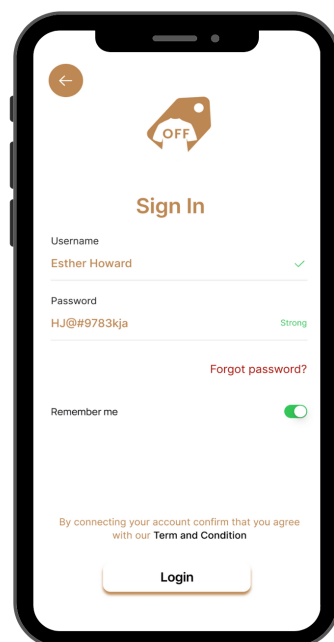


Figure 5.3: Login Screen

4. Forgot Password Screen

The Mobile Forgot Password Screen is a crucial component of a mobile application's authentication process. This screen allows users to reset their forgotten passwords securely.

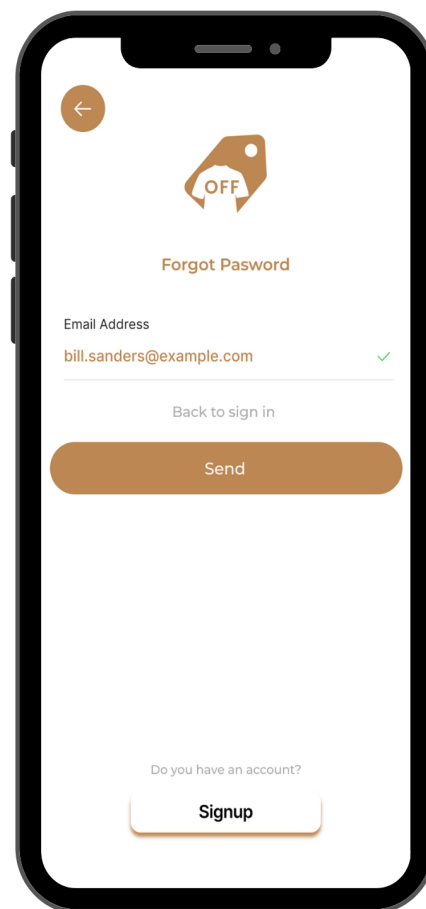


Figure 5.4: Forgot Password Screen

5. New Password Screen

This screen is a user interface within a mobile application or device that allows users to change their current password to a new one. This screen typically includes fields for entering the old password and the desired new password, along with validation requirements such as password strength indicators or rules (e.g., minimum length, special characters). Users are usually prompted to confirm the new password to ensure accuracy and security. This functionality is commonly found in apps and devices to enhance account security and protect user data.

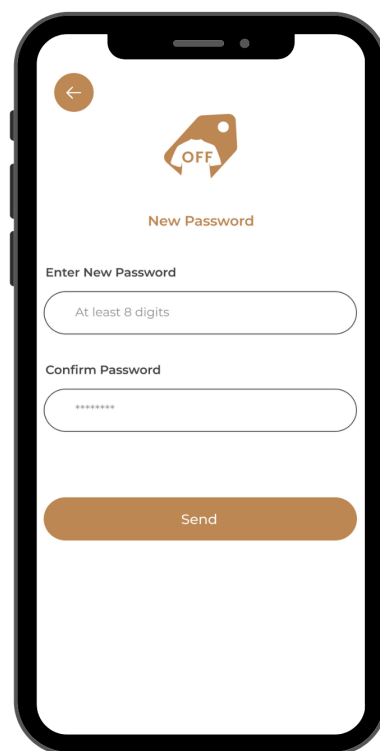


Figure 5.5: New Password Screen

6. Home Screen

The HOME screen serves as the initial landing page or main interface of a mobile application built using the React Native framework. This screen typically provides users with easy access to essential features and information. It may include components such as navigation menus, buttons, icons, and text elements to guide users to various sections of the app. The HOME screen often reflects the app's branding and design aesthetics while maintaining a user-friendly and intuitive layout to ensure a positive user experience. Developers can customize the HOME screen to meet the specific requirements and goals of their React Native mobile application.

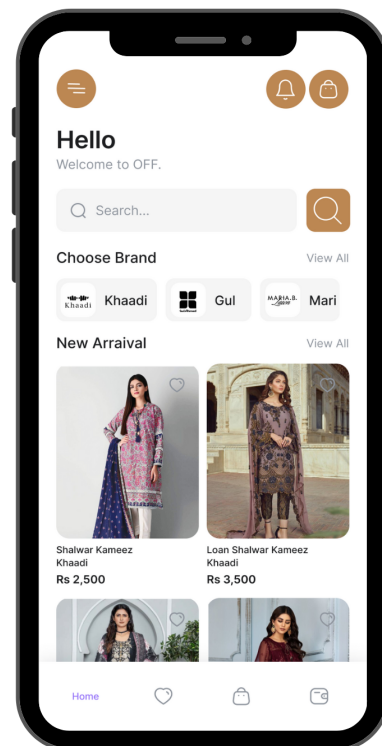


Figure 5.6: Home Screen

7. Menu Screen

A side menu bar on a documentation screen is a vertical panel typically located on the left or right side of the screen. It serves as a navigation tool, providing quick access to various sections, categories, or topics within the documentation. Users can click or interact with items in the side menu to easily browse, search, and access relevant information, making it a user-friendly and efficient way to explore and find documentation resources.

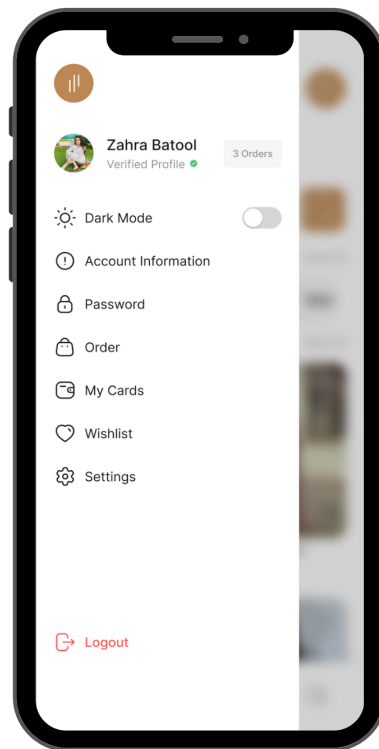


Figure 5.7: Menu Screen

8. Payment Method Screen

A payment method screen is a user interface (UI) component or page within the software or an application that allows users to manage and select their preferred payment methods. It typically includes options for entering or updating credit card information, bank account details, digital wallets, and other payment options. Users can use this screen to make payments, set up recurring payments, view transaction history, and ensure the security of their financial information. The design and functionality of this screen are crucial for providing a seamless and secure payment experience to users.

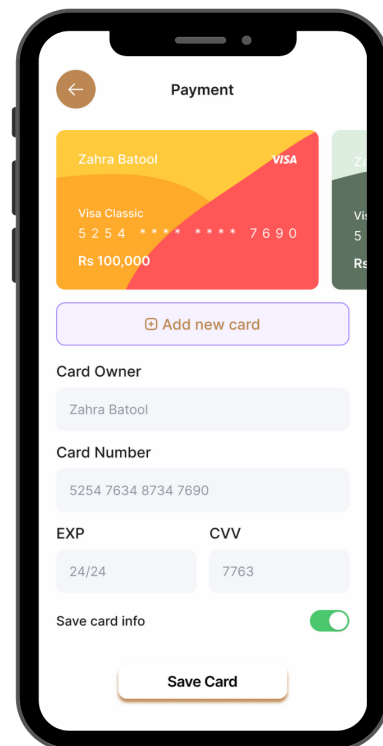


Figure 5.8: Payment Method Screen

9. Add Card Screen

A payment card screen typically refers to a visual representation or image of a payment card used in documents, manuals, or presentations. This screen displays the front and sometimes the back of a payment card, such as a credit card, debit card, or prepaid card, along with key information like cardholder name, card number, expiration date, and possibly a security code. It serves as a reference or illustration for users, employees, or readers to understand how a payment card looks and where to find important details when dealing with financial transactions or related documentation.

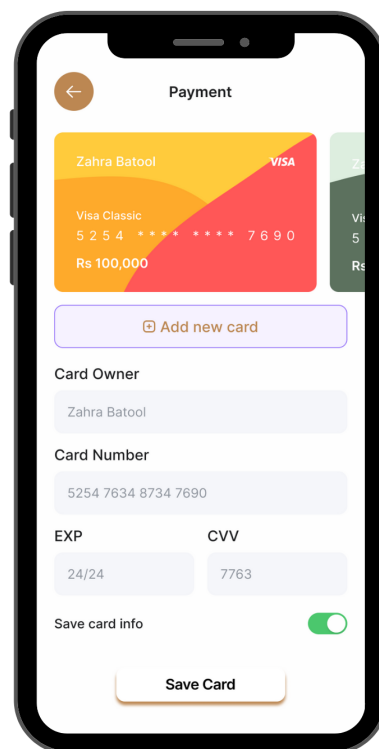


Figure 5.9: Add Card Screen

10. Order Confirmation Screen

An order confirmation screen is a digital document or page typically displayed on a computer or mobile device to provide customers with a summary of their recent purchase or order. It includes essential information such as the items or services ordered, quantities, prices, taxes, shipping details, and the total cost. This screen serves as a confirmation of the transaction and allows customers to review and verify their order before finalizing it, ensuring accuracy and transparency in the buying process. Customers often receive an email containing this information for their records after completing their purchase.

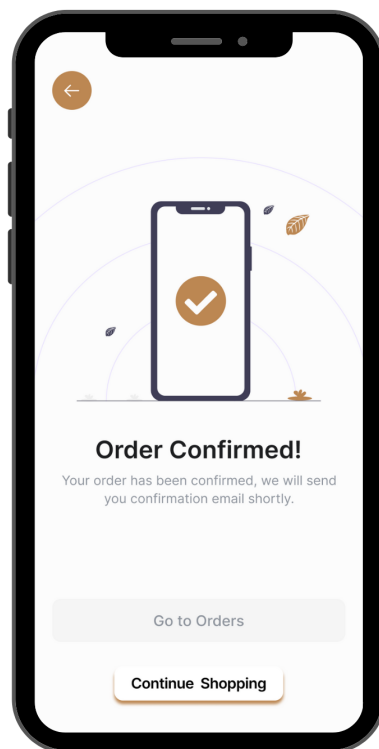


Figure 5.10: Oder Confirmation Screen

11. My Order Screen

The "My Orders" screen is a user interface component within an application or website that provides a summary of a user's past orders and their relevant details.

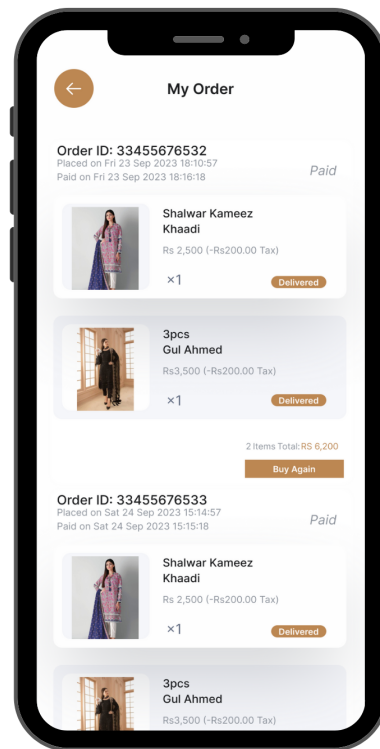


Figure 5.11: Orders Screen

12. Cart Screen

The CART screen is a user interface designed to facilitate data reporting and tracking activities. The CART screen serves as a central hub for users to input, manipulate, and access data, making it a crucial component of systems designed for reporting and tracking purposes. This documentation should provide users with a clear understanding of how to interact with the screen and its various functionalities.

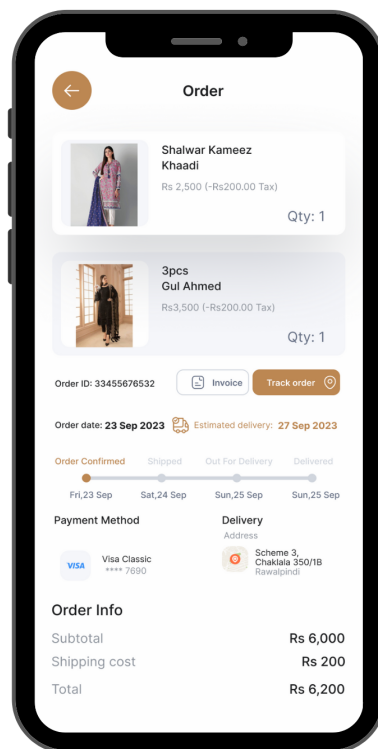


Figure 5.12: Cart Screen

12. Reviews Screen

The "Reviews" screen is a section within a software application or platform where users can access and interact with feedback, evaluations, or assessments provided by other users or experts. This screen typically displays a list of reviews, ratings, and comments related to a product, service, or content. Overall, the "Reviews" screen serves as a valuable resource for users to assess the quality, performance, and user satisfaction associated with a particular item or experience, helping them make informed decisions.

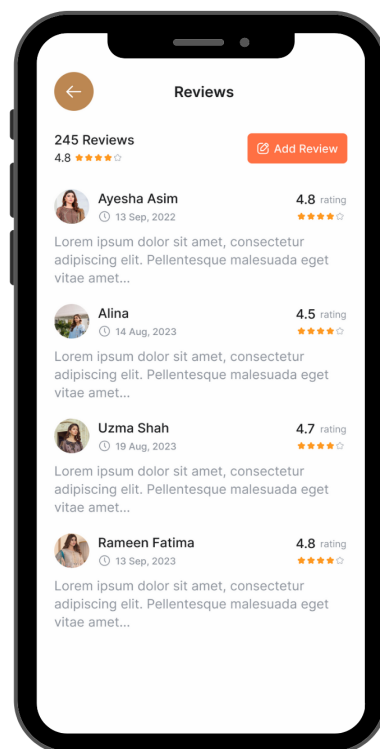


Figure 5.13: Reviews Screen

5.2.2 State Management

State management in React Native applications is the process of managing and controlling the data that drives the user interface and behavior of your mobile app. In React Native, as in React, there are various approaches to managing state, depending on the complexity and requirements of your application. State management is crucial because it helps you keep track of user interactions, data changes, and application state throughout the app's lifecycle.

5.2.3 Navigation

- React Navigation is often used for handling navigation between screens and creating a smooth user flow within the app.
- Navigation components include stack navigation for product details and tab navigation for different app sections.

5.2.4 Data Storage

Local storage or AsyncStorage is used for caching data locally on the user's device to improve app performance and offline functionality.

5.2.5 Payment processing

- Integration with payment gateways (e.g., Stripe, PayPal) is essential for handling secure payment transactions.
- User payment information is securely processed to complete purchases.

5.2.6 Security layer

Implement security measures, including encryption of sensitive data, secure communication with APIs (HTTPS), and best practices to protect user information.

5.2.7 Caching and Image Loading

Caching mechanisms are used to store images and frequently accessed data locally to reduce load times and improve app responsiveness.

5.2.8 Backend Server

- The backend server manages the e-commerce platform's core functionalities, including product catalog, order processing, and user management.
- It communicates with the mobile app through APIs.

5.2.9 Load Balancing and Scaling

To handle varying levels of traffic, load balancers and auto-scaling mechanisms are employed to ensure the system's scalability and availability.

5.3 Tools And Technologies

5.3.1 Visual Studio Code

We used VS Code for the development of our web application, it is a code editor with support for development operations like debugging, task running, and version control. We developed a client-side and server-side application on vs code which we will later deploy on Heroku for real-time alerts. We build the server-side application in the Django framework of python and used the REST framework for API. [1]

5.3.2 Android Studio

Figma is a versatile, cloud-based design and prototyping platform that empowers designers, UX professionals, and teams to create, collaborate on, and iterate user interfaces and user experiences in real-time. Offering a wide array of vector editing tools, interactive prototyping capabilities, design system support, and seamless cross-platform accessibility, Figma streamlines the entire design process. It facilitates collaboration through collaborative editing, comments, and shared design files, while also enabling version control and design history tracking. Figma has become a go-to tool for its ability to enhance design consistency, encourage efficient workflows, and promote effective communication among design teams and stakeholders.

5.3.3 Java Development kit

The Java Development Kit (JDK) is a software package provided by Oracle Corporation that serves as the foundation for developing Java applications. It includes essential tools and components, such as the Java compiler (javac), runtime environment (JRE), and various utility programs, enabling developers to write, compile,

debug, and run Java code. The JDK ensures cross-platform compatibility by allowing Java applications to run on different operating systems without modification. It also provides libraries and documentation for building Java applications, making it an integral part of Java software development.

5.3.4 Node.js

Node.js is an open-source, server-side JavaScript runtime environment built on the V8 JavaScript engine. It enables developers to execute JavaScript code outside of a web browser, making it a powerful platform for building scalable and efficient server applications. Node.js is known for its non-blocking, event-driven architecture, which allows for high concurrency and excellent performance, particularly in I/O-heavy applications. It has a rich ecosystem of packages and modules available through npm (Node Package Manager) and is widely used for developing web servers, APIs, real-time applications, and microservices, offering a versatile and efficient solution for a wide range of server-side programming needs.

5.3.5 Software Development Kit

A Software Development Kit (SDK) is a set of software tools, libraries, documentation, and resources that developers use to build applications for a specific platform, framework, or operating system. SDKs provide pre-built functions, APIs (Application Programming Interfaces), and development environments tailored to a particular technology, allowing developers to streamline and simplify the app development process. They typically include everything needed to create, test, and deploy applications, saving developers time and effort by providing ready-made solutions for common tasks and challenges associated with the targeted platform. SDKs are instrumental in accelerating software development for various domains, such as mobile app development, game development, web development, and IoT (Internet of Things) development.

5.3.6 Node Package Manager

npm, short for Node Package Manager, is a widely used package manager for JavaScript, primarily associated with Node.js development. It serves as a central repository for a vast ecosystem of open-source libraries and tools that can be easily installed and managed in JavaScript projects. With npm, developers can effortlessly add, update, and manage dependencies, simplifying the process of building and maintaining JavaScript applications. It includes a command-line interface for

executing various package-related tasks, and its robust `package.json` configuration file allows developers to specify project dependencies and scripts, facilitating efficient project development and distribution.

5.3.7 Expo

Expo is an open-source framework and platform for building cross-platform mobile applications using JavaScript and React Native. It simplifies the mobile app development process by providing a set of pre-configured development tools, libraries, and a unified workflow. With Expo, developers can create iOS and Android apps without the need for platform-specific code, and they can quickly develop and test their apps on physical devices using the Expo client app or easily share their projects with collaborators. Expo also offers a wide range of built-in components and modules for common mobile app functionalities, making it a popular choice for both beginners and experienced developers looking to streamline the development of mobile applications.

5.3.8 Yarn

Yarn is a fast and reliable package manager for JavaScript applications. It was developed by Facebook in collaboration with other major companies and the open-source community. Yarn simplifies the process of managing dependencies for JavaScript projects by efficiently resolving and fetching packages, ensuring consistency across development environments, and offering features like parallel installation and a lock file for deterministic builds. With its speed and flexibility, Yarn has become a popular choice among developers for managing packages and dependencies in modern web development, enhancing the efficiency and reliability of the development workflow.

5.3.9 Redux

Redux is an open-source JavaScript library commonly used in web and React Native applications for managing and centralizing application states. It employs a predictable and immutable data flow, where the entire application's state is stored in a single, read-only JavaScript object called the store. Actions, which represent user interactions or events, are dispatched to update the store's state through pure functions called reducers. This architecture simplifies state management, facilitates debugging, and ensures that state changes are traceable and predictable, making it a powerful tool for maintaining complex application states and data flow in a structured and maintainable manner.

5.3.10 Figma

Figma is a versatile, cloud-based design and prototyping platform that empowers designers, UX professionals, and teams to create, collaborate on, and iterate user interfaces and user experiences in real-time. Offering a wide array of vector editing tools, interactive prototyping capabilities, design system support, and seamless cross-platform accessibility, Figma streamlines the entire design process. It facilitates collaboration through collaborative editing, comments, and shared design files, while also enabling version control and design history tracking. Figma has become a go-to tool for its ability to enhance design consistency, encourage efficient workflows, and promote effective communication among design teams and stakeholders.

5.3.11 QT Designer

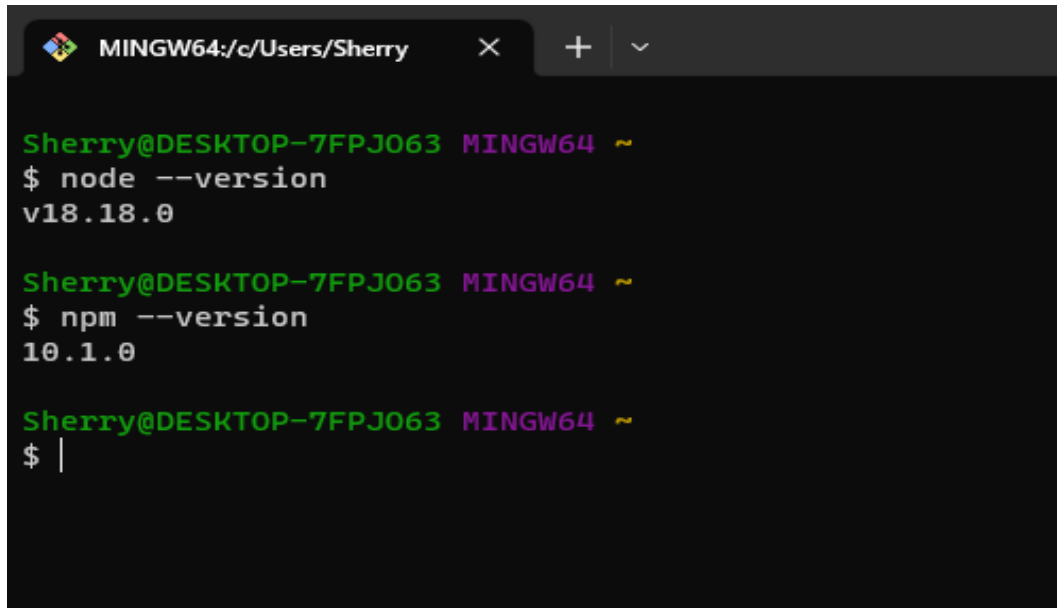
We created our application GUI by using the QT Designer which helped us to create productively and efficiently. Qt Designer is a tool that provides us user interface to create GUIs for our PyQt applications. With this tool, we can create GUIs by dragging and dropping QWidget objects on an empty form. After this, we can arrange them into a coherent GUI using different layout managers. Qt Designer is platform and programming language independent. It doesn't produce code in any particular programming language, but it creates a file with UI extensions which are XML files.

5.4 Experimental Setup

React Native is an open-source framework for building mobile applications developed by Facebook. It allows developers to create cross-platform mobile apps using a single codebase, primarily using the JavaScript programming language and React, a popular JavaScript library for building user interfaces. Setting up an experimental environment for React Native development involves several steps, and it may vary depending on your development platform (macOS, Windows, Linux). Here's a general guide to get you started

5.4.1 Node Js and NPM

Make sure you have Node.js and npm (Node Package Manager) installed on your system. You can download them from nodejs.org. After the installation is complete, you can open a command prompt or PowerShell window and verify that Node.js and npm are installed by running the following commands:

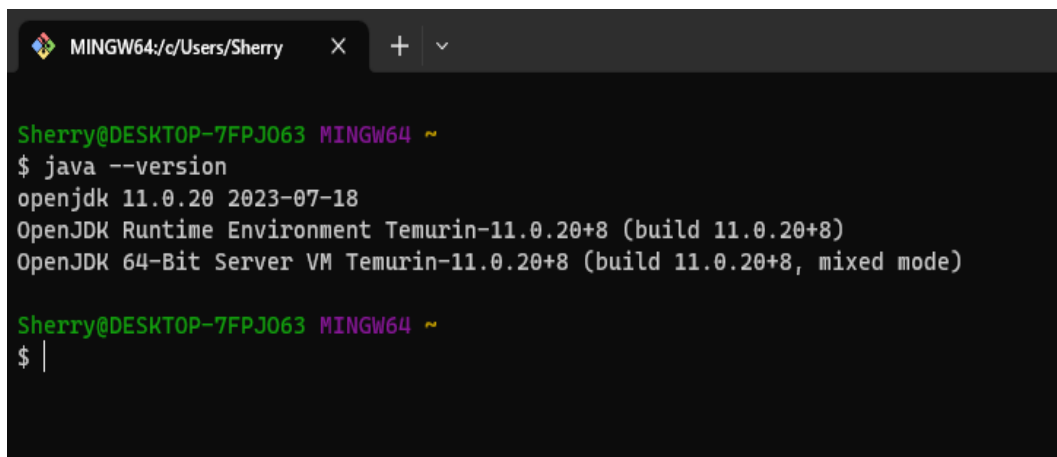
A terminal window titled 'MINGW64:/c/Users/Sherry' with a dark background. The prompt is 'Sherry@DESKTOP-7FPJ063 MINGW64 ~'. The user enters '\$ node --version' and the output is 'v18.18.0'. Then the user enters '\$ npm --version' and the output is '10.1.0'. Finally, the user enters '\$' and a cursor is visible.

```
Sherry@DESKTOP-7FPJ063 MINGW64 ~  
$ node --version  
v18.18.0  
  
Sherry@DESKTOP-7FPJ063 MINGW64 ~  
$ npm --version  
10.1.0  
  
Sherry@DESKTOP-7FPJ063 MINGW64 ~  
$ |
```

Figure 5.14: Node version

5.4.2 Java Development Kit (JDK)

React Native requires a JDK version 8 or later. You can download and install it from the Oracle website or use OpenJDK. After the installation is complete, you can open a command prompt and verify that the JDK is installed by running:

A terminal window titled 'MINGW64:/c/Users/Sherry' with a dark background. The prompt is 'Sherry@DESKTOP-7FPJ063 MINGW64 ~'. The user enters '\$ java --version' and the output is 'openjdk 11.0.20 2023-07-18', 'OpenJDK Runtime Environment Temurin-11.0.20+8 (build 11.0.20+8)', and 'OpenJDK 64-Bit Server VM Temurin-11.0.20+8 (build 11.0.20+8, mixed mode)'. Then the user enters '\$' and a cursor is visible.

```
Sherry@DESKTOP-7FPJ063 MINGW64 ~  
$ java --version  
openjdk 11.0.20 2023-07-18  
OpenJDK Runtime Environment Temurin-11.0.20+8 (build 11.0.20+8)  
OpenJDK 64-Bit Server VM Temurin-11.0.20+8 (build 11.0.20+8, mixed mode)  
  
Sherry@DESKTOP-7FPJ063 MINGW64 ~  
$ |
```

Figure 5.15: Java version

5.4.3 Android Studio

If you plan to develop for Android, install Android Studio. Make sure to set up the Android Virtual Device (AVD) for testing. Follow the official documentation for Android Studio installation.

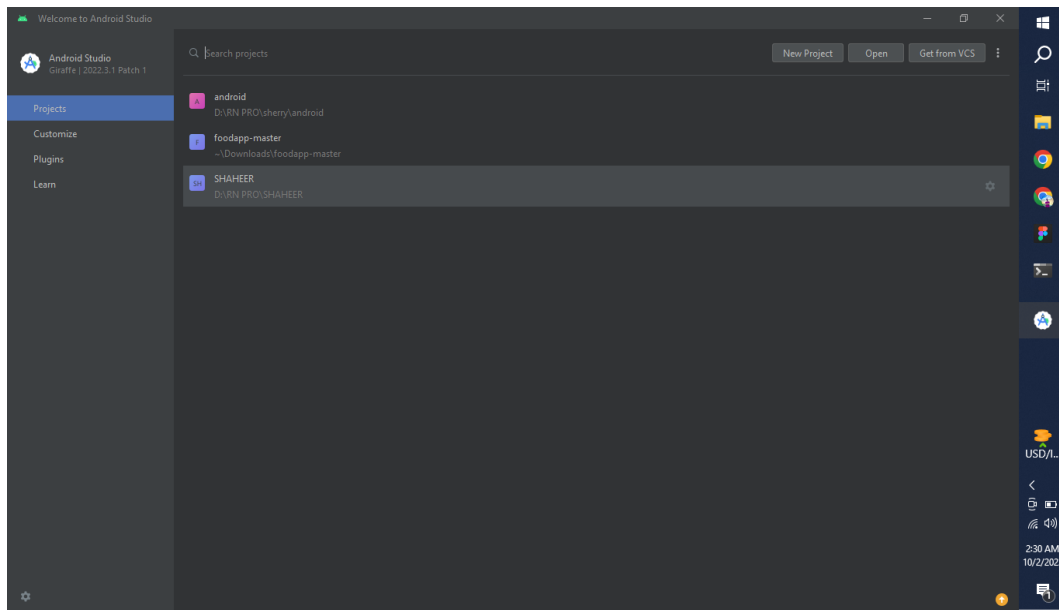


Figure 5.16: Android Studio

5.4.4 Install React Native CLI

Open your terminal and run the following command to install the React Native CLI globally:

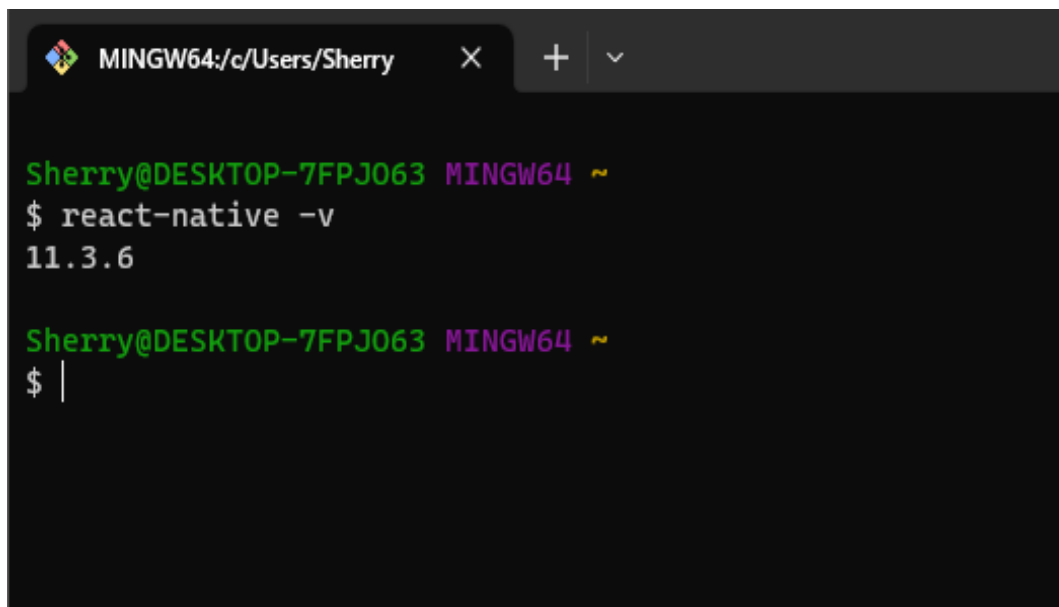


Figure 5.17: React Native CLI

5.4.6 Debugging and Testing

You can use various debugging and testing tools to diagnose and test your app as you develop it. Expo provides a range of helpful tools for this purpose, including the Expo DevTools in your browser and the ability to run your app on physical devices.

5.4.7 Building and Publishing Your App

When you're ready to share your app with others or publish it to app stores, you can build it using the Expo build commands or eject from Expo to use the React Native CLI for more customization. Building and publishing apps require additional configuration and steps beyond this initial setup.

That's a high-level overview of creating a new React Native project using Expo CLI. React Native development can be an exciting journey, and there are many resources available online to help you learn and build amazing mobile apps.

5.5 Methodology

Developing a React Native application involves several steps and methodologies to ensure a structured and efficient development process. Here is a methodology you can follow when creating a React Native application:

5.5.1 Project Setup

- Install Node.js and npm if not already installed.
- Install React Native CLI or Expo CLI (as explained in the previous response).
- Create a new project using the chosen CLI tool.

5.5.2 UI/UX Design

- Define the user interface (UI) and user experience (UX) of your app.
- Create wireframes and design mockups to plan the app's layout and visual elements.
- Decide on navigation patterns and screen transitions.

5.5.3 Component Structure

- Plan and organize the structure of your React Native components.

- Break down the UI into reusable components for a modular and maintainable codebase.
- Use stateless functional components and class-based components as needed.

5.5.4 State Management

- Choose a state management library (e.g., Redux, MobX, React Context API) to manage the app's global state if required.
- Define the application's state and actions.

5.5.5 Styling

- Style your components using CSS-in-JS libraries like StyleSheet or use third-party libraries like Styled-components.
- Maintain a consistent and visually appealing design throughout the app.

5.5.6 Navigation

Implement navigation between screens using a navigation library such as React Navigation. Set up navigation stacks, tabs, and drawers based on your app's design and user flow.

5.5.7 API Integration

- Connect your app to backend APIs or services to fetch and send data.
- Handle asynchronous operations, such as data fetching and processing.

5.5.8 Testing

- Write unit tests for your components and functions using testing frameworks like Jest.
- Perform manual testing on different devices and platforms to ensure compatibility.

5.5.9 Error Handling and Logging

- Implement error handling to gracefully handle unexpected situations.
- Set up logging and error reporting tools for monitoring and debugging.

5.5.10 Performance Optimization

- Optimize performance by reducing unnecessary re-renders.
- Implement code-splitting and lazy loading to improve initial app load times.
- Profile and optimize performance bottlenecks.

5.5.11 Security

- Implement security best practices to protect user data and the app itself.
- Handle user authentication and authorization securely.

5.5.12 Deployment

- Prepare your app for deployment to app stores (Google Play Store and Apple App Store) or distribution platforms (Expo, TestFlight, etc.).
- Generate production builds for Android and iOS.

5.5.13 App Store Submission

- Follow the guidelines and requirements of the respective app stores for submission.
- Create app icons, screenshots, and promotional materials.

5.5.14 Monitoring and Maintenance

Monitor app performance and user feedback after launch. Release updates and bug fixes as needed.

5.5.15 Documentation

- Document your codebase, API endpoints, and any external libraries or services used.
- Create user documentation or help resources.

5.5.16 Analytics and User Feedback

- Integrate analytics tools to gather insights about user behavior.
- Collect and analyze user feedback to improve the app.

Remember that React Native development is an iterative process, and it's essential to stay updated with best practices, tools, and libraries to deliver a successful and maintainable mobile application.

Chapter 6

Testing

6.1 System Testing

System testing is a testing procedure carried out on your system, to evaluate the performance of your product, if it is working as planned, and if the system complies with its specified requirements. System testing is one of the most essential parts or processes in the building of your system or application. It is confirmed that many systems face failure when they're testing, and the evaluation is poor. Evaluation of our work is extremely important, not only of our own system but also evaluating it with the existing systems, software, hardware, and methods to have better knowledge and idea of our system. Testing can be of two types, qualitative and quantitative. In qualitative, we look at the minor details, understanding more in detail what the user wants whereas quantitative is all about objectivity and group behavior. We must know where our system excels and where our system lacks, there is no such thing as development that is perfect. Flaws and imperfections are a part of every process or system. The following are different types of testing that should be considered during the system testing.

6.2 Functional testing

Functional testing is basically in which through our team of testers a quality assurance is determined whether our application is acting the way it is supposed to as our defined or mentioned requirements. In our system, the functional testing would be testing the functionalities of the web application such as the user interaction, and as well as the tasks or functionalities are being performed correctly and logically. Our main purpose is to make sure that the quality is being met according to our

expectations of the system and to also reduce errors so that in return there is a complete customer satisfaction. The functional testing, we performed as follows:

- To check If all the required and mandatory fields are present and being displayed correctly on the screen.
- To check, if all our fields are working fine, such as enter password or log in details.
- To check, how the application responds to closing or reopening of the react native application.

6.3 Interface testing

In the Interface testing, the system's GUI functionality is tested. We check whether the system is meeting the requirements of the application or not. The main testing of our system mainly comprised of the system being able to complete the payment successful and the user data privacy. In our web application, there is a dashboard that contains alert information, so we see if the panel can scroll properly or not. As the main purpose of Interface testing is to verify the functionality of an interface, therefore we tested our app's interface from all the aspects.

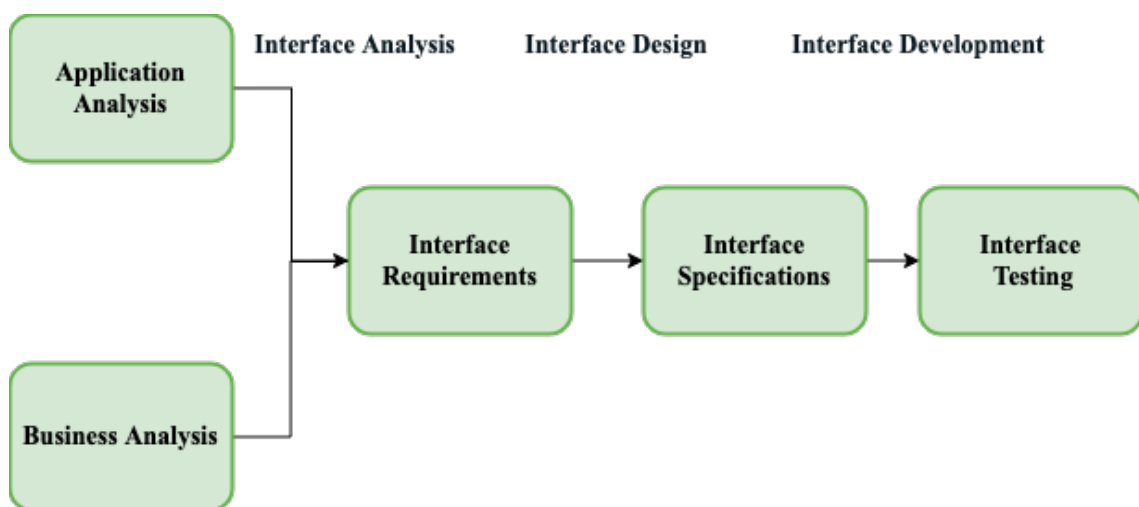


Figure 6.1: Interface Testing

6.4 Usability testing

Usability in simpler words is testing out how easy it is to use your system or application. How practical and user-friendly system have you built. Keeping in mind the user experience comes with the user being satisfied and that is achieved by building a user-friendly app, thus considering this factor, we have tried to keep our design quite simple and minimalistic so that the user faces no difficulty in using our application. Usability testing requires representative users to test your application to get better feedback rather than just your own developer testing the app. We have kept our application quite simple; all the user has to do is register and expose a gun to the camera so that it can detect it.

6.5 Compatibility testing

Compatibility testing is used to check the compatibility of our system with different computing models, meaning which platform supports the working of our application. Our web application is compatible with all ios and android devices.

6.6 Performance testing

Performance testing of an OFF React Native application involves evaluating various aspects of the app's performance to ensure it functions efficiently and provides a seamless user experience. This type of testing helps identify bottlenecks, slow-loading components, and any issues that might affect user satisfaction.

6.7 Testing Strategies

- Black Box Testing
- Specification Testing
- White Box testing

6.7.1 Black Box Testing

Black box testing of an OFF React Native application involves assessing the application's functionality without delving into its internal code or structure. Testers treat the application as a "black box," where they interact with its user interface and functionalities to verify if it behaves as expected. In the context of an e-commerce app,

this testing focuses on various aspects, such as user registration, product browsing, cart management, payment processing, and order placement.

Testers start by defining test cases based on the app's requirements and specifications. They simulate user interactions like searching for products, adding items to the cart, applying discounts, filling out shipping information, and completing the purchase. Testers also evaluate how the app handles edge cases, such as incorrect user inputs, network disruptions, or invalid payment details.

During black box testing, testers verify that:

Navigation and User Experience: The app's navigation flows smoothly, providing a user-friendly experience. Testers ensure that users can easily move between screens, categories, and products.

Functional Requirements: All core features, such as product searching, product details, reviews, and order history, work correctly. Testers confirm that users can perform essential actions like adding/removing items from the cart and applying promo codes.

Data Validation: Input fields, forms, and validation messages are accurate and consistent. Testers check for proper error handling when users enter incorrect or incomplete information.

Payment Processing: The payment gateway functions correctly, processing payments securely and accurately. Testers verify that users can make payments using various payment methods (credit cards, PayPal, etc.).

Compatibility: The app performs well on different devices (phones and tablets) and screen sizes. Testers assess how the app adapts to various Android and iOS versions.

Security: Sensitive user data is protected, and there are no vulnerabilities that could compromise user information or payment details.

Performance: The app responds quickly, even under heavy user loads. Testers monitor its response times, ensuring it doesn't crash or freeze during usage.

Error Handling: The app gracefully handles errors, providing clear error messages to users when issues arise. Testers verify that users receive helpful guidance when problems occur.

Cross-Platform Consistency: In the case of React Native, testers ensure that the app behaves consistently across both Android and iOS platforms.

Regulatory Compliance: Depending on the jurisdiction and nature of the e-commerce, testers may also check if the app complies with legal and regulatory requirements, such as GDPR, PCI DSS, or local tax laws.

Black box testing is essential to identify user-facing issues, usability problems, and functional defects without needing knowledge of the app's internal code. It helps ensure that the e-commerce React Native application delivers a reliable and user-friendly experience to customers.

6.7.2 Specification Testing

Specification testing for an OFF React Native application involves ensuring that the app meets all the specified requirements and functionality defined for it. This type of testing aims to validate that the application behaves correctly according to its specifications

6.7.3 White Box Testing

White-box testing of an OFF React Native application involves examining the internal structure, code, and logic of the application to assess its functionality, security, and robustness. Testers typically have access to the application's source code, which allows them to design test cases based on an understanding of the codebase.

- Code functionalities
- Testing each function
- Response time

6.8 Testing Performance Test Cases

To evaluate the performance of our Application, we performed several tests to analyze the performance of our application. Following are test cases we applied to our system.

- To ensure that the response time of application are according to requirements.
- To check whether our client-side application and server-side application remain connected during all time
- To ensure the UI/UX quality is also satisfactory
- To ensure payment transactions

- To ensure that the snapshot of the order and transaction history is also saved

6.9 Testing Usability Test Cases

As discussed earlier, usability testing involves how user- friendly the application is. The main purpose is to have an easy-to-use app rather than a complicated one that will be difficult to understand and operate. Therefore, we intend to build an application that is easy to use and acceptable as well to be sold in the market. For this purpose, we ensured the following:

- Font size to be visible and easy to read
- Buttons to be of a standard size
- Buttons to be placed on the same screen
- Logo to be consistent with the application
- Color schemes not to be too sharp

6.10 Test Cases

Various test cases were performed and run through our OFF Clothing application to check the system's performance & effectiveness. We have listed them below:

6.10.1 Test Case 1 : Registration

In test case 1, we will be testing our application's registration process. We tested this process and our application successfully passed this test as our ios and android both was signed up successfully several times without any bugs and errors.

TestID	1
Test Case Description	Testing of Registration Process
Initial State	Application Should be Running
Input	User will enter details
Expected Output	Successful Sign Up
Output	User Input is valid and account is created
Status	Pass

Table 6.1: Test Case: Register

6.10.2 Test Case 2 : Log In

In test case 2, we will be testing our ios and android both application's registration process. We tested this process and we were able to login multiple times and our application passed this test successfully without any errors.

TestID	2
Test Case Description	Testing of login Process
Initial State	Application Should be Running
Input	User will enter details
Expected Output	Successful Sign In
Output	User log in details are valid and signed in
Status	Pass

Table 6.2: Test Case: Login

6.10.3 Test Case 3 : Verification Code

In test case 3, we will be testing our client-side application core functionality of validation. We tested this process and we were able to validate Emails multiple times and our server side received alerts and snapshot of information successfully without any errors.

TestID	3
Test Case Description	Testing of Validation Process
Initial State	User must have a internet running with machine
Input	User put the email and important to sign in on otherdevice
Expected Output	The verification code is received successfully
Output	Our Application is validate successfully
Status	Pass

Table 6.3: Test Case: Gun Detection

6.10.4 Test Case 4 : Saving Snapshot

In test case 4, we will be testing our client-side application where we are testing that our application saves the snapshot after successful users registration. We tested this process and we were able to snapshot multiple times and passed this test.

TestID	4
Test Case Description	Testing of Snapshot Saving Process
Initial State	Application must be running and client must be logged in
Input	The information will be successfully saved the snapshot
Expected Output	Saved Snapshot of users information
Output	Successfully saved snapshot of users information
Status	Pass

Table 6.4: Test Case: Saving Snapshot

6.10.5 Test Case 6 : Notification

In test case 6, we will be testing our alert notification by receiving email verification code. We tested this process and we were able to receive alerts multiple times and passed this test.

TestID	6
Test Case Description	Testing of mobile notifications
Initial State	Server-Side application must be running
Input	Alert must be received.
Expected Output	Receiving mobile notification alert
Output	Received Email Code including alert link
Status	Pass

Table 6.5: Test Case: Notification

6.11 Limitations

React Native, while a powerful framework for cross-platform mobile app development, has its limitations. One significant limitation is the potential for performance bottlenecks, especially in complex or graphically intensive applications. Due to its reliance on JavaScript and a bridge to native code, React Native may exhibit slower performance compared to fully native apps, particularly in scenarios where rapid data updates or complex animations are involved. Additionally, accessing platform-specific features and libraries might require writing native modules, which can be time-consuming. Overall, while React Native offers significant advantages in terms of code reuse and development speed, its performance and access to certain platform-specific features may pose challenges in resource-intensive or specialized applications.

Chapter 7

Conclusion

A React Native application named **OFF** represents a versatile and promising mobile application that has been developed using the React Native framework. In conclusion, "OFF" is designed to offer a wide range of features and functionalities that cater to the needs of its target audience. Here are some key points to summarize the application:

7.0.1 Cross-Platform Compatibility

OFF leverages the power of React Native, allowing it to run on both iOS and Android platforms with a single codebase. This cross-platform nature makes it cost-effective and efficient for development and maintenance.

7.0.2 User-Centric Design

The application's user interface (UI) and user experience (UX) have been thoughtfully designed to provide an intuitive and enjoyable experience for its users. User feedback and usability testing have likely played a crucial role in shaping the app's design.

7.0.3 Functionality

As an e-commerce application, "OFF" likely offers a wide range of functionalities, such as product browsing, searching, adding items to the cart, secure payment processing, order tracking, and user account management. These features are essential for a successful e-commerce platform.

7.0.4 Security

Security is a top priority for any e-commerce application. "OFF" is expected to employ robust security measures, including data encryption, secure authentication, and protection against common web application vulnerabilities like SQL injection and cross-site scripting (XSS).

7.0.5 Performance

The application is likely optimized for performance, ensuring that it loads quickly, responds promptly to user interactions, and handles high traffic loads efficiently. Performance testing and optimization are ongoing processes to deliver a smooth user experience.

7.0.6 Scalability

To accommodate growth and changing demands, "OFF" is designed to be scalable, allowing it to handle an increasing number of users, products, and transactions without compromising performance or reliability.

7.0.7 Maintenance

Regular updates and maintenance are essential to keep the application secure and up-to-date with the latest technologies and user expectations. The development team behind "OFF" is likely committed to continuous improvement.

7.0.8 Community and Support

Engaging with the React Native and e-commerce communities is crucial for addressing issues, receiving feedback, and staying informed about best practices and industry trends. "OFF" likely benefits from a supportive community and provides customer support channels for users.

7.0.9 Future Development

The development of "OFF" is an ongoing process. The application will continue to evolve with new features, improvements, and updates based on user feedback and changing market conditions.

In conclusion, **OFF** represents a robust and user-friendly e-commerce React Native application that is poised to provide a seamless shopping experience for its users. Its success depends on factors such as user adoption, effective marketing, and the

ability to adapt to the ever-changing mobile app landscape. With dedication and a focus on user satisfaction, "OFF" has the potential to thrive in the competitive e-commerce market.

References

- [1] Visual studio code information. <https://code.visualstudio.com/>.