CAKE ORDERING APPLICATION

MINI PROJECT REPORT

Submitted by

GR SANCHITHA (220701243)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE,

CHENNAI ANNA UNIVERSITY CHENNAI

MAY 2025

BONAFIDE CERTIFICATE

Certified that this project titled **"CAKE ORDERING APPLICATION"** is the bonafide work of **SANCHITHA GR (220701243)** who carried out the work under my supervision.Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. P. Kumar M.E., Ph.D.
M.E.
Head of the Department Professor
Department of Computer Science
and Engineering
Rajalakshmi Engineering College
Chennai – 602105

SIGNATURE

Mr.Bhuvaneswaran B,
Supervisor
AssistantProfessor
Department of Computer Science
and Engineering
Chennai – 602105

Submitted to Project Viva-Voice Examination held on _____

Internal Examiner **External Examiner**

ABSTRACT

The Cake Ordering Application is a user-friendly Android-based mobile solution developed using Android Studio with Kotlin and XML, designed to streamline the process of ordering cakes through a seamless digital interface. The application enables customers to conveniently select cake types, specify quantities, provide personal details, choose optional toppings, and place orders—thereby eliminating the need for traditional in-person cake ordering.

The core functionality begins with a clean and intuitive interface that allows users to select from a variety of cake types such as Chocolate, Vanilla, Red Velvet, and Cheesecake via a dropdown spinner. The customer then inputs the desired quantity, along with essential details like name, phone number, and delivery address. For added personalization, users can select optional toppings including whipped cream and chocolate sauce through interactive checkboxes.

Once the inputs are validated, the application generates a detailed order summary displaying all selected options and calculates the total payable amount, factoring in the cost of the chosen cake and additional toppings. This summary is presented in a neatly formatted TextView, and users can finalize the process by clicking the Place Order button, after which a confirmation toast message appears.

The app follows modular architecture, ensuring scalability and easy maintenance. Key Android components like Spinner, EditText, CheckBox, Toast, and TextView are utilized effectively to deliver a responsive user experience. The code structure promotes clean separation of logic into methods such as initializeViews(), setupCakeTypeSpinner(), validateInputs(), and createOrderSummary(). Security and performance considerations are observed by restricting sensitive activity access and minimizing memory footprint

through optimized view initialization. Layouts follow NoActionBar themes for modern UI interaction, and all components are registered appropriately in the AndroidManifest.xml. The Cake Ordering Application addresses common ordering challenges such as manual calculation errors, lack of personalization, and inefficient communication between customer and vendor. It serves as a digital assistant for cake shops, enhancing both customer satisfaction and business efficiency. Future enhancements may include payment gateway integration, order tracking, user accounts for order history, and image previews of cake options to enrich the customer experience. This project demonstrates practical application of Android development principles to solve real-world retail challenges in a concise, accessible, and engaging format.

ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S.Meganathan, B.E, F.I.E.,** our Vice Chairman **Mr. Abhay Meganathan, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) Thangam Meganathan, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution. Our sincere thanks to **Dr. S.N.Murugesan, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P.Kumar, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Mr. B.Bhuvaneswaran, M.E.,** Assistant Professor (SG), Department of Computer Science and Engineering, Rajalakshmi Engineering College for their valuable guidance throughout the course of the project. We are very glad to thank our Project Coordinator, **Mr. B.Bhuvaneswaran, M.E.,** Assistant Professor (SG), Department of Computer Science and Engineering for his useful tips during our review to build our project.

GR SANCHITHA (21162207091243)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Expansion
UI	User Interface
UX	User Experience
SDK	Software Development Kit
API	Application Programming Interface
IDE	Integrated Development Environment
XML	Extensible Markup Language
JVM	Java Virtual Machine
APK	Android Package Kit
OOP	Object-Oriented Programming
MVVM	Model-View-ViewModel (Architecture Pattern)
DB	Database
SQLite	Lightweight Database used in Android
SQL	Structured Query Language
HTTP	HyperText Transfer Protocol
URL	Uniform Resource Locator
Intent	Android Component for Activity Communication
Activity	Android Screen Component
RecyclerView	Dynamic Scrollable List View
Bundle	A map for passing data between Android components

INTRODUCTION

In today's fast-paced world, digital solutions are rapidly replacing traditional retail methods, offering users the convenience of accessing services on the go. The **Cake Ordering Application** is a usercentric, Android-based mobile application designed to digitize and streamline the process of ordering cakes. Traditionally, ordering a cake required either visiting a bakery or placing a phone call, both of which lack customization, real-time interaction, and error-proof order tracking. This application addresses such limitations by providing a seamless and interactive platform where users can browse cake options, customize their orders, and provide delivery details—all within a unified interface. The goal is to enhance customer satisfaction and improve operational efficiency for cake shops through a clean, accessible, and responsive mobile experience.

The application is developed using **Android Studio** with **Kotlin** for the backend logic and **XML** for the user interface layout. The app begins with a clean interface that features a visually appealing background, a structured layout built using FrameLayout and ScrollView, and responsive design practices to accommodate various screen sizes. The layout employs elements such as Spinner, EditText, CheckBox, TextView, and Button—all styled with proper contrast, padding, and elevation for readability and accessibility. From the main screen, users can select a cake type (e.g., Chocolate, Vanilla, Red Velvet, Cheesecake), specify the desired quantity, and enter essential customer information including name, phone number, and address.

To personalize the order further, the app provides optional toppings like whipped cream and chocolate sauce through interactive checkboxes. Once all inputs are provided, the system performs input validation and proceeds to generate a real-time, itemized order summary. This summary, displayed using a styled TextView, presents all selections and the calculated total cost based on the chosen cake and toppings. Upon final review, users can tap the **Place Order** button, which triggers a confirmation toast message, enhancing the sense of completion and feedback.

The application's logic is modular and follows best practices of Android development. Key methods such as initializeViews(), setupCakeTypeSpinner(), validateInputs(), and createOrderSummary() are implemented to ensure code reusability, maintainability, and clarity. The application structure adheres to a **NoActionBar** theme for a modern look, while performance and security considerations are addressed by minimizing memory usage and declaring non-exported activities in the AndroidManifest.xml. Components are optimized for accessibility, with sufficient touch targets and content descriptions included where needed.

Ultimately, the **Cake Ordering Application** is more than just a digital order form—it is a fully functional, end-to-end ordering system designed to solve real-world inefficiencies in retail baking services. It simplifies customer interaction, reduces manual errors, and offers a scalable foundation for future improvements such as payment integration, order tracking, image-based cake selection, and user profiles. The application demonstrates practical implementation of Android development concepts to meet contemporary business and customer needs within the food retail sector.

1.1 GENERAL

Ordering cakes through traditional methods often involves in-person visits, phone calls, or unclear communication regarding customization, pricing, and availability. This manual and fragmented approach can lead to errors, delays, and limited personalization. In an era where mobile technology is reshaping consumer behavior, there is a growing demand for a digital, centralized solution that simplifies and modernizes the cake ordering experience. The **Cake Ordering Application** meets this need by offering a user-friendly Android platform that enables customers to browse cake options, select quantities, choose toppings, enter delivery details, and instantly view order summaries—all from the convenience of their smartphones.

Built on the robust foundation of Android Studio using Kotlin and XML, the application combines visual clarity with functional efficiency. It eliminates the need for manual coordination and enhances customer satisfaction by streamlining the entire ordering process. With features like interactive dropdowns, autofill-friendly input fields, real-time cost calculation, and order confirmation feedback, the app ensures that every order is accurate and personalized. Its modular architecture supports scalability for future enhancements, while its intuitive design makes it accessible for users of all ages. The Cake Ordering Application not only saves time for both customers and vendors but also represents a significant step forward in transforming traditional retail bakery services into a modern, efficient, and digital experience.

1.2 OBECTIVES

1. **To provide a digital, user-friendly platform** for customers to order cakes conveniently from their Android devices, eliminating the need for in-person visits or phone-based ordering.

2. **To simplify the cake ordering process** by integrating key features such as cake type selection, quantity input, customer details entry, optional toppings, and instant order summary generation into a single seamless application.
3. **To enhance personalization and accuracy**, allowing users to customize their orders through an intuitive interface that includes dropdown spinners, editable text fields, and interactive checkboxes for toppings.
4. **To improve order management and reduce errors** by validating user input and displaying a clear summary of selected items, customer information, and the total payable amount before finalizing the order.

5. **To deliver an efficient and responsive user experience** through the use of essential Android components such as EditText, Spinner, CheckBox, and Button, supported by a clean layout and real-time feedback via Toast messages.
6. **To ensure scalability and maintainability** by following a modular code structure, enabling future integration of features like image previews, payment gateways, order tracking, and user account systems.

1.3 EXISTING SYSTEM

The cake ordering industry, particularly in the online space, faces several challenges that impact both consumers and businesses. Despite the convenience of online platforms, customers often encounter issues such as limited customization options, especially for themed cakes due to licensing restrictions. For instance, retailers like Sam's Club have shifted to online-only ordering policies for characterthemed cakes, leading to customer frustration when in-person customizations are no longer accepted

Additionally, cybersecurity incidents have disrupted online cake ordering services, causing cancellations and delays. Marks & Spencer, for example, experienced a cyberattack that halted online orders, leaving customers without their wedding cakes.

From a business perspective, cake shops often struggle with ineffective inventory management, especially during peak seasons. This can lead to overstocking, waste, or understocking, resulting in missed sales opportunities and customer dissatisfaction . Moreover, many cake shops lack a robust online presence and effective marketing strategies, limiting their reach and growth potential.

These challenges highlight the need for a unified and intelligent cake ordering system that addresses customization limitations, enhances cybersecurity, improves inventory management, and strengthens

online marketing efforts. The Cake Ordering Application aims to provide a comprehensive solution to these issues, offering a seamless and personalized cake ordering experience for customers while supporting cake shops in optimizing their operations.

1.4 PROPOSED SYSTEM

The Cake Ordering Application is designed as an efficient, user-friendly mobile solution for simplifying the cake ordering process. Built with Android Studio using Kotlin and XML, the application streamlines the selection, customization, and order placement processes. The proposed system integrates the following key features:

1. **Cake Type Selection (Spinner)**:

   The app provides a dropdown Spinner for users to select from various cake types like Chocolate, Vanilla, Red Velvet, and Cheesecake. This easy-to-use feature allows users to make a quick choice without navigating through multiple screens.

2. **Quantity and Customization Input (EditText & CheckBox)**:

   Users can input the quantity of cakes they wish to order via an EditText field, with validation to ensure only valid numbers are accepted. For additional customization, CheckBox options allow users to select toppings such as whipped cream and chocolate sauce, adding personalization to the order.

3. **Order Summary (TextView)**:

Once the user selects their desired cake and customizations, the application dynamically generates an order summary. The TextView displays the chosen options, the quantity, and the total cost, providing the user with a clear overview of their order before confirmation.

4. **Order Confirmation (Toast Message)**:

After placing the order, a confirmation Toast message is shown, acknowledging the successful order placement. This feature ensures that users receive instant feedback about their action.

5. **Data Validation**:

The system includes robust input validation to ensure that all fields (name, phone number, delivery address) are filled correctly before proceeding with the order. This prevents errors and ensures that users submit valid information.

6. **Modular Architecture**:

The code structure follows a modular approach with separate methods for handling different tasks, such as initializeViews(), setupCakeTypeSpinner(), validateInputs(), and createOrderSummary(). This ensures better maintainability, scalability, and easy debugging of the application.

7. **Optimized Performance**:

The application has been designed to minimize memory usage and ensure smooth performance. Unnecessary resources are avoided, and the initialization of views and data is optimized to prevent delays during user interactions.

8. **Minimal Dependencies**:

The system avoids the use of external libraries or SDKs, relying solely on Android's native components (such as Spinner, EditText, CheckBox, and Toast) to perform all necessary functions. This reduces the complexity of the codebase and ensures faster compilation times.

9. **UI/UX Design (XML Layouts)**:

The application utilizes clean and simple XML layouts to ensure a modern and responsive user interface. A NoActionBar theme is applied for a more immersive user experience, eliminating the traditional title bar for a full-screen layout.

10. **Simplified Flow**:

The app's flow is straightforward, from cake selection to order confirmation. This minimizes the number of steps required from the user and ensures that they can easily place an order with minimal effort.

BENEFITS OF THE PROPOSED SYSTEM

- **Streamlined Cake Ordering Process**: The application integrates cake selection, customization, and order placement into a single platform, reducing the need for multiple steps or external tools.

- **Personalized User Experience**: Offers easy cake type selection, customizable options (e.g., toppings), and personalized order summaries. Users can tailor their orders with ease, ensuring satisfaction with every purchase.
- **Real-Time Order Confirmation**: After placing an order, users receive immediate feedback through a confirmation Toast message, ensuring they are notified about the successful completion of their order.
- **Efficient Customization and Validation**: The app provides a seamless customization process with dropdown menus, checkboxes, and input validation. This helps users avoid errors while placing orders and guarantees a smooth experience.
- **Optimized User Interface**: A simple, intuitive interface with a modern design and minimal distractions ensures a pleasant user experience, from cake selection to order summary.
- **Cost Transparency**: Users can view a clear breakdown of their order, including the total payable amount, helping them make informed purchasing decisions.
- **Modular Architecture**: The system is designed for easy maintenance and scalability, allowing for future enhancements like adding more cake options, integrating payment gateways, or including delivery tracking features.
- **Performance-Optimized**: The application ensures smooth performance by minimizing unnecessary memory usage and optimizing view initialization, allowing users to interact with the app without lag or crashes.

2. LITERATURE REVIEW

The proliferation of smartphones and the increasing demand for convenience have significantly influenced consumer behaviour, particularly in the food industry. Cake ordering applications have

emerged as a response to this trend, offering users the ability to browse, customize, and order cakes from the comfort of their homes.

Research indicates that integrating online ordering systems into bakery businesses enhances operational efficiency and customer satisfaction. For instance, the development of the "Cake-Hut" ecommerce platform enabled customers to purchase customized cakes online, streamlining the ordering process and reducing manual errors . Similarly, the implementation of an online cake ordering system for a cake shop allowed customers to order cakes online, make payments on delivery, and helped cake shops take credit card payments and provide order confirmations

The success of cake ordering applications heavily relies on user experience (UX) and interface design. A case study on the "Tutu’s Cake Shop" Android application highlighted the importance of a userfriendly interface in enhancing customer satisfaction. The application allowed users to check for various cakes available at the store and purchase online, providing flexibility and convenience.

From a business perspective, cake ordering applications facilitate efficient order management and customer engagement. The development of Gooti Pastries' web-based Cake Ordering System addressed issues such as manual order recording and order mix-ups, leading to improved business operations and customer satisfaction.

Market research underscores the growing preference for online food ordering systems. Statistics reveal that restaurants with an online ordering system can raise their takeout profits by 30% higher than those who do not . This trend reflects a shift towards digital solutions in the food industry, with consumers seeking convenience and efficiency in their purchasing decisions.

In addition, cake ordering applications offer significant opportunities for personalization and customer engagement. Users can select from a variety of cake types, sizes, and customizations such as toppings, flavours, and decorations, making the ordering process more interactive and tailored to individual preferences. By allowing users to track the progress of their orders in real time and receive notifications

about delivery status, these apps foster a sense of transparency and reliability, contributing to improved customer satisfaction and retention. This personalized approach is a key factor driving the success of cake ordering apps in a competitive market.

The literature suggests that cake ordering applications play a pivotal role in modernizing bakery services, enhancing customer experience, and improving business efficiency. By integrating online ordering systems, focusing on user-friendly design, and adapting to market trends, bakeries can meet the evolving demands of consumers and stay competitive in the digital age.

2.1 KOTLIN AND ITS USES

Kotlin is a modern, statically typed programming language developed by JetBrains, and it is fully interoperable with Java. Initially introduced in 2011, Kotlin was designed to address several of Java's limitations while offering a more concise, expressive, and safe syntax. In 2017, Google officially announced Kotlin as a first-class language for Android development, significantly increasing its popularity among mobile developers.

Key Uses of Kotlin:

1. **Android Development**: Kotlin is widely used for Android app development due to its compatibility with Java, enhanced syntax, and features that improve productivity.

2. It reduces boilerplate code, offers better type inference, and eliminates common programming pitfalls such as null pointer exceptions, making it a preferred choice for creating efficient and reliable Android applications.

3. **Backend Development**: Kotlin can be used for backend development with frameworks like Ktor, Spring Boot, and Vert.x. It provides a smooth experience for building scalable serverside applications due to its concise syntax, functional programming capabilities, and compatibility with Java-based ecosystems.

4. **Cross-platform Development**: Kotlin Multiplatform allows developers to write shared code that works across multiple platforms (Android, iOS, Web, etc.). This feature significantly reduces development time and cost for projects targeting multiple platforms.

5. **Data Science and Scripting**: Kotlin is also making its mark in the data science and scripting domains. With libraries like KotlinDL and support for JVM-based tools, Kotlin is used for machine learning and other data-intensive tasks.

6. **Desktop Applications**: Kotlin, in combination with JavaFX or other frameworks, is used to develop desktop applications, providing a modern alternative to Java-based desktop solutions.

Kotlin's ability to seamlessly integrate with existing Java codebases, along with its modern features, has made it a popular choice for developers across various domains.

2.2 CAKE ORDERING APPLICATION

The Cake Ordering Application is designed to streamline the cake ordering process, offering a range of features that simplify customer interaction and order management. One of the primary functions is the seamless selection of cake types, customization options (e.g., toppings and flavors), and quantity. The

app allows users to select from various pre-defined cake types like Chocolate, Vanilla, Red Velvet, and more, ensuring a simple yet customizable ordering experience.

The app's user interface is built to be intuitive, with key features such as Spinner for selecting cake types, EditText fields for inputting quantity, name, phone number, and delivery address, and CheckBox for selecting additional toppings. Once the user inputs their details, the app automatically calculates the total payable amount and displays an order summary in a TextView, ensuring transparency in the ordering process.

Additionally, the application incorporates an order confirmation process with a Toast message, confirming the user's order and providing an assurance of order placement. This helps to avoid confusion, ensuring customers have a clear overview of their order before finalizing it. The app provides users with a stress-free way to select, customize, and order cakes with the ability to track progress.

The Cake Ordering Application also features user-friendly navigation, with separate methods for input validation (to ensure correct data entry) and order summary generation (ensuring accuracy in the payment and itemization process). It integrates key Android components like Spinner, CheckBox, and TextView to create a responsive and efficient user experience, ensuring customers can easily place their orders.

2.3 ORDER CUSTOMIZATION AND SUMMARY GENERATION

The customization feature of the Cake Ordering Application plays a vital role in enhancing the user experience by allowing customers to personalize their cake order. This includes selecting from a variety of cake types, specifying the quantity, and choosing optional toppings such as whipped cream or

chocolate sauce. The customization options are clearly presented in the interface, with each option made available through simple UI elements like spinners and checkboxes.

Once the user selects their preferred cake type and customizations, the application dynamically calculates the total cost of the order. This is done by factoring in the base price of the selected cake type and any additional costs from chosen toppings. The final summary is displayed in a neatly formatted TextView for review, allowing the user to ensure that all details are correct before proceeding with the order.

This feature addresses one of the key challenges in traditional cake ordering processes—ensuring that the order is customized and calculated accurately. By incorporating real-time price calculation and summary generation, the Cake Ordering Application reduces the likelihood of errors and ensures a smooth, transparent user experience.

2.4 USER FEEDBACK AND ODER CONFIRMATION

To enhance customer trust and satisfaction, the Cake Ordering Application features a streamlined order confirmation system. After customizing their cake order, including selecting the cake type, quantity, and toppings, users are presented with a clear order summary, which includes all details such as cake type, quantity, toppings, and total cost.

Once the user confirms their order, a Toast message immediately appears, notifying them with a simple message like “Order confirmed for [cake type] with [toppings].” This non-intrusive confirmation ensures the user is promptly informed without disrupting their experience. The use of Toast notifications provides instant feedback, reassuring customers that their order has been successfully placed.

Additionally, users can review their order details, including estimated delivery time and payment options, ensuring full transparency and reducing uncertainty. This process fosters customer confidence by ensuring their order is processed accurately and promptly, enhancing the overall user experience and trust in the app.

2.5 CHALLENGES AND OPPORTUNITIES

Developing the Cake Ordering Application presents both challenges and opportunities. One of the main challenges is ensuring the accuracy of order calculations, especially when handling complex customizations such as multiple toppings or large orders. The app must also handle user inputs efficiently, ensuring that quantities and personal details are correctly validated before proceeding with order confirmation.

The simplicity and modularity of the app also present opportunities for further expansion. For example, integrating payment gateway options for online transactions could significantly enhance the app's functionality. Additionally, the ability to allow users to track their order status or receive delivery notifications would further improve the customer experience. Future enhancements may include user accounts, where customers can save their order preferences for quicker future purchases.

While there are technical challenges involved in data validation and ensuring accurate order calculation, the growing demand for mobile cake ordering applications presents vast potential. There is an opportunity to cater to a growing market of tech-savvy consumers who prefer to handle their cake orders through convenient, efficient mobile apps.

2.6 CONCLUSION

The Cake Ordering Application serves as an innovative solution to modernize the traditional cake ordering process. By offering an easy-to-use platform for selecting cakes, customizing orders, and ensuring accurate pricing, the app simplifies the entire ordering experience for customers. Key features, such as real-time price calculation, order summary display, and secure order confirmation, make the app a user-friendly tool for both casual and frequent cake buyers.

Despite the challenges of ensuring accurate calculations and validating user input, the app has a vast potential for growth. Future updates can include payment gateway integration, delivery tracking, and user account functionalities, further enhancing the app's appeal. Ultimately, the Cake Ordering Application aims to provide a seamless, personalized, and efficient cake ordering experience, making it an indispensable tool for cake enthusiasts.

3. SYSTEM DESIGNS

The System Design section outlines the architecture, components, and the design approach for the Cake Ordering Application. This system is built to provide a seamless cake ordering experience, from order customization to payment processing and delivery tracking. The following details explain how the components of the application work together to ensure a smooth, efficient, and scalable system.

3.1 GENERAL

3.1.1. SYSTEM ARCHITECTURE

The system architecture for the Cake Ordering Application follows a multi-layered design that prioritizes performance, scalability, and ease of maintenance. The core components of the system include:

- **Mobile Application (Android)**: The main interface where users interact with the app, customize their cakes, place orders, and track deliveries. It is built using Android Studio and ensures a smooth user experience with responsive UI elements.
- **Database**: A central database stores customer data, cake orders, payment information, and delivery details. This is typically implemented using Firebase or a cloud-based relational database (e.g., MySQL) for scalability and reliability.
- **Payment Gateway**: A secure API for processing payments (e.g., PayPal, Stripe) to handle the financial transactions of the users.
- **Backend Server**: The backend server handles business logic, user authentication, order processing, and communication with the database. It is built using a backend framework (e.g., Node.js, Spring Boot) and ensures that the app runs smoothly by managing data flow and ensuring security.
- **Notification System**: Sends push notifications to users about order status, delivery updates, and promotional offers. This is integrated using services like Firebase Cloud Messaging (FCM) or similar services.

3.1.2. FUNCTIONAL DESIGN

The Cake Ordering Application operates through several workflows and activities designed to provide a seamless ordering experience:

3.1.2.1 Cake Customization and Order Management

- **Input**: The user selects a cake type, size, and custom toppings, along with any personalization options like messages or decorations. The app collects these preferences through an interactive user interface.
- **Process**: Once the order is placed, the backend processes the user's selection, calculates the total price based on the customizations, and prepares the order for checkout.
- **Output**: The user receives an order summary, including cake details, price, and estimated delivery time. Upon confirmation, a payment gateway is triggered for the payment process.

3.1.2.2. Data Flow

The data flow in the system is as follows:

1. **Order Input**: The user inputs their order details through the mobile app, which are then sent to the backend server.
2. **Order Processing**: The backend processes the order by calculating costs and checking the availability of ingredients. It sends the order details to the kitchen or bakery team for preparation.
3. **Payment Processing**: Once the user confirms their order, the app triggers a payment request via the integrated payment gateway. The transaction is processed securely.
4. **Notification & Confirmation**: After successful payment, the user receives an instant confirmation of their order and estimated delivery time through push notifications.
5. **Delivery Tracking**: The user can track the delivery status in real time. The system updates the user on the cake's journey until it reaches the customer.

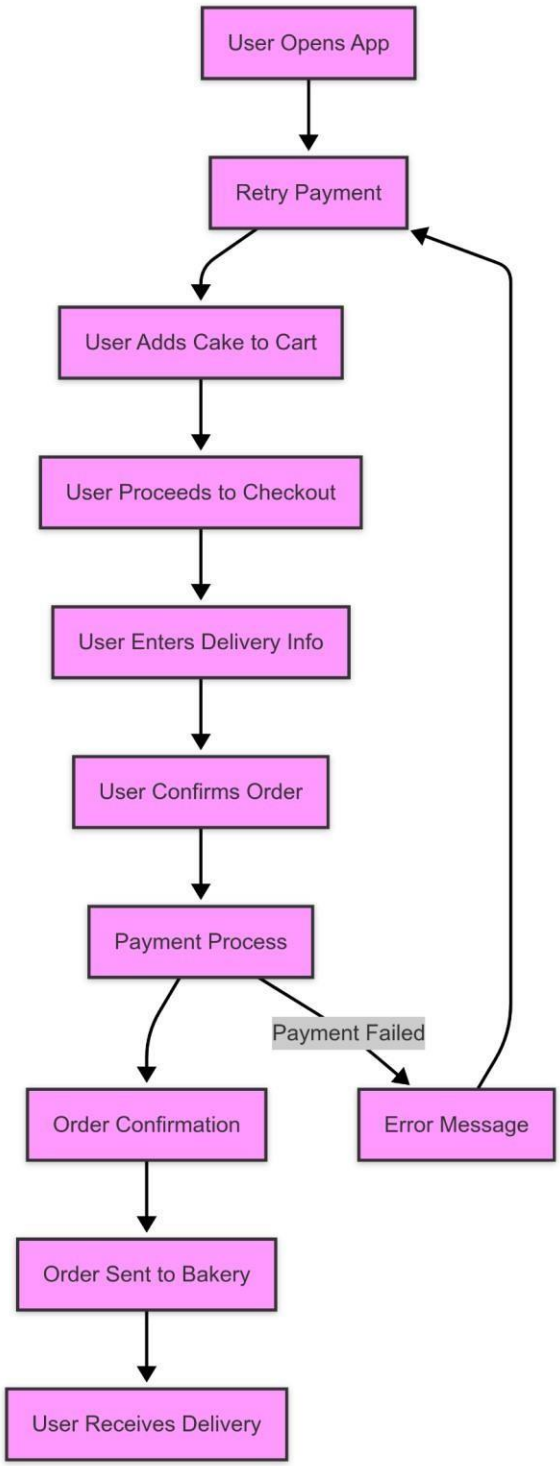
3.1.2.3. Technical Design

The technical design of the Cake Ordering Application is based on the following tools and technologies:

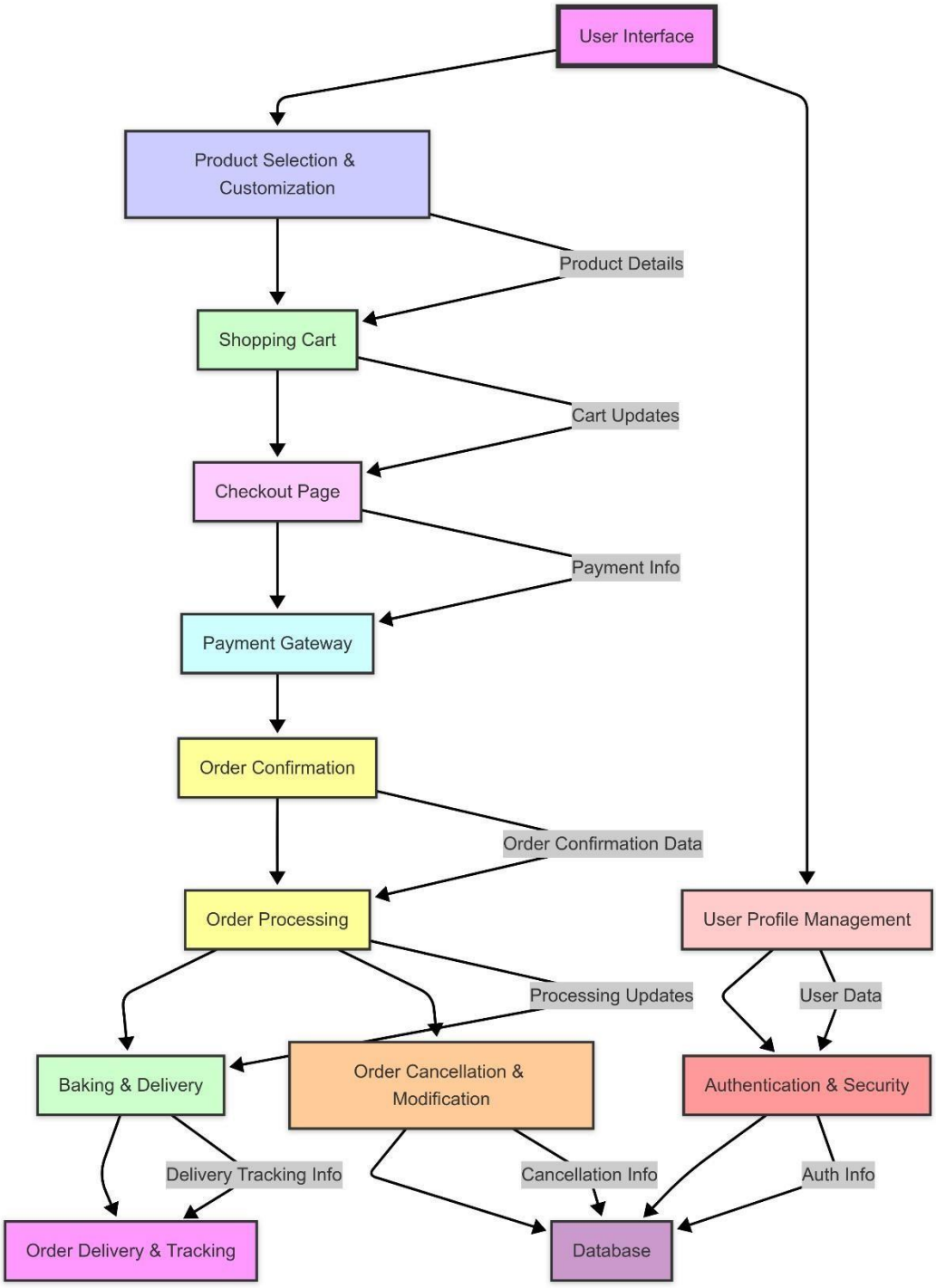
- **Android Studio**: The primary IDE for developing the Android application. It handles UI design, user interaction, and backend communication through APIs.
- **Firebase or SQL Database**: The database used to store user profiles, order details, and transaction records. Firebase allows real-time data syncing, while a traditional SQL database can be used for structured data storage.
- **Payment Gateway (e.g., Stripe, PayPal)**: Secure payment processing services that handle credit card transactions and ensure financial security during the checkout process.
- **Push Notification Service (e.g., Firebase Cloud Messaging)**: For sending real-time notifications to users regarding order updates and promotions.
- **Backend Framework (e.g., Node.js, Spring Boot)**: The server-side framework that manages the app's business logic, authentication, and communication between the mobile app and database.
- **Google Maps API**: For delivery tracking and showing the delivery route to users.

By combining these technologies, the Cake Ordering Application ensures a seamless, efficient, and secure process for cake customization, order management, payment, and delivery.

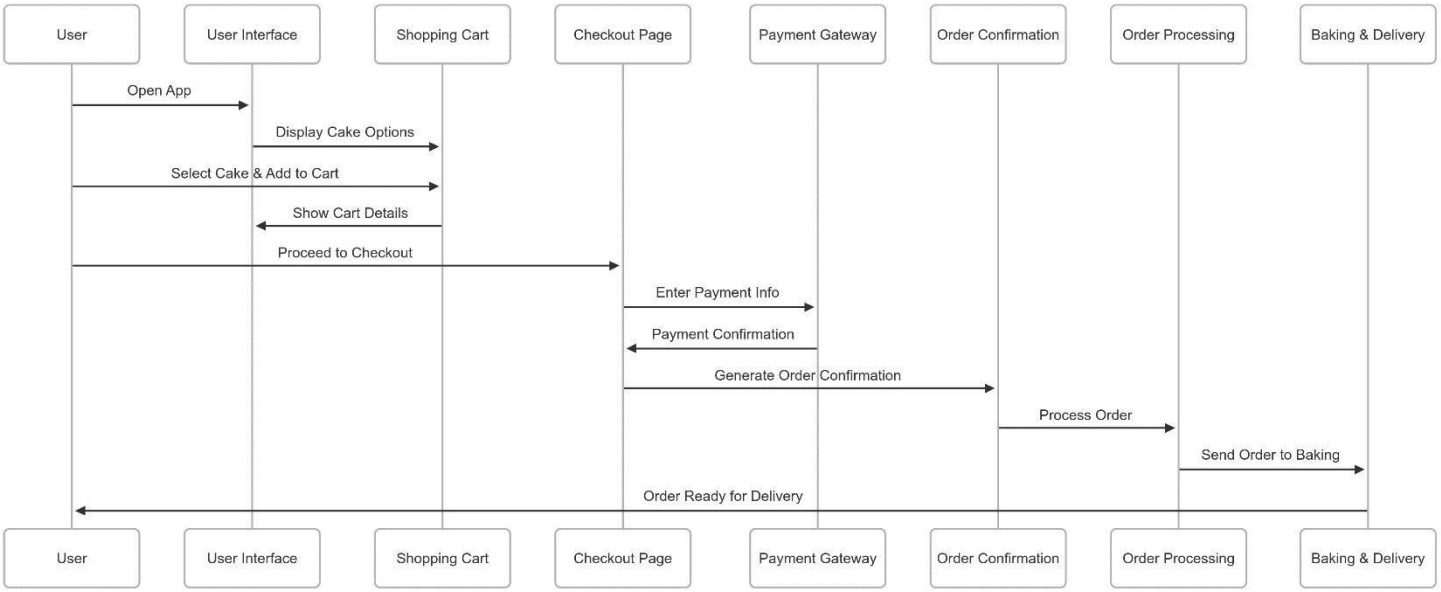
3.1. SYSTEM FLOW DIAGRAM



3.2 ARCHITECTURE DIAGRAM



3.2 SEQUENCE DIAGRAM



4. PROJECT DESCRIPTION

The **Cake Ordering Application** is an Android-based mobile application developed using Kotlin and XML in Android Studio. It is designed to provide a seamless and user-friendly experience for customers to browse, customize, and order cakes online. The app addresses common issues such as disorganized orders, complicated navigation, and lack of customization options by integrating all essential features into a single platform, making the cake ordering process easy and enjoyable.

The application begins with an eye-catching Splash Screen and provides secure Login/Signup pages for personalized access. After successful authentication, users are directed to the Home Page, which serves as the central navigation point for all features. The app follows a modular architecture, ensuring scalability, ease of maintenance, and smooth navigation between activities using Intents, with all activities registered in the AndroidManifest.xml file.

Key Features:

- **Product Selection & Customization**: Users can browse various cake options, customize flavors, sizes, and toppings, allowing them to personalize their orders.
- **Shopping Cart**: Users can review their selected items before proceeding to checkout, with options to modify quantities or remove items.
- **Checkout & Payment**: After confirming the order, users enter payment details and proceed to the payment gateway, ensuring a secure transaction process.
- **Order Confirmation**: A confirmation page displays order details, estimated delivery times, and a unique order ID. A toast message also confirms successful order placement.
- **Order Tracking**: Users can track the status of their orders in real-time, from baking to delivery.
- **Order History**: Allows users to view their past orders, track repeat purchases, and reorder with just a few taps.
- **Notifications & Reminders**: Sends timely push notifications for order updates, such as delivery confirmation or payment reminders.
- **Settings & Personalization**: Users can customize notification preferences, update their profile, and change account settings.
- **Carousel and Product Adapters**: Enhances the visual appeal with interactive image carousels and dynamic UI components for displaying cake options.
- **Lottie Animations**: Provides engaging, modern animations to enhance user interaction and keep the app visually appealing.

The Cake Ordering Application integrates all necessary features to provide a hassle-free, personalized cake ordering experience for customers, from browsing to payment and delivery tracking, making it a one-stop solution for cake enthusiasts.

4.1 METHODOLOGY

The methodology adopted for developing the **Cake Ordering Application** follows a modular and iterative approach, ensuring flexibility, scalability, and efficient development. The app is built using **Android Studio**, with **Kotlin** as the programming language and **XML** for designing the user interface. Each feature, such as cake selection, order customization, checkout, and delivery tracking, is developed as a separate activity, and Intents are used for smooth navigation between modules. The app's activities are registered in the **AndroidManifest.xml** file, ensuring proper handling of each screen transition. Emphasis is placed on **user experience**, **performance**, and **security**, with a focus on ease of use and engaging UI elements, including Lottie animations for smooth visual transitions.

1. Requirements Gathering

- The **requirements gathering** phase began with identifying the target users of the Cake Ordering Application. This included cake enthusiasts, people planning special occasions like birthdays or weddings, and customers who want to order cakes for delivery. User feedback was gathered using methods such as surveys, focus groups, and competitor analysis to identify common pain points in online cake ordering, such as confusion around cake customization options, missed orders, and long delivery times.
- Based on this analysis, the functional requirements were outlined, which included features such as cake selection, customizations for flavors and toppings, delivery scheduling, and payment processing. Non-functional requirements such as usability, performance, security, and responsive design were also considered to ensure the app's smooth operation.

- The gathered requirements were documented in a **Software Requirements Specification (SRS)** document to guide the development process. Continuous feedback was incorporated into the development cycle to ensure the app met the users' needs.

2. System Design and Architecture

The **System Design and Architecture** of the Cake Ordering Application follows a modular and usercentric approach. The app uses **Android Studio** with **Kotlin** for the backend logic and **XML** for the user interface. The architecture follows the **Model-View-Controller (MVC)** design pattern, separating the system into three core components:

1. **Model**: This layer handles data management and business logic. It includes classes for managing user profiles, cake details, orders, payment processing, and delivery information. The **Model** ensures data integrity and consistency.
2. **View**: The user interface is designed using **XML** layouts and **Material Design** principles to create a visually appealing and responsive design. The **View** interacts with users, displaying data such as cake options, order status, and payment confirmation. Components like Lottie Animations and dynamic adapters enhance the interactivity of the application.
3. **Controller**: This layer handles user input, processes it, and updates the **View**. Each feature (e.g., cake selection, order customization, checkout) is implemented as an individual **activity** or **fragment**, allowing for clear separation of concerns and easy maintenance. **Intents** are used to navigate between activities and pass data.

3. Development Process

The development process for the Cake Ordering Application followed an **iterative approach** with an emphasis on **Agile** principles, allowing flexibility and the ability to adapt to user feedback. The project was divided into multiple stages:

1. **Requirements Gathering**: Target user needs were identified through surveys, interviews, and competitor analysis.
2. **System Design**: The app's architecture, database structure, and UI layouts were planned out, and the team decided to follow a modular design for scalability.
3. **Core Module Development**: Development started with key modules, such as user authentication (Login/Signup), followed by cake browsing and selection, order customization, payment processing, and notifications.
4. **Feature-Specific Development**: The team focused on smaller sprints to implement individual features, like checkout, order confirmation, and delivery tracking. Each feature was tested and validated after completion to ensure that it integrated smoothly with other parts of the app.
5. **UI/UX Design**: Interactive features like Lottie animations, dynamic cake carousel, and responsive designs were integrated to improve the user experience.
6. **Version Control**: **Git** was used for version control to manage the codebase and collaborate effectively. Continuous feedback was gathered from users, and the team adjusted features accordingly.
7. **Testing**: Unit testing and integration testing were conducted on individual features, ensuring that the app functions correctly. Performance testing was done to ensure that the app could handle multiple simultaneous orders without significant delays.
8. **Deployment**: Once the app was finalized and passed the testing stages, it was deployed to the **Google Play Store** for users to download. Regular updates and bug fixes were provided postdeployment.

4. Testing and Quality Assurance

Testing and quality assurance were critical in the development process to ensure that the Cake Ordering Application functions reliably and securely:

- **Unit Testing**: Each module, like the payment gateway, order tracking, and notifications, was tested individually to ensure correctness.
- **Integration Testing**: After unit testing, the integration of all modules was tested to ensure that they worked seamlessly together. For example, ensuring that the checkout process correctly reflects the total price and delivery details.
- **UI Testing**: Users tested the app's interface to ensure that the app was intuitive, easy to navigate, and free of usability issues. Feedback was used to refine the design.
- **Performance Testing**: The app's performance was tested by simulating high traffic, ensuring that the app can handle multiple orders, especially during peak hours, without any lag or crashes.
- **Security Testing**: Security tests were conducted to ensure safe transactions, including encryption of user data and payment details. Secure authentication methods were implemented for login/signup processes.
- **Regression Testing**: New updates or bug fixes were tested to ensure that they did not break the functionality of previously working features.

5. Deployment and Maintenance

Once the app passed all the necessary testing stages, it was deployed to the **Google Play Store**. During the deployment phase, the app was optimized for various Android devices, ensuring that it was compatible with a wide range of screen sizes and Android versions.

Post-deployment, the app entered the maintenance phase, where updates were pushed regularly to fix bugs, improve performance, and add new features based on user feedback. Regular integration of thirdparty APIs (e.g., payment gateway) and updates to improve security were part of ongoing maintenance. **6. Iterative Improvements**

The Cake Ordering Application followed an **iterative improvement** approach, which focused on refining the app based on real-world feedback. Post-launch updates included:

- Enhanced cake customization options.
- Additional payment methods.
- Improved user interface for better cake browsing experience. Real-time order tracking and notification improvements.

Each update was aimed at making the app more intuitive, responsive, and feature-rich, allowing the app to continuously adapt to users' changing needs and preferences.

5.CONCLUSION

In conclusion, the Cake Ordering App offers an efficient and user-friendly solution for cake enthusiasts and individuals looking for personalized cake orders. Developed using Android technologies such as Kotlin and XML, the app integrates key features like cake customization, seamless ordering, real-time order tracking, secure payment processing, and reminders. The app is designed to address common pain points like complicated ordering processes and missed order deliveries, providing a smooth and enjoyable experience for users.

The modular architecture of the app ensures scalability and ease of maintenance, making it possible to introduce new features based on user feedback. The app prioritizes user security, employing secure authentication mechanisms to safeguard user data and protect financial transactions. Rigorous testing

and quality assurance throughout the development process ensure the app’s reliability, ensuring it functions smoothly across devices and meets user expectations.

By continuously improving and introducing new features, such as loyalty programs or multi-order capabilities, the Cake Ordering App is well-positioned to evolve with the needs of its users. The app aims to make cake ordering as convenient, personalized, and enjoyable as possible, making it a valuable tool for both casual cake buyers and special occasion celebrations.

5.1 GENERAL

The Cake Ordering App is a fully-featured Android-based mobile application developed using Kotlin and XML, specifically designed to offer a seamless and personalized cake ordering experience. The app caters to a wide range of customer preferences by offering predefined cake types, sizes, and toppings, as well as the option for full cake customization. This allows users to select from an array of flavor options, icing styles, and decorative elements, or to upload their own custom cake designs for special events like birthdays, weddings, and anniversaries. This flexibility makes the app suitable for both standard and personalized cake orders, meeting the needs of all types of celebrations.

One of the key features of the app is the intuitive and efficient shopping cart functionality. Users can easily add cakes or other items to their cart, modify quantities, and review their selections before proceeding to checkout. The app supports various secure payment methods, ensuring a safe and hasslefree transaction process. After an order is confirmed, the app immediately notifies users with an order confirmation and updates regarding the status of their cake, providing real-time tracking throughout the baking and delivery process. This feature enhances transparency and helps manage customer expectations regarding delivery times.

In addition to cake selection and ordering, the app personalizes the user experience by offering cake suggestions based on the user’s past orders and the occasion. For example, if a user frequently orders cakes for birthdays, the app may suggest trending birthday cake designs or seasonal specials. It also

allows users to set reminders for upcoming events, ensuring they don’t forget to place orders in advance. These notifications are sent in a timely manner to remind users to complete their purchases, further improving convenience.

In summary, the Cake Ordering App is not only a convenient and flexible platform for ordering cakes but also a highly personalized service that adapts to users' preferences. Its comprehensive features, seamless ordering process, and commitment to customer satisfaction make it an invaluable tool for cake enthusiasts and those looking to celebrate special occasions with ease.

APPENDICES

SOURCE CODE

```
package com.example.cakeapp  import
android.os.Bundle import android.widget.* import
androidx.appcompat.app.AppCompatActivity


class MainActivity : AppCompatActivity() {


    private lateinit var cakeTypeSpinner: Spinner
private lateinit var quantityEditText: EditText private
lateinit var nameEditText: EditText    private lateinit
var phoneEditText: EditText    private lateinit var
```

```
addressEditText: EditText     private lateinit var
whippedCreamCheckBox: CheckBox     private lateinit
var chocolateToppingCheckBox: CheckBox     private
lateinit var orderButton: Button     private lateinit var
orderSummaryTextView: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)


        initializeViews()
setupCakeTypeSpinner()
setupOrderButton()
    }



    private fun initializeViews() {         cakeTypeSpinner =
findViewById(R.id.cakeTypeSpinner)         quantityEditText =
findViewById(R.id.quantityEditText) nameEditText =
findViewById(R.id.nameEditText) phoneEditText =
findViewById(R.id.phoneEditText) addressEditText =
findViewById(R.id.addressEditText)
whippedCreamCheckBox =
findViewById(R.id.whippedCreamCheckBox)
chocolateToppingCheckBox =
findViewById(R.id.chocolateToppingCheckBox)
orderButton = findViewById(R.id.orderButton)
```

```
orderSummaryTextView =
findViewById(R.id.orderSummaryTextView)
    }


    private fun setupCakeTypeSpinner() {         val
cakeTypes = arrayOf(
getString(R.string.cake_type_select),
getString(R.string.cake_type_chocolate),
getString(R.string.cake_type_vanilla),
getString(R.string.cake_type_red_velvet),
getString(R.string.cake_type_cheesecake)
        )


        val adapter = ArrayAdapter(

            this,

            android.R.layout.simple_spinner_item,
cakeTypes
        )


        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
        cakeTypeSpinner.adapter = adapter

    }
```

```
   private fun setupOrderButton() {
orderButton.setOnClickListener {            if
(validateInputs()) {                val orderSummary =
createOrderSummary()
orderSummaryTextView.text = orderSummary
            Toast.makeText(this, getString(R.string.order_success), Toast.LENGTH_SHORT).show()

        }

    }

   }



   private fun validateInputs(): Boolean {        if
(cakeTypeSpinner.selectedItemPosition == 0) {
        Toast.makeText(this, getString(R.string.error_select_cake), Toast.LENGTH_SHORT).show()

        return false

    }



    val quantityStr = quantityEditText.text.toString()
if (quantityStr.isEmpty() || quantityStr.toInt() <= 0) {
         Toast.makeText(this, getString(R.string.error_quantity), Toast.LENGTH_SHORT).show()

        return false
```

```
    }  if
    (nameEditText.text.toString().isEmpty()) {
       Toast.makeText(this, getString(R.string.error_name), Toast.LENGTH_SHORT).show()
        return false
    }

    if (phoneEditText.text.toString().isEmpty()) {
        Toast.makeText(this, getString(R.string.error_phone), Toast.LENGTH_SHORT).show()
        return false
    }

    if (addressEditText.text.toString().isEmpty()) {
        Toast.makeText(this, getString(R.string.error_address), Toast.LENGTH_SHORT).show()
        return false
    }

    return true
}
```

```
    private fun createOrderSummary(): String {        val
cakeType = cakeTypeSpinner.selectedItem as String val quantity
= quantityEditText.text.toString().toInt() val name
= nameEditText.text.toString()

        val phone = phoneEditText.text.toString()
    val address = addressEditText.text.toString()


        val                price                =                when                {
cakeType.contains(getString(R.string.cake_type_chocolate))   ->       10
cakeType.contains(getString(R.string.cake_type_vanilla)) -> 8
cakeType.contains(getString(R.string.cake_type_red_velvet)) -> 12
cakeType.contains(getString(R.string.cake_type_cheesecake)) -> 15             else -> 0

        }



        var total = price * quantity



        val toppings = StringBuilder()        if
(whippedCreamCheckBox.isChecked) {
            total += 2 * quantity
toppings.append(getString(R.string.topping_whipped_cream)).append("\n")
        }

        if (chocolateToppingCheckBox.isChecked) {
```

```
        total += 3 * quantity
    toppings.append(getString(R.string.topping_chocolate)).append("\n")
    } if
    (toppings.isEmpty()) {
    toppings.append(getStri
    ng(R.string.no_toppings
    ))

    }


    return """

===== ${getString(R.string.order_summary_title)} =====

${getString(R.string.label_name)}: $name

${getString(R.string.label_phone)}: $phone

${getString(R.string.label_address)}: $address



${getString(R.string.label_cake_type)}: $cakeType

${getString(R.string.label_quantity)}: $quantity



${getString(R.string.label_toppings)}:

$toppings  // Removed curly braces
```

```
    ${getString(R.string.label_total)}: \$$total

""".trimIndent()

    }

}
```

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" tools:context=".MainActivity">

    <!-- Background Image with proper contrast -->

    <ImageView        android:layout_width="match_parent"
android:layout_height="match_parent"
android:src="@drawable/bg"        android:scaleType="centerCrop"
android:alpha="0.7"
android:contentDescription="@string/background_description" />


    <ScrollView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:padding="16dp"
android:fillViewport="true">
```

```
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical" android:padding="16dp"
android:background="#EEFFFFFF" <!-- More opaque for
better contrast --> android:elevation="4dp">

<!-- App Title --> <TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/app_name" android:textSize="24sp"
android:textColor="#FF000000" <!-- Pure black for best contrast -->
android:textStyle="bold" android:gravity="center"
android:layout_marginBottom="16dp"/>

<!-- Form Elements with Improved Contrast -->

<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/cake_type_label"
android:textColor="#FF000000"
android:layout_marginBottom="4dp"/>

<Spinner

android:id="@+id/cakeTypeSpinner"
```

```
android:layout_width="match_parent"
android:layout_height="56dp"
android:minHeight="48dp"
android:background="#FFFFFFFF"
android:layout_marginBottom="16dp"/>


        <!-- Quantity Input with proper contrast -->

        <TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/quantity_label"
android:textColor="#FF000000"
android:layout_marginBottom="4dp"/>             <EditText
android:id="@+id/quantityEditText"
android:layout_width="match_parent"
android:layout_height="56dp"
android:minHeight="48dp"        android:inputType="number"
android:importantForAutofill="yes"
android:textColor="#FF000000"
android:background="#FFFFFFFF"
android:hint="@string/hint_quantity"
android:hintTextColor="#80000000" <!-- Semitransparent black
-->
            android:layout_marginBottom="16dp"/>


        <!-- Customer Details Section -->
```

```
        <TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/customer_details"
android:textSize="18sp"
android:textColor="#FF000000"
android:textStyle="bold"
android:layout_marginBottom="8dp"/>


        <!-- Name Input with proper contrast -->
<TextView               android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/name_label"
android:textColor="#FF000000"
android:layout_marginBottom="4dp"/> <EditText
android:id="@+id/nameEditText"
android:layout_width="match_parent"
android:layout_height="56dp"
android:minHeight="48dp"
android:inputType="textPersonName"
android:importantForAutofill="yes"
android:textColor="#FF000000"
android:background="#FFFFFFFF"
android:hint="@string/hint_name"
android:hintTextColor="#80000000"
android:layout_marginBottom="16dp"/>
```

```
        <!-- Phone Input with proper contrast -->

        <TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/phone_label"
android:textColor="#FF000000"
android:layout_marginBottom="4dp"/>

        <EditText android:id="@+id/phoneEditText"
            android:layout_width="match_parent"
            android:layout_height="56dp"
            android:minHeight="48dp"
            android:inputType="phone"
android:importantForAutofill="yes"
android:textColor="#FF000000"
android:background="#FFFFFFFF"
android:hint="@string/hint_phone"
android:hintTextColor="#80000000"
android:layout_marginBottom="16dp"/>

        <!-- Address Input with proper contrast -->

        <TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/address_label"
android:textColor="#FF000000"
android:layout_marginBottom="4dp"/>
```

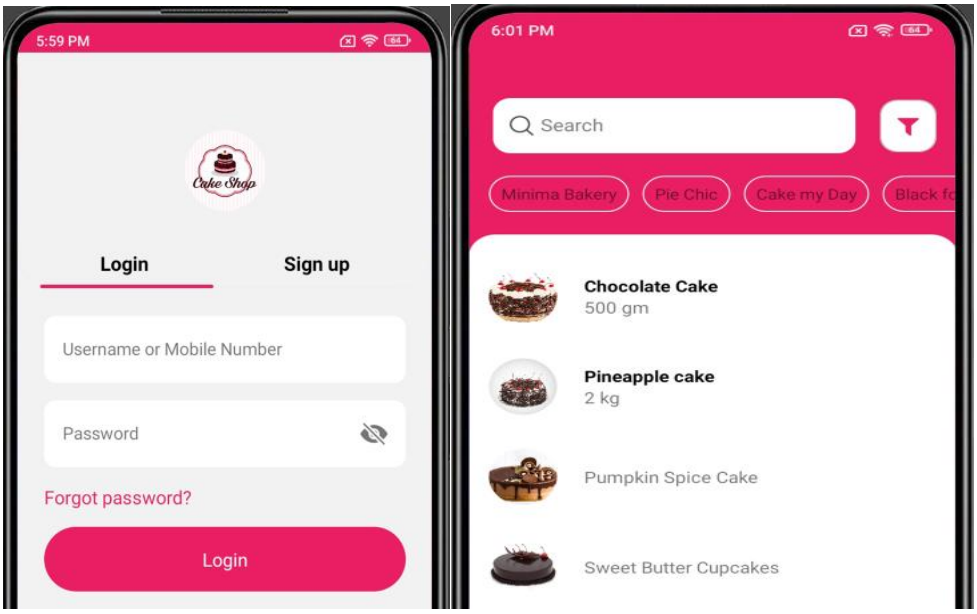
```
        <EditText

            android:id="@+id/addressEditText"
            android:layout_width="match_parent"
            android:layout_height="56dp"
            android:minHeight="48dp"
            android:inputType="textPostalAddress"
            android:importantForAutofill="yes"
            android:textColor="#FF000000"
            android:background="#FFFFFFFF"
            android:hint="@string/hint_address"
            android:hintTextColor="#80000000"
            android:layout_marginBottom="16dp"/>


        <!-- Toppings Section with improved CheckBoxes -->

        <TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/toppings_label"
android:textSize="18sp"
android:textColor="#FF000000"
android:textStyle="bold"
android:layout_marginBottom="8dp"/>


        <!-- CheckBox with proper contrast and touch target -->
```

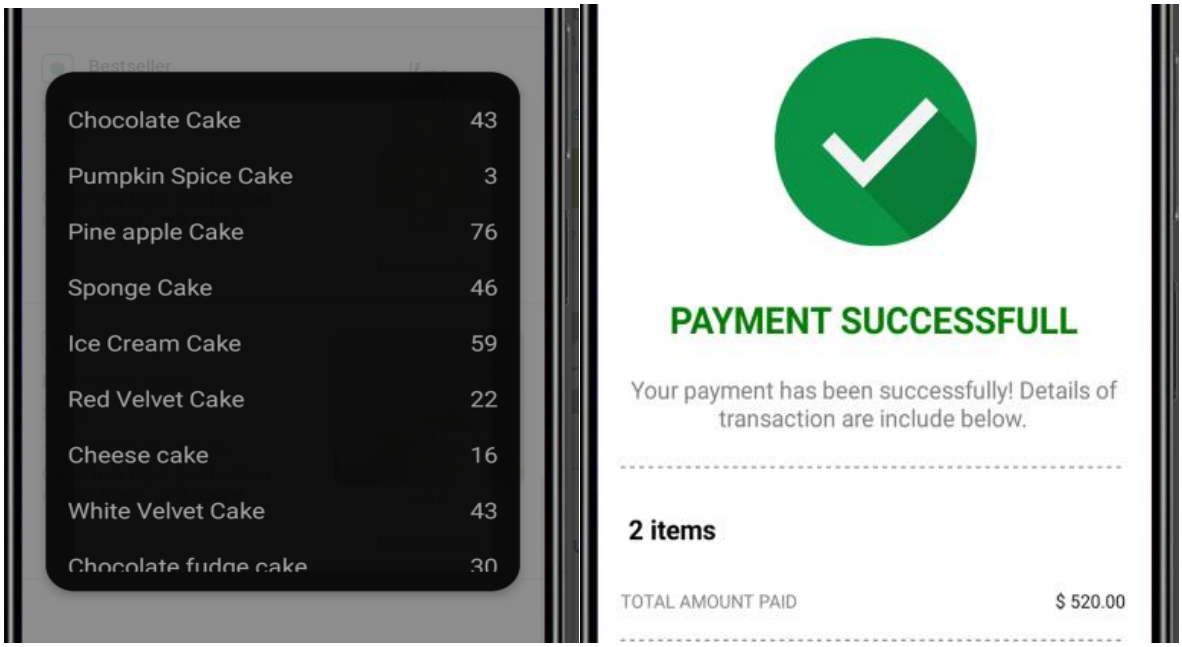
```
        <LinearLayout android:layout_width="match_parent"
            android:layout_height="72dp" <!-- Increased touch area -->
            android:orientation="horizontal"
            android:gravity="center_vertical" android:padding="8dp">


            <CheckBox
android:id="@+id/whippedCreamCheckBox"
android:layout_width="wrap_content"
android:layout_height="48dp"
android:minWidth="48dp"                    android:minHeight="48dp"
android:text="@string/topping_whipped_cream"                    android:textColor="#FF000000" <!-- Pure black
-->                    android:buttonTint="@color/purple_500"/>


        </LinearLayout>


        <LinearLayout
android:layout_width="match_parent"
android:layout_height="72dp"
android:orientation="horizontal"
android:gravity="center_vertical"
android:padding="8dp">

            <CheckBox
android:id="@+id/chocolateToppingCheckBox"
android:layout_width="wrap_content"
android:layout_height="48dp"
```

```
android:minWidth="48dp"                android:minHeight="48dp"
android:text="@string/topping_chocolate"
android:textColor="#FF000000"
android:buttonTint="@color/purple_500"
android:layout_marginBottom="16dp"/>
        </LinearLayout>



        <!-- Order Button -->

        <Button
android:id="@+id/orderButton"
android:layout_width="match_parent"
android:layout_height="56dp"
android:minHeight="48dp"
android:text="@string/button_order"
android:textAllCaps="false"
android:textColor="#FFFFFFFF"
android:backgroundTint="@color/purple_500"
android:layout_marginBottom="16dp"/>

        <!-- Order Summary -->

        <TextView
android:id="@+id/orderSummaryTextView"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/default_summary"
android:textSize="16sp"
```

```
android:textColor="#FF000000"
android:padding="12dp"
android:background="#FFFFFFFF"/>
        </LinearLayout>

    </ScrollView>

</FrameLayout>
```

OUTPUT SCREENSHOTS





REFERENCES

1. **Android Developers.** (2021). *Building Your First App*. from Retrieved https://developer.android.com/training/basics/firstapp
2. **Kotlinlang.** (2021). *Kotlin Programming Language*. from Retrieved https://kotlinlang.org/docs/home.html
3. **Firebase.** (2021). *Firebase Authentication*. Retrieved from https://firebase.google.com/docs/auth
4. **Google.** (2021). *Material Design*. Retrieved from https://material.io/design
5. **Android Developers.** (2021). *Android Studio Overview*. Retrieved from https://developer.android.com/studio
6. **Singh, J.** (2020). *Building E-Commerce Apps in Android with Kotlin*. Medium. Retrieved from https://medium.com/swlh/building-an-e-commerce-app-with-androidand-kotlind0b8c68f4b60
7. **Udemy.** (2020). *Android App Development with Firebase in Kotlin*. Retrieved from https://www.udemy.com/course/android-app-development-with-firebase-in-kotlin/
8. **Android Developers.** (2021). *Jetpack Libraries*. Retrieved from https://developer.android.com/jetpack
9. **Ahsan, N.** (2019). *Developing a Cake Delivery App for Android*. Medium. Retrieved from https://medium.com/@ahsan007/developing-a-cake-delivery-app-forandroid561fc7c8b03
10. **Firebase.** (2021). *Firebase Cloud Messaging (FCM)*. Retrieved from https://firebase.google.com/docs/cloud-messaging