

**Java Institute For Advanced Technology**

Department of Examinations



|  |  |
| --- | --- |
| COURSE(S) – (LEADING TO) | BSC (Hons) IN SOFTWARE ENGINEERING |
| UNIT NAME | HANDHELD DEVICE PROGRAMMING II |
| UNIT ID | JIAT/HHDPII |
| EXAMINATION ID | JIAT/HHDPII |
| NIC | 200128603662 |
| NAME | K.G PASINDU CHAMIKA |

**FINAL PROJECT DOCUMENTATION**

Project Name:

**Floral Fete**

by

**“K.G Pasindu Chamika”**

1. **Acknowledgments and Abstract**

First I would like to sincerely thank Dr. Tharaka Sankalpa for his tremendous help and guidance during this subject. I also want to express my gratitude to my friends at Java Institute for their support and wise counsel. Also I would like to thank my family for their consistent understanding and support throughout the development of my project. Lastly, I want to express my gratitude to the Firebase and Google Maps teams, as well as the open-source community, for providing the resources and tools that helped this project.

1. **Abstract**

Floral Fete is an Android m-commerce application that built for the sale and management of fresh flowers business. The application features user friendly interfaces for browsing products, placing orders, and tracking orders. The admin panel has efficient order, product, user management, and dashboard to show business statistics. This application also include functionalities like secure user authentication, a dynamic product search, real time order status update notifications using Firebase Cloud Messaging(FCM),Firebase cart and SQLite cart, shake your mobile to add items to wishlist using sensor framework in android and Google maps integrated into the app to locate Floral Fete head office. Additionally, the application incorporates UI animations such as a fling animation for a improve UX. A robust "Forgot Password" mechanism is utilizing a Java EE backend for email-based code verification. This project use Firebase as the main backend and Java EE backend as the secondary. This approach ensures to support both user and admin functionalities while maintaining scalability and security. In addition to discussing future enhancements to this application for increasing user experience and operational efficiency, this document describes the application's motivation, design, implementation, and evaluation.

1. **Table of Contents**

[**1. Introduction** 6](#_Toc191313013)

[**Overview:** 6](#_Toc191313014)

[**1.1 Motivation** 6](#_Toc191313015)

[**1.2 Problem Statement** 6](#_Toc191313016)

[**2. Background** 7](#_Toc191313017)

[**2.1 Context** 7](#_Toc191313018)

[**2.2 Industry Challenges** 7](#_Toc191313019)

[**2.3 Related Work and Technical Foundations** 7](#_Toc191313020)

[**2.4 Technological Rationale** 8](#_Toc191313021)

[**3. Specification** 8](#_Toc191313022)

[**3.1 Functional Requirements** 8](#_Toc191313023)

[**3.2 Non-Functional Requirements** 9](#_Toc191313024)

[**4. Design** 10](#_Toc191313025)

[**4.1 System Architecture** 10](#_Toc191313026)

[**4.2 Data Model** 12](#_Toc191313027)

[**4.3 UI/UX Design** 14](#_Toc191313028)

[**4.4 Design Decisions** 15](#_Toc191313029)

[**5. Implementation** 16](#_Toc191313030)

[**5.1 Development Process** 16](#_Toc191313031)

[**5.2 Key Modules and Their Implementation** 16](#_Toc191313032)

[**5.2.1 User Interface (Android App)** 16](#_Toc191313033)

[**5.2.2 Firestore and Real-Time Data Synchronization** 18](#_Toc191313034)

[**5.2.3 Admin Authentication and Forgot Password (Java EE Backend)** 18](#_Toc191313035)

[**5.2.4 Push Notifications and Order Updates** 18](#_Toc191313036)

[**5.3 Implementation Challenges and Solutions** 20](#_Toc191313037)

[**5.4 Summary** 20](#_Toc191313038)

[**6. Results and Evaluation** 21](#_Toc191313039)

[**6.1 Functional Evaluation** 21](#_Toc191313040)

[**6.2 Performance Evaluation** 21](#_Toc191313041)

[**6.3 Usability Testing** 22](#_Toc191313042)

[**6.4 Limitations and Challenges** 22](#_Toc191313043)

[**6.5 Evaluation Metrics** 22](#_Toc191313044)

[**7. Future Work** 23](#_Toc191313045)

[**7.1 Enhancing Scalability and Performance** 23](#_Toc191313046)

[**7.2 Security Enhancements** 23](#_Toc191313047)

[**7.3 Feature Extensions** 23](#_Toc191313048)

[**7.4 UI/UX Improvements** 23](#_Toc191313049)

[**7.5 Future Admin Features** 24](#_Toc191313050)

[**7.6 Integration with Emerging Technologies** 24](#_Toc191313051)

[**8. Conclusions** 24](#_Toc191313052)

[**9. Reflection on Learning** 25](#_Toc191313053)

[**10. References and Bibliography:** 26](#_Toc191313054)

[**10.1 References** 26](#_Toc191313055)

[**10.2 Bibliography** 26](#_Toc191313056)

1. **List of Tables, Figures, and Abbreviations**
2. **List of Figures**

[Figure 1:Data Flow of the frontend and backend for the Push Notification 11](#_Toc191313796)

[Figure 2:Class Diagram for Firestore Collections 13](#_Toc191313797)

[Figure 3:UI Flow Diagram showing user navigation and key screens 14](#_Toc191313798)

# **1. Introduction**

**Overview:**  
Floral Fete is an Android based mobile commerce application that aims to transform the way people buy fresh flowers. The application serves two core audiences: end users who browse and order flowers, and administrators who handle goods, orders, and customer data. Floral Fete is a simple, user-friendly layout that incorporates with current features like real-time order tracking, secure authentication, dynamic product search, separate carts for guest and logged users ,push notification for order status updates. In addition, this application includes a smooth launch animation and transition animations enhancing the UI. Floral Fete has a robust backend support for the administrative panel. For backend we used Firebase and Java EE ensuring secure and responsive UX.

**1.1 Motivation**  
Floral Fete was founded because the retail floral industry needed an effective, user-friendly solution. Current consumer expectations are not being satisfied by traditional flower ordering systems, which frequently involve in-person visits or antiquated websites. Modern customers need a quick, accessible, convenience  and reliable services that allow them to place orders from their mobile devices. Floral Fete’s goal is to simplify the entire ordering process, improve the user experience, and give administrators with powerful tools for effectively managing their business operations by integrating latest mobile technology and cloud services.

**1.2 Problem Statement**  
Many floral industry solutions now in use are lacking comprehensive, integrated system that considers both admin and customer needs despite the widespread usage of mobile applications. Main difficulties include:

* **Inefficient Order Processes:** Difficult ordering processes are a common feature of current systems in floral industry. which can cause delays in orders that leading to unhappy customers.
* **Separate Management:** Using separate platforms that don't interact well, administrators find it difficult to manage orders, inventories, and customer data
* **Limited Real Time Updates:** In some applications customers rarely receive real time notifications about their order status which causes a lack of trust and uncertainty about the business.
* **User Experience Problems:** A lot of the application that use today are lack an interesting or user friendly interface. which can make the whole buying experience less enjoyable for the customers. That can lead to less customers

Floral Fete is designed to solve these problems by offering a consistent platform that simplifies customer orders and gives admins strong, real-time capabilities for handling all transactions.

# **2. Background**

**2.1 Context**  
In the floral industry to handle orders, inventory control, and customer interaction, the floral business has always depended on either physical stores or outdated web technologies. Customers expect to have fast, convenient, and real-time solutions for purchasing everyday goods including flowers thanks to the increasing use of smartphones and mobile applications.. This change has created a demand for modern mobile commerce solutions such as m-commerce applications that not simplify the purchasing process. Also provide a robust platform for floral business management.

**2.2 Industry Challenges**  
As noted in the Problem Statement, traditional floral ordering systems suffer from inefficiencies and disjointed customer engagement. In this section, we go into additional detail about these difficulties by looking at the existing gaps in technology and investigating how modern solutions, such as mobile-first design and real-time data synchronization, may help.

**2.3 Related Work and Technical Foundations**  
Several mobile commerce platforms have try to address these issues. But few provide a fully integrated solution that covers both the customer and administrative sides. Notable examples include:

* **E-commerce Applications:** Major e-commerce applications usually include real-time notifications and smooth order tracking, their approaches might not be suited for some industries like floral industry.
* **Custom Web Solutions:** Some businesses build custom web portals, but they not meet modern, user friendly interfaces expected by today's customers.

Floral Fete uses a variety of modern technology to overcome these obstacles:

* **Android Platform:** For a responsive and easy to use interfaces that supports real time updates and smooth animations.
* **Firebase Firestore:** For scalable, real-time data storage and synchronization between users and administrators.
* **SQLite**: For handing the “Cart” of users who are not sign in to the application yet.
* **Firebase Cloud Messaging (FCM):** For sending push notifications to customers when there are changes to their orders status.
* **Java EE Backend:** For handling login functions and functions like forgot password to improve their security.

**2.4 Technological Rationale**  
The following reasons influenced the selection of above metioned technologies:

* **Real-Time Data Synchronization:** Firestore’s real time features ensure that both users and administrators have access to most recent data available.
* **Scalability:** Firebase services are designed to scale with the application by handling increasing numbers of users and orders without significant re plan or re design.
* **Improve User Engagement:** A better user experience is provided by features like push notifications ,user friendly UI animations, “shake your mobile to add product to wishlist” function, separate carts like SQLite cart and firestore cart to sync data across the logged and not logged users.

# **3. Specification**

## **3.1 Functional Requirements**

* **User Registration and Sign in:**
  + User should be able to register an account and log in using their email and password.
  + The application shall support secure login and provide a "Forgot Password" option for user that can use to reset their passwords through email based verification code.
* **Product Browsing and Search:**
  + The application should display a catalogue of flowers with details. It should include images, names, prices, and product descriptions.
  + The application should include a search option allowing users to filter products by name, category, and occasion and simply search by typing the text
* **Shopping Cart and Order Management:**
  + Users should be able to add products to a cart and adjust quantities if they want before checkout part.
  + Users can add items to cart even if they are not signed in to the application
  + Application should sync not signed in user cart data to firestore when user sign in.
  + The system should process orders and update order statuses.
* **Order history Management:**
  + User should be able to look in to their order history and give feedbacks to their orders. They also can update the feedback they gave earlier.
* **Wishlist Functionality:**
  + Users should be able to add products to a wishlist. Each user can maintain a their own list of favorite products. which can remove items/prodcuts if they prefer.
  + Users can shake their mobile device in the Single Product View page to add product to the wishlist
* **Push Notifications:**
  + The system should be able to send notifications to users when their order status is updated.
* **Admin Panel:**
  + Admins should have access to a their admin panel where they can view and manage orders, update order statuses, manage products ,view user data, change user status and display statistics of their business.
* **UI Animations:**
  + The application should include UI animations such as a fling, spring animations, Lottie animation to improve UX.

## **3.2 Non-Functional Requirements**

* **Performance and Scalability:**
  + The system should provide real time data using Firebase Firestore .
  + The system must scale to support an increasing number of users and transactions without significant performance drop.
* **Security:**
  + User data including passwords shall be securely stored in firestore.
  + All communication between the client and the server side should be secured using HTTPS.
  + Admin functions should be accessible only after proper authentication.
* **Usability:**
  + The user interface must be simple and easy to navigate with clear action buttons and feedback alerts for users.
  + The application should be optimized for mobile devices, ensuring a smooth and responsive UX.
* **Maintainability:**
  + When using cloud services, should follow best practices to simplify maintenance and updates.

# **4. Design**

## **4.1 System Architecture**

Floral Fete follows a multi-tier architecture that seamlessly integrates client-side and server-side components. The primary layers include:

* **Client Layer (Android App):**
  + The Android app is built using native android components and animation files such as Lottie animations to deliver a rich, responsive user interface.
  + It has features like product browsing, order placement, real-time status updates, and push notifications using FCM.
  + The app also includes engaging UI animations (e.g., fling animation on launch).
* **Data Layer (Firebase Firestore):**
  + Firestore is used as the primary data storage solution for real-time synchronization between users and administrators.
  + Collections include **Users, Products, Orders, Wishlist,** and **Feedback**.
* **Backend Layer:**
  + For push notifications (such as triggering notifications when order status changes), Firebase Cloud Functions or Activepieces are utilized to automatically respond to Firestore updates.
* **Communication:**
  + The Android app communicates with Firestore and FCM directly, while the admin panel communicates with the Java EE backend.
  + The overall system uses HTTPS to secure data exchange.

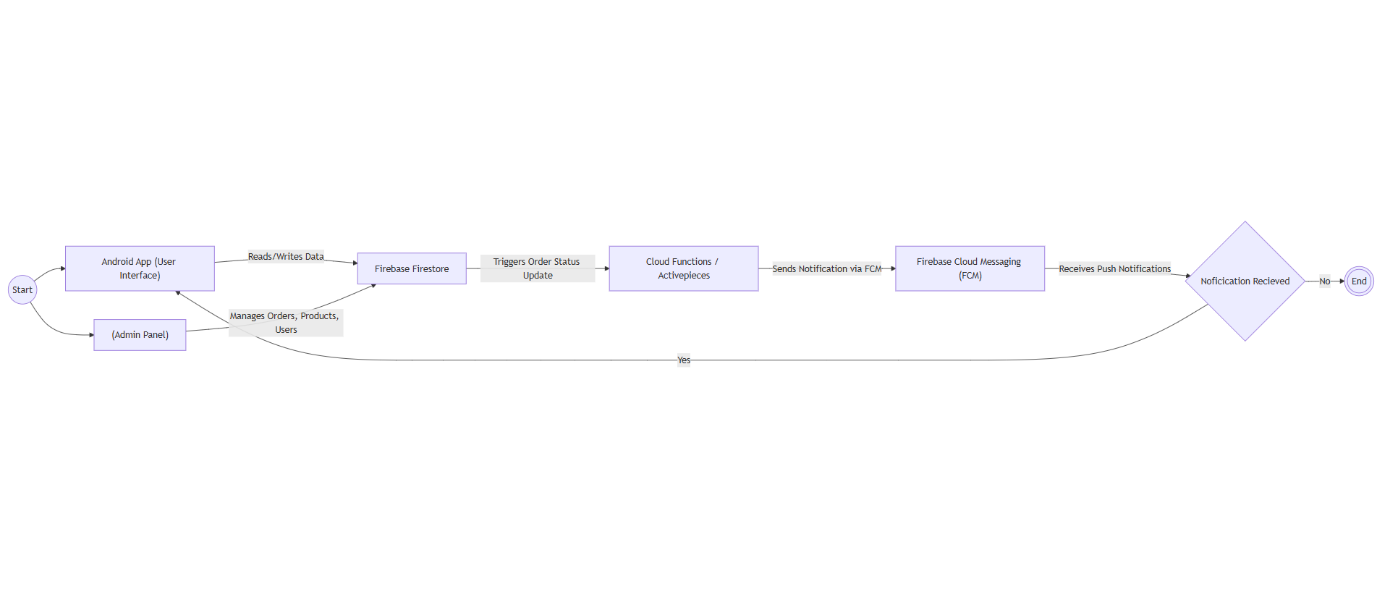


Figure 1:Data Flow of the frontend and backend for the Push Notification

## **4.2 Data Model**

The data model for Floral Fete is designed to support both user and administrative functionalities. Some of key collections in Firestore include:

* **Users:**
  + Fields: fname, lname, email, password, fcmToken.
* **Address**:
  + Fields: address, userId
* **Products:**
  + Fields: id, name, description, price, qty flowerTypeId, occasionIds, imageUrls(Array)
* **FlowerTypes**
  + Fields: name
* **Occations**
  + Fields: name
* **Orders:**
  + Fields: orderId, userId, products (Array), orderStatus (Processing, Packing, Shipped, Delivered), orderedDatetime, deliveryDate
* **Wishlist:**
  + Fields: userId, productId, productImageUrl, productName
* **Cart:**
  + Fields: items(collection)
* **Feedback:**
  + Fields: feedback, orderId, userId, username, productId.

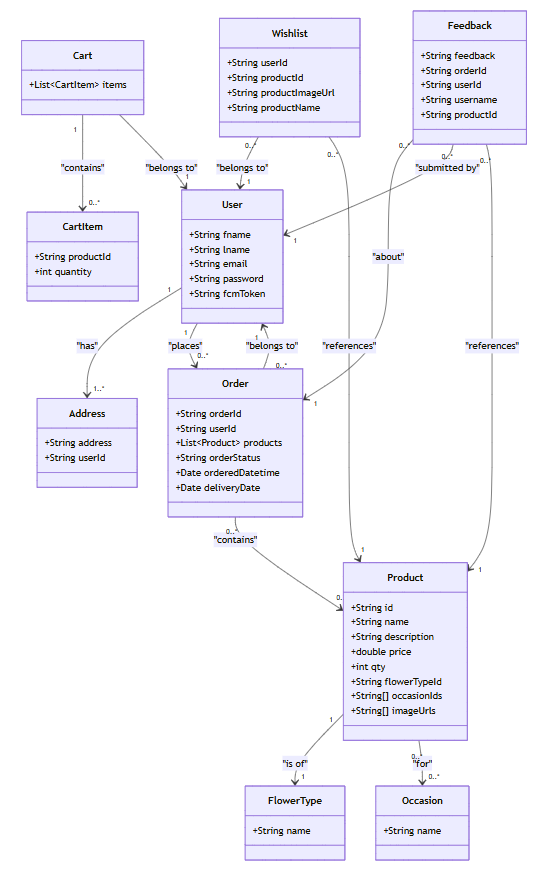


Figure 2:Class Diagram for Firestore Collections

## **4.3 UI/UX Design**

The design of Floral Fete is mainly focus on UI/UX ensuring a smooth and engaging experience:

* **User Interface :**
  + The Android app has a clean, modern layout using Bottom Navigation View for the main page.
  + Key screens include: Home, Single Product View, Search, Cart, Wishlist, Checkout, Orders, Order History, and Profile.
  + Animated transitions between bottom navigation view fragments to improve the user experience.
* **Admin Interface:**
  + The admin panel is designed using a navigation drawer with fragments for Dashboard, Manage Orders, Manage Products, and Manage Users.
  + Admin dashboard includes charts and summary statistics (bar charts, pie charts) to monitor business performance.
* **Interaction Flow:**
  + The ordering process is simplified from product selection to checkout.
  + Real time data updates using Firestore ensure that order status changes in real time.
  + Push notifications keep users informed of their order updates.

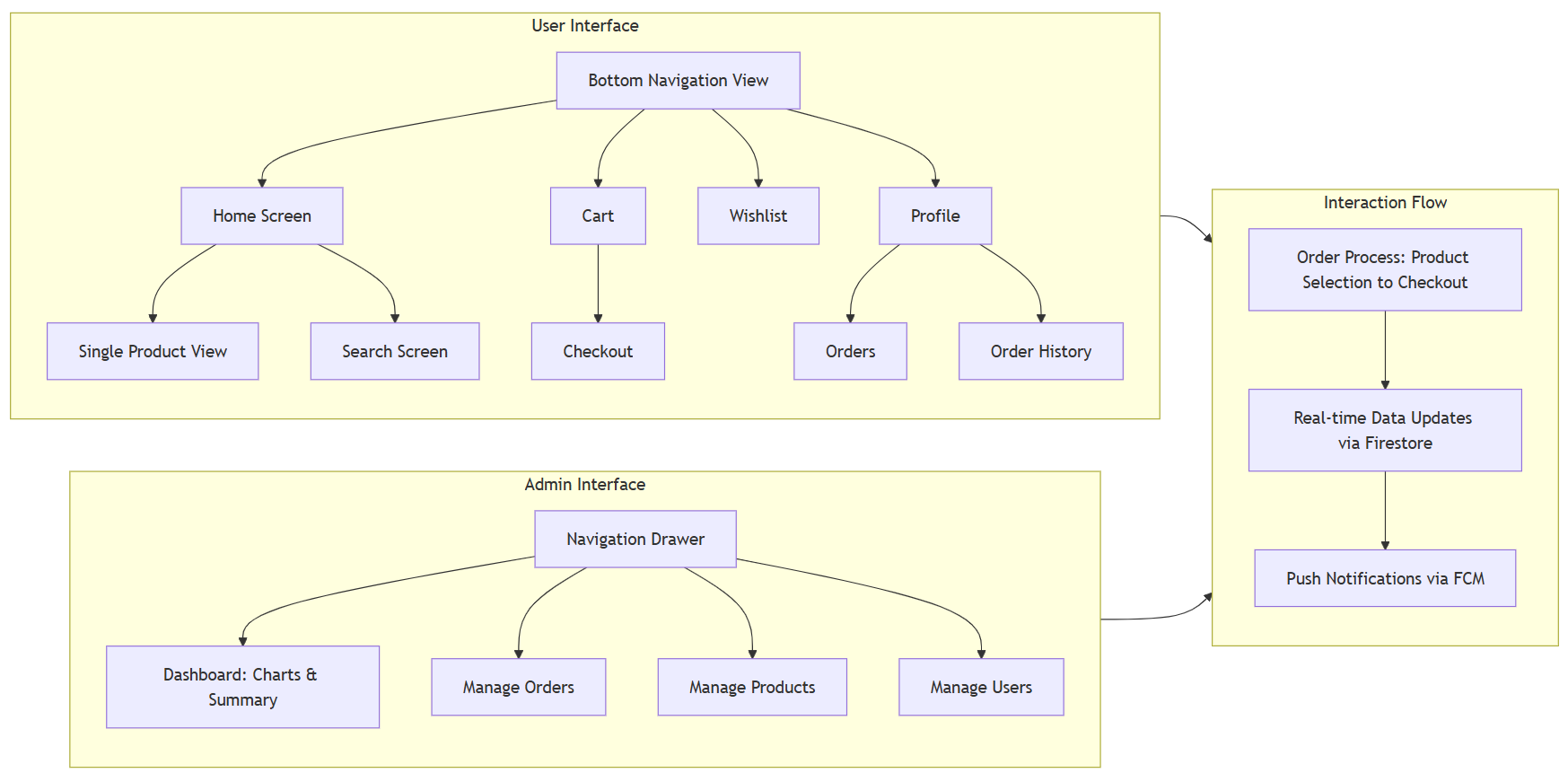


Figure 3:UI Flow Diagram showing user navigation and key screens

## **4.4 Design Decisions**

* **Real-Time Updates:**
  + The decision to use Firebase Firestore is made by the need of real time data synchronization. This ensures that users and administrators see the latest data without a delay.
* **Cloud Messaging for Notifications:**
  + FCM is integrated for push notifications, keeping users informed about order updates.
* **Java EE Backend for Admin Sign In and Forgot Password:**
  + The admin Sign In and Forgot password option relies on a Java EE backend ensuring a more robust security model.
* **UI Animations:**
  + Engaging animations (e.g., fling animation on app launch) to improve the user experience and make the app more feel dynamic.
* **Scalability and Maintainability:**
  + The architecture is designed to be modular, this clear separation between the client, data, and backend layers, making it easier to maintain and scale as the customer base grows.

# **5. Implementation**

## **5.1 Development Process**

While building Floral Fete the project followed an agile, iterative methodology. The project was divided into separate modules. With each module developed and integrated gradually. This included the user interface, Firestore, push notifications using FCM, and the Java EE backend for admin sign in and forgot password functionality. Regular code reviews helped keep the code quality while sticking to requirements.

## **5.2 Key Modules and Their Implementation**

### **5.2.1 User Interface (Android App)**

* **Home, Product, and Order Screens:**  
  The Android app was built using Android Studio. A Bottom Navigation View provides access to key screens such as Home, Single Product View, Search, Cart, Wishlist, Checkout, Orders, Order History, and Profile.
  + **UI Animations:**  
    Custom animations like fling animation that plays during app launch were created using Android’s animation framework. Also Lottie animations used to pages with empty views such as Cart, Wishlist.
  + **Real-Time Data Updates:**  
    The app uses Firebase Firestore’s snapshot listeners to update the UI in real time.
* **Dynamic Order Status Update Example:**  
  In the order management adapter, the order status button cycles through statuses as follows:

Java Code:

holder.statusButton.setOnClickListener(v -> {  
 String nextStatus = getNextStatus(order.getOrderStatus());  
 order.setOrderStatus(nextStatus);  
 holder.statusButton.setText(nextStatus);  
 firestore.collection("orders").document(order.getOrderId())  
 .update("orderStatus", nextStatus)  
 .addOnSuccessListener(aVoid -> {  
 *successCustomToast*("Order Updated",context);  
 notifyDataSetChanged();  
 })  
 .addOnFailureListener(e -> Toast.*makeText*(context, "Error updating order", Toast.*LENGTH\_SHORT*).show());  
});

The helper method getNextStatus() cycles through order status:  
Processing → Packing → Shipped → Delivered → Processing.

### **5.2.2 Firestore and Real-Time Data Synchronization**

* **Firestore Collections:**  
  The app uses Firestore for storing data for Users, Products, Orders, Wishlist, Cart, Feedback, etc.
* **Data Model:**  
  The Firestore data model is optimized to support quick reads and writes while minimizing redundant operations. Using composite keys (e.g., in the Wishlist collection) and Auto generated keys for uniqueness.

### **5.2.3 Admin Authentication and Forgot Password (Java EE Backend)**

* **Admin Sign-In and Forgot Password:**  
  The Java EE backend is used solely for handling admin authentication and the forgot password process.
  + **Admin Sign-In:**  
    A Java EE servlet processes JSON-formatted sign-in requests and validates admin credentials.
  + **Forgot Password:**  
    Another Java EE servlet handles forgot password requests by generating a random verification code, sending it via email using the JavaMail API, and returning the code to the Android app.
  + **Integration with Android:**  
    The Android app sends HTTP requests using OkHTTP to these servlets. Password recovery give a option to user to update the use password in Firestore if user successfully enter the code.

### **5.2.4 Push Notifications and Order Updates**

* **Push Notifications:**  
  Firebase Cloud Messaging (FCM) is integrated into the Floral Fete app. FMC to send push notifications. User’s FCM tokens are retrieved upon login and they stored in their Firestore document.
* **Cloud Functions for Notifications:**  
  A Firebase Cloud Function is set up to listen for updates in the Firestore orders collection. When an order status changes the function retrieves the relevant user’s FCM token from Firestore and sends a push notification to the user informing about the order status change.

**Javascript code:**

exports.sendOrderStatusUpdate = functions.firestore

  .document('orders/{orderId}')

  .onUpdate(async (change, context) => {

    const orderData = change.after.data();

    const userId = orderData.userId;

    const newStatus = orderData.orderStatus;

    try {

      const userDoc = await admin.firestore().collection('user').doc(userId).get();

      if (!userDoc.exists) {

        console.error(`No user found with ID: ${userId}`);

        return;

      }

      const fcmToken = userDoc.data().fcmToken;

      if (!fcmToken) {

        console.error(`No FCM token found for user ID: ${userId}`);

        return;

      }

      const message = {

        notification: {

          title: 'Order Status Updated',

          body: `Your order is now ${newStatus}`,

        },

        token: fcmToken,

      };

      await admin.messaging().send(message);

      console.log(`Notification sent to user ID: ${userId}`);

    } catch (error) {

      console.error('Error sending notification:', error);

    } });

## **5.3 Implementation Challenges and Solutions**

* **Real Time Data Handling:**  
  Firestore snapshot listeners were used to handle real time data updates. This required careful management of async callbacks.
* **Push Notification Reliability:**  
   It was difficult to integrate FCM and configure Firebase Cloud Functions. To make sure that alerts were consistently sent out when order statuses are changing. It was helped by a YouTube video (Automated Android Notifications with Firebase Cloud Functions, Messaging, and Firestore [Video], Jul 17,2021)
* **UI Responsiveness and Animations:**  
  It was necessary to make adjustments while implementing fluid animations (such as the fling animation) to allow for different screen sizes and device performance. These transitions were improved by iterative testing and the usage of physics-based animation classes.
* **Security:**  
  One of the top priorities was making sure that user credentials were transmitted securely and that admin authentication was handled correctly. All communications had to use HTTPS.

## **5.4 Summary**

* Building a robust and responsive Android app that use real time data synchronization with engaging UI animations.
* Utilizing Firebase Firestore and FCM for seamless user interactions including push notifications.
* Implementing admin authentication and forgot password functionalities using a Java EE backend.
* Overcoming challenges related to asynchronous operations, UI performance, and security through accurate testing and adherence to best practices.

# **6. Results and Evaluation**

## **6.1 Functional Evaluation**

* **Feature Implementation:**  
  Floral Fete successfully implemented all main functionalities in the below:
  + User registration, login, and password recovery.
  + Product browsing, search, and Single product view with smooth UI animations.
  + A simple ordering process, including real-time order status updates via Firestore.
  + Push notifications using Firebase Cloud Messaging (FCM) triggered when Firestore updates.
  + Admin sign-in and User password reset were successfully handled by the Java EE backend.
* **Verification:**  
  Each feature was tested individually:
  + Unit tests and manual testing confirmed that the order status cycling (Processing → Packing → Shipped → Delivered → Processing) works as intended.
  + The forgot password flow including code generation, email delivery, and password update was verified using test accounts.

## **6.2 Performance Evaluation**

* **Real-Time Data Updates:**  
  Firestore provided almost real time updates. In tests, data updates were reflected on user devices within a few seconds.
* **Push Notification Latency:**  
  Notifications delivered via Firebase Cloud Messaging (FCM) were received on within 3–5 seconds after an order status change.
* **Load Testing:**  
  Major Loading testing were unable to do because lack of the test users.

## **6.3 Usability Testing**

* **User Interface and Experience:**
  + The Android app’s clean UI, modern layout and easy navigation we work as intended in user testing sessions.
  + Animated transitions, including the fling animation at launch where work as expected without an issue.
* **Feedback:**
  + Test Users appreciated the real-time updates and push notifications system.
  + Some testers suggested minor improvements in the search functionality and visual design of certain screens, which have been noted for future updates.

## **6.4 Limitations and Challenges**

* **Scalability:**  
  The system performs well under moderate load. But further testing is needed to ensure scalability for larger user bases.
* **Error Handling:**  
  Occasional latency in Firestore updates was observed. Might need to improve retry mechanisms in future updates.
* **Security:**  
  The password reset process currently uses in memory storage for verification codes. Need to add more robust mechanism in the future updates
* **Lack of Test Users:**

Lack of test users because of the testing level application. Need to undergo more testing using test uses.

## **6.5 Evaluation Metrics**

* **Quantitative Metrics:**
  + Average order status update latency: 3–5 seconds.
  + Push notification delivery time: average of 5 seconds.
  + Firestore query response times: <3 second under normal load.
* **Qualitative Metrics:**
  + User satisfaction scores from surveys averaged 4.2/5(using 10 users).
  + Admin panel testing feedback highlighted the ease of managing orders,users and products.

# **7. Future Work**

## **7.1 Enhancing Scalability and Performance**

* **Load Testing:**  
  Further scalability tests should be conducted under high user loads
* **Caching Mechanisms:**  
  Implementing more caching or offline data synchronization methods to reduce network problems and improve UX in low bandwidth situations.

## **7.2 Security Enhancements**

* **Secure Password Reset:**  
  Instead of temporary in memory storage for reset codes, consider integrating a more robust mechanism using a secure database.
* **Data Encryption:**  
  Encrypting sensitive user data would strengthen overall application security.
* **Two Factor Authentication (2FA):**  
  Implementing 2FA for admin and user logins.

## **7.3 Feature Extensions**

* **Advanced Payment Integration:**  
  Future updates could include with multiple payment gateways and support for digital wallets.
* **Enhanced Analytics:**  
  Add a detailed analytics to monitor user behavior, sales trends, and inventory levels which can improve business and over all app performance.
* **Personalized User Experience:**  
  Introducing recommendation systems for product suggestions based on past orders and user preferences.
* **Loyalty and Rewards Programs:**  
  Implement loyalty programs or discount offering programs to grab more user base for the business.

## **7.4 UI/UX Improvements**

* **Responsive Design Improvements:**  
  Optimize the user interface for more screen sizes and orientations.
* **Accessibility Features:**  
  Include accessibility improvements such as voice commands.
* **New Animations and Transitions:**  
  Add additional UI animations and transitions to further improve the user experience.

## **7.5 Future Admin Features**

* **Improved Admin Dashboard:**  
  Develop a admin panel has advanced reporting features, role based access control, and real time operational analytics.
* **Automated Inventory Management:**  
  Integrate machine learning algorithms to forecast inventory requirements to keep optimal stock levels and reducing waste.

## **7.6 Integration with Emerging Technologies**

* **IoT Integration:**  
  Consider integrating IoT devices (e.g:- for temperature and humidity monitoring in flower storage) to maintain product quality.

# **8. Conclusions**

Floral Fete was developed with the primary goal of providing a robust, user friendly mobile commerce application for purchasing fresh flowers. Also let administrators use efficient tools for managing orders, products, and users. The project successfully achieved its objectives.

Key achievements of the project include:

* **Seamless User Experience:**  
  The Android app features a clean, modern design with a Bottom Navigation View, Navigation Drawer View engaging UI animations, and real time updates.
* **Robust Data Management:**  
  Firestore was effectively used to store and manage data using various collections (Users, Products, Orders, Wishlist, etc.), ensuring that the data consistency is maintained.
* **Efficient Administrative Operations:**  
  Java EE backend was limited to handling admin sign in and forgot password functionalities. But it has provided a secure and straightforward mechanism for managing admin access and forgot password option for the users.
* **Push Notifications:**  
  The implementation of Firebase Cloud Messaging and a Cloud Function trigger keeping users informed of any updates in their order statuses. Which makes app more engaging

Throughout the development of Floral Fete, several challenges were encountered. Each challenge was solve by iterative testing and improvements, resulting in a system that not only meets the initial requirements but also has a solid foundation for future enhancements.

# **9. Reflection on Learning**

As my first Android project Floral Fete extended my knowledge to a new branch. This project gave me a rich setting for what Chris Argyris calls "double-loop learning," in which I not only picked up fresh technical skills but also re evaluated and changed the fundamental presumptions and techniques driving my work (Argyris, 1976). One of the most significant insights came from integrating real time data synchronization with Firebase Firestore. At first, I thought that managing dynamic user data would only require regular CRUD operations. But as the project developed, I realized that in order to satisfy the swiftness and responsiveness demands of modern users, real time synchronization was crucial.

When integrating push notifications using Firebase Cloud Messaging (FCM)it challenged my previous understanding of asynchronous communication. It also lead me to learn more about Firebase and cloud functions.

Working with a hybrid backend lead me to understand the benefits of using specialized cloud services like firebase for efficiency and scalability. My perspective on backend design has changed as a result of this experience. I now focus on flexible design and select the appropriate tool for each task.

In conclusion, Floral Fete wasn’t just about exploring the latest technology, it reshaped the way I think about designing systems. Working on this project teach me a more user first approach for application development. This project has not only improved my technical skills but also taught me the importance of flexibility, and continuous self improvement.

# **10. References and Bibliography:**

## **10.1 References**

Automated Android Notifications with Firebase Cloud Functions, Messaging, and Firestore[Video]. (Jul 17, 2021). YouTube. Retrieved April 27, 2024, from <https://youtu.be/If2eDphtutI?si=mezr5YT_mRldEu69>

Argyris, C. (1976). Single-loop and double-loop models in research on decision making. *Administrative Science Quarterly, 21*(3), 363–375.

## **10.2 Bibliography**

PhilJay. (n.d.). *MPAndroidChart: A powerful & easy to use chart library for Android*. GitHub. Retrieved February 24, 2025, from <https://github.com/PhilJay/MPAndroidChart>

Google. (n.d.). *Firebase Cloud Messaging documentation*. Firebase. Retrieved February 24, 2025, from <https://firebase.google.com/docs/cloud-messaging>

Google. (n.d.). *Cloud Firestore documentation*. Firebase. Retrieved February 06, 2025, from <https://firebase.google.com/docs/firestore>

Google. (n.d.). *Android developers*. Retrieved February 06, 2025, from <https://developer.android.com>

Square, Inc. (n.d.). OkHttp: HTTP & HTTP/2 client for Android and Java applications. Retrieved February 13, 2025, from <https://square.github.io/okhttp>

PayHere. (n.d.). *Android SDK*. Retrieved February 14, 2025, from <https://support.payhere.lk/api-&-mobile-sdk/android-sdk>

Abdul Rahman, N. (2020, May 15). *Getting started with Lottie animations in an Android app*. LottieFiles. Retrieved February 16, 2025, from <https://lottiefiles.com/blog/working-with-lottie-animations/getting-started-with-lottie-animations-in-android-app>

Bumptech. (n.d.). *Glide: An image loading and caching library for Android focused on smooth scrolling*. GitHub. Retrieved February 10, 2025, from <https://github.com/bumptech/glide>

Google. (n.d.). *Get started with the Google Maps Android SDK*. Retrieved February 16, 2025, from <https://developers.google.com/maps/documentation/android-sdk/start>