DaFoodies: A Seamless Solution for Campus Food Discovery and Ordering

By

Misha Fahamida
203-51-027

A report submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Information Technology & Management

Department of Information Technology & Management (ITM)
DAFFODIL INTERNATIONAL UNIVERSITY

Fall – 23

APPROVAL

This project titled on "**DaFoodies: A Seamless Solution for Campus Food Discovery and Ordering**", submitted by "**Misha Fahamida, 203-51-027**", to the Department of Information Technology & Management, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Information Technology & Management, and approval as to its style and contents

BOARD OF EXAMINNERS

Chairman

Nusrat Jahan
Head, and Assistant Professor
Department of Information Technology & Management
Faculty of Science and Information Technology
Daffodil International University

Internal

Dr. Bimal Chandra Das
Associate Dean, Faculty of Science and Information Technology
Professor, Dept. of Information Technology & Management
Daffodil International University

Internal

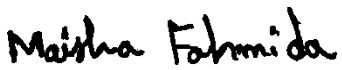
Nafees Imran,
Lecturer
Dept. of Information Technology & Management
Faculty of Science and Information Technology
Daffodil International University

External

Mohammad Abu Yousuf, PhD,
Professor
Institute of Information Technology
Jahangirnagar University

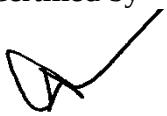
DECLARATION

It hereby declare that this project has been done by **Misha Fahamida** under the supervision of **Nafees Imran, Lecturer,** Department of Information Technology & Management (ITM), Daffodil International University. It also declares that neither this project nor any part of this has been submitted elsewhere for award of any degree.



Misha Fahamida
203-51-027
Batch: 203
Department of Information Technology & Management (ITM)
Faculty of Science & Information Technology
Daffodil International University

Certified by



Nafees Imran
Lecturer
Department of Information Technology & Management (ITM)
Faculty of Science & Information Technology
Daffodil International University

ACKNOWLEDGEMENT

First and foremost, I want to sincerely thank Allah for his divine favor, which allowed us to successfully finish the final year project.

I am appreciative to **Nafees Imran**,**Lecturer of ITM**, Daffodil International University. The completion of this project has been made possible by his unending patience, scholarly direction, ongoing encouragement, constant and energetic supervision, constructive criticism, insightful counsel, reviewing numerous subpar versions, and revising them at every level. I feel lucky to have him as my supervisor.

I would like to express our heartiest gratitude to **Nusrat Jahan, Head and Assistant Professor,** Department of Information Technology & Management, Daffodil International University, for his kind help in finishing my project and also to other faculty members and the staff of the Department of ITM, Daffodil International University.
We would like to thank our entire course mate in Daffodil International University, who took part in this discussion while completing the course work.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

TABLE OF CONTENTS

LIST OF FIGURE

ABSTRACT

The project report titled "DaFoodies" focuses on creating a mobile application aimed at improving the dining experience for university students and staff. The application allows users to discover, review, and order food from nearby shops with ease. It caters to four distinct roles: customers (students and teachers), shopkeepers, delivery personnel, and administrators. Each role is given specific functionalities, such as managing orders, reviews, and deliveries, as well as overseeing shop registrations.

Developed using Flutter and Firebase, the app leverages real-time data synchronization and secure user authentication. The project follows a structured process of requirement analysis, design, development, and testing. Key features include shop management, order tracking, and role-based access, with future enhancements such as map integration and payment options planned to expand its utility.

By addressing key needs within university campuses, DaFoodies aims to provide a streamlined and efficient solution for food discovery and order management. The report outlines the app's goals, strengths, and areas for improvement, offering a comprehensive guide to its implementation and future scope.

CHAPTER 01

INTRODUCTION

1.1 Project Overview

The primary objective of **DaFoodies** is to enhance the dining experience for university students and staff by offering a mobile application that simplifies the process of discovering, reviewing, and ordering food from nearby food shops. Developed specifically for university campuses, this application provides a convenient platform that addresses the unique needs of a campus environment, ensuring users can find quality food quickly and efficiently. The app supports four distinct user roles shopkeepers, customers (teachers and students), delivery personnel, and administrators each with tailored functionalities to ensure a seamless user experience.

For shopkeepers, the application offers an interface to manage shop details, menus, orders, and customer reviews, ensuring their offerings are always current and accessible. Customers can explore food shops using various filters, view detailed profiles and reviews, place orders for delivery or pickup, and track their orders in real-time. Delivery personnel benefit from a dedicated dashboard to manage their deliveries, update statuses, and navigate efficiently. Administrators oversee the verification and management of shop listings and customer reviews, maintaining the quality and reliability of the platform.

Developed using Flutter and Firebase for the database, **DaFoodies** integrates Firebase Authentication for secure user management. The project is structured in several phases: requirement analysis and planning, design and prototyping, development, testing and quality assurance, and deployment and maintenance.

Overall, **DaFoodies** aims to revolutionize the dining experience on university campuses by providing an all-in-one solution that connects the campus community, making the process of finding and ordering food more convenient and efficient. This documentation offers a comprehensive guide to the application's features, user interactions, and technical specifications, ensuring a clear understanding of its implementation and usage.

1.2 Project Purpose

The purpose of **DaFoodies** is to enhance the dining experience on university campuses by providing a convenient mobile platform for discovering, reviewing, and ordering food from nearby shops. The application aims to streamline food discovery and ordering processes for students and staff, improve operational efficiency for shopkeepers and delivery personnel, and ensure high-quality service through a robust review and verification system.

1.2.1 Background

In the dynamic environment of a university campus, students and staff often struggle to find quality food options quickly and conveniently. Traditional methods of discovering nearby food shops through word-of-mouth or random exploration are inefficient and time-consuming. Additionally, managing orders and deliveries can be chaotic without a streamlined system. Recognizing these challenges, I developed **DaFoodies**, a mobile application designed to address these specific needs by providing a centralized platform for discovering, reviewing, and ordering food from nearby food shops.

1.2.2 Benefits & Beneficiaries

Benefits

- **Convenience:** Users can easily find and order food from the best-reviewed shops nearby, saving time and effort.
- **Informed Choices:** The app offers detailed reviews and ratings, helping users make informed decisions about where to eat.
- **Efficient Order Management:** Shopkeepers can manage their orders and menus more efficiently, ensuring a smooth operation.
- **Delivery Coordination:** Delivery personnel have a dedicated interface to manage their tasks, ensuring timely and accurate deliveries.
- **Quality Control:** Administrators can verify shops and monitor reviews, maintaining a high standard of quality and reliability.

Beneficiaries

- **Students and Teachers:** The primary users who will benefit from quick access to a variety of food options and a streamlined ordering process.
- **Shopkeepers:** Local food shop owners who can reach a wider customer base and manage their operations more effectively.
- **Delivery Personnel:** Individuals responsible for delivering food, who will benefit from a well-organized system for managing deliveries.
- **University Administrators:** Who can ensure a controlled and high-quality food service environment on campus.

1.2.3 Goals

• **Enhance User Experience:** Provide an intuitive and user-friendly platform for discovering and ordering food.

• **Improve Operational Efficiency:** Help shopkeepers and delivery personnel manage their tasks more effectively.
• **Ensure Quality and Reliability:** Maintain high standards through a robust review and verification system.
• **Foster Community Connection:** Create a connected campus community where food shops and customers can interact seamlessly.

1.2.4 SWOT Analysis

Strengths

- **User-Friendly Interface:** Designed for ease of use, making it accessible to all users.
- **Targeted for Campus Environment:** Specifically tailored to meet the unique needs of university campuses.
- **Comprehensive Features:** Includes functionalities for all user roles (shopkeepers, customers, delivery personnel, and administrators).

Weaknesses

- **Limited Payment Options:** Currently only supports cash on delivery, which may limit convenience for some users.
- **Dependence on User Adoption:** Success depends on widespread adoption by the campus community.

Opportunities

- **Expansion to Other Campuses:** Potential to expand the app to other university campuses.
- **Feature Enhancements:** Opportunity to add more features like online payment methods, loyalty programs, and promotional offers.
- **Partnerships with Local Businesses:** Collaborations with local food shops and delivery services could enhance the app's offerings and reach.

Threats

- **Competition:** Other food delivery and discovery apps may compete for the same user base.
- **User Retention:** Ensuring long-term engagement and satisfaction among users can be challenging.
- **Technological Issues:** Potential technical difficulties or bugs that could affect user experience.

By addressing these factors, **DaFoodies** aims to provide a comprehensive solution that enhances the food discovery and ordering process on university campuses, benefiting all stakeholders involved.

1.3 Stakeholder

DaFoodies serves a diverse range of stakeholders within the university community, each benefiting from its unique functionalities and features:

1. **Students and Teachers (Customers):**
    - **Benefit:** Access to a variety of nearby food options, ability to read reviews and ratings, and convenience in placing orders for delivery or pickup.
    - **Interaction:** Actively engage with the app to explore food choices, rate experiences, and provide feedback.
2. **Shopkeepers:**
    - **Benefit:** Increased visibility and customer base through a digital platform, efficient management of menus, orders, and customer interactions.
    - **Interaction:** Manage shop profiles, update menus, handle orders, and respond to customer reviews to maintain service quality.
3. **Delivery Personnel:**
    - **Benefit:** Clear visibility of delivery assignments, efficient route planning, and timely delivery management.
    - **Interaction:** Utilize the app to update delivery statuses, navigate efficiently, and ensure prompt service.
4. **Administrators:**
    - **Benefit:** Oversight of shop verification, review management, and platform maintenance to uphold quality standards.
    - **Interaction:** Verify new shop registrations, monitor reviews, and ensure compliance with campus dining regulations.

These stakeholders collectively contribute to the success and usability of **DaFoodies**, ensuring it meets the diverse needs of the university community while enhancing the overall dining experience on campus.

1.4 Proposed System Model

DaFoodies is a mobile application developed using Flutter, designed to operate seamlessly across iOS and Android platforms. It utilizes Firebase as the backend infrastructure, offering scalable data storage, real-time synchronization, and robust authentication capabilities. The system model is structured to facilitate efficient food discovery, ordering, and management on university campuses, with future plans to integrate map functionalities for enhanced user experience.

Components of the System:

1. **Mobile Application (Flutter):**
    - **User Interfaces:** Developed using Flutter's widget-based framework to ensure a responsive and intuitive user experience.

- **Functionalities:** Tailored interfaces for customers, shopkeepers, delivery personnel, and administrators, facilitating distinct roles and operations within the app.

2. **Firebase Backend:**
   - **Authentication:** Firebase Authentication for secure user login and management.
   - **Realtime Database:** Firebase Realtime Database for storing and synchronizing app data in real-time, maintaining consistency across devices and users.
3. **Planned Integration:**
   - **Maps Integration:** Future integration with Google Maps API to enhance user experience with location-based services.
     - Displaying shop locations on a map interface for easy navigation and discovery.
     - Optimizing delivery routes for efficiency and timely order fulfillment.
     - Providing users with real-time location tracking of delivery personnel for enhanced transparency and convenience.
4. **Key Features and Interactions (Current and Planned):**
   - **Current Features:**
     - Customers can browse nearby food shops, view menus, place orders for delivery or pickup, and track order statuses.
     - Shopkeepers manage shop details, update menus, handle orders, and respond to customer reviews.
     - Delivery personnel access a dashboard to manage deliveries, update statuses, and navigate efficiently.
     - Administrators oversee shop verifications, manage listings, and monitor reviews to maintain platform integrity.
   - **Future Enhancements:**
     - **Maps Integration:** Enhance user experience with visual map displays, route optimizations, and real-time location updates.
     - **Advanced Analytics:** Implement Firebase Analytics for deeper insights into user behavior, performance metrics, and app optimization.
     - **Enhanced Notifications:** Expand Firebase Cloud Messaging capabilities for personalized notifications and alerts.

System Flow:

1. **User Registration and Authentication:**
   - Users securely register and log in using Firebase Authentication, with role-based access control for different user categories.
2. **Data Management:**
   - Firebase Realtime Database ensures seamless synchronization of user profiles, shop details, menus, orders, and reviews across all app instances.
3. **User Interactions:**
   - Customers engage in food discovery, order placement, and tracking within an intuitive app interface.

- Shopkeepers efficiently manage shop operations, respond to customer interactions, and maintain service quality.
- Delivery personnel handle delivery assignments, update statuses, and navigate routes for timely service.

4. **Scalability and Performance:**
   - Leveraging Firebase's scalable infrastructure and serverless capabilities ensures the application handles varying loads and performs optimally.
   - Continuous monitoring and optimization of backend processes maintain high availability and responsiveness.

By incorporating planned map integration and other future enhancements, **DaFoodies** aims to elevate the dining experience on university campuses, providing a comprehensive and user-friendly platform for all stakeholders involved.

1.5 Project Schedule

1.5.1 Gantt Chart

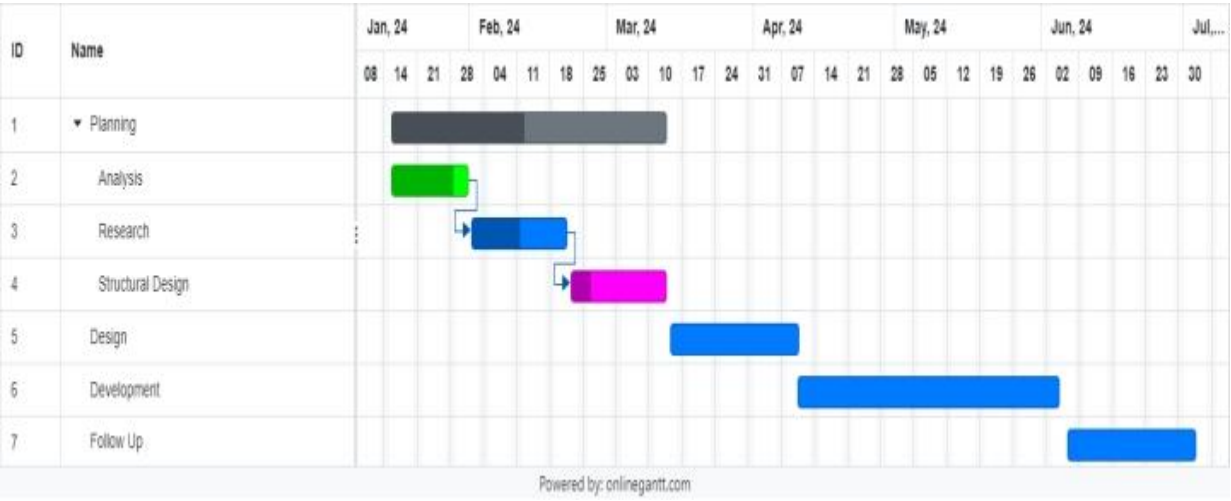


Fig 1.5.1.1 Gantt Chart

1.5.2 Release Plan

Release Plan for DaFoodies Android App

Overview

The release plan for DaFoodies Android app outlines the stages and processes involved in delivering a high-quality, stable application to users. This plan encompasses development, testing, deployment, and post-launch activities to ensure a smooth and successful release.

Goals

- Ensure thorough testing to identify and resolve issues before deployment.
- Gradual rollout to manage user feedback and ensure stability.
- Enhance user experience through iterative improvements and updates.

Release Stages

1. Development Phase

- **Objective:** Develop new features and functionalities, as well as address any bugs or issues identified in the previous version.
- **Tasks:**
    - Implement planned features such as map integration, enhanced user interfaces, and performance optimizations.
    - Conduct code reviews and ensure adherence to coding standards.
    - Integrate backend services (Firebase) for seamless data management and synchronization.

2. Testing Phase

- **Objective:** Conduct comprehensive testing to ensure app functionality, usability, and compatibility across a variety of devices and Android versions.
- **Tasks:**
    - **Unit Testing:** Test individual components and modules to verify functionality.
    - **Integration Testing:** Validate interactions between different parts of the app and backend services.

Compatibility Testing: Ensure compatibility with various Android OS versions, screen sizes, and device configurations

    - **User Acceptance Testing (UAT):** Solicit feedback from a select group of users to identify usability issues and gather suggestions for improvement.

3. Deployment Phase

- **Objective:** Deploy the app to Google Play Store in a staged manner to manage risks and ensure a smooth rollout.
- **Tasks:**
    - **Alpha Release:** Release to a small group of internal testers to gather initial feedback and identify major issues.
    - **Beta Release:** Expand the release to a larger group of external testers, including trusted users and stakeholders.
    - **Gradual Rollout:** Release incrementally to a percentage of users to monitor app performance and stability.
    - **Monitor Crash Reports:** Use crash reporting tools to monitor app stability and address any critical issues promptly.

4. Post-Launch Activities

- **Objective:** Monitor app performance, gather user feedback, and iterate based on insights to continuously improve the app.
- **Tasks:**
    - **Performance Monitoring:** Monitor key metrics such as app crashes, latency, and user engagement using Firebase Analytics.
    - **Feedback Collection:** Gather user feedback through app reviews, surveys, and customer support channels.
    - **Iterative Updates:** Plan and implement iterative updates based on user feedback and identified priorities.
    - **Promotion and Marketing:** Develop strategies to promote the app and increase user adoption, including social media campaigns and app store optimization (ASO).

5. Maintenance and Support

- **Objective:** Provide ongoing maintenance and support to address issues, release updates, and ensure continued app functionality.
- **Tasks:**
    - **Bug Fixes:** Address reported bugs and issues promptly through hotfixes or scheduled updates.
    - **Security Updates:** Implement security patches and updates to protect user data and maintain app security.
    - **Customer Support:** Provide responsive customer support to address user queries and concerns.
    - **Version Control:** Manage app versions and ensure backward compatibility with older versions where necessary

CHAPTER 02

2.1 Functional Requirements

Functional Requirements for DaFoodies Mobile App

1. **User Authentication and Profiles:**
    - Users (customers, shopkeepers, delivery personnel, administrators) should be able to register, login securely, and manage their profiles.
    - Differentiate access levels based on user roles to ensure appropriate permissions and functionalities.
2. **Shop Management:**
    - **Shopkeepers** should be able to:
        - Create and manage shop profiles including business hours, contact details, and location.
        - Update menus with current offerings, prices, and availability.
        - Receive and manage orders, update order statuses, and communicate with customers.
        - Respond to customer reviews and maintain service quality.
3. **Ordering and Delivery:**
    - **Customers** should be able to:
        - Browse nearby food shops based on location and preferences.
        - View detailed shop profiles including menus, ratings, and reviews.
        - Place orders for delivery or pickup and track order status in real-time.
        - Provide feedback through ratings and reviews.
    - **Delivery Personnel** should be able to:
        - Access a dedicated dashboard to view assigned deliveries.
        - Update delivery statuses, manage routes, and ensure timely deliveries.
        - Navigate efficiently using integrated maps and location services (future integration).
4. **Administrator Controls:**
    - **Administrators** should be able to:
        - Verify new shop registrations and manage shop listings.
        - Monitor and moderate customer reviews to maintain platform integrity.
        - Generate reports and analytics on app usage and performance metrics.

5. **User Interaction and Notifications**
    - Implement real-time notifications to inform users about order status updates, new reviews, and promotions.

- Facilitate seamless interactions between customers, shopkeepers, and delivery personnel through messaging or chat functionalities.

6. **Offline Support and Data Synchronization:**
   - Ensure basic app functionalities are available offline with automatic synchronization once connectivity is restored.
   - Maintain data integrity and consistency across devices using Firebase Realtime Database.
7. **Security and Privacy:**
   - Implement robust security measures to protect user data and transactions.
   - Comply with data privacy regulations and ensure secure storage and transmission of sensitive information.
8. **Performance and Scalability:**
   - Design the app architecture to handle varying loads and scale effectively as user base and usage grow.
   - Optimize app performance to deliver a responsive and smooth user experience on different devices and Android versions.
9. **Analytics and Reporting:**
   - Integrate Firebase Analytics to track user engagement, monitor app performance metrics, and gather insights for continuous improvement.
   - Generate reports on key metrics such as active users, order volumes, and user feedback trends.

These functional requirements outline the core features and capabilities that DaFoodies Android app will provide, aiming to deliver a comprehensive and user-friendly experience for all stakeholders involved in the campus dining ecosystem.

2.2 Data Requirements

Data Requirements for DaFoodies Android App

1. **User Data:**
   - **Attributes:** Name, email address, password (hashed), profile picture, role (customer, shopkeeper, delivery personnel, administrator), contact information.
   - **Usage:** Used for user authentication, profile management, and role-based access control within the app

2.3 Performance Requirements

Performance Requirements for DaFoodies Android App

Ensuring optimal performance is crucial for DaFoodies, a mobile application designed to streamline food discovery, ordering, and management on university campuses. These performance requirements outline the expectations for speed, latency, precision, accuracy, and capacity to deliver a seamless user experience.

2.3.1 Speed and Latency Requirements

- **Response Time:** The app should respond within 1 second on average to user interactions, ensuring a smooth and responsive experience.
- **Loading Times:** Screens and data should load within 2 seconds during app launch and major transitions.

2.3.2 Precision or Accuracy Requirements

- **Order Accuracy:** Orders must accurately reflect customer selections without discrepancies.
- **Location Accuracy:** Future map integration should provide precise location tracking and mapping functionalities.

2.3.3 Capacity Requirements

- **User Capacity:** Support a minimum of 10,000 concurrent users without significant performance degradation.
- **Data Handling:** Firebase Realtime Database should manage spikes in transactions during peak periods without data loss or latency.

2.4 Dependency Requirements

Dependencies play a critical role in ensuring the functionality, reliability, and security of **DaFoodies**, a mobile application designed for efficient food discovery and ordering on university campuses. These requirements outline essential dependencies and their impact on the app's performance and user experience.

2.4.1 Reliability Requirements

DaFoodies aims for high reliability to ensure consistent performance and minimal downtime for users.

- **Error Handling:** Implement comprehensive error handling mechanisms to gracefully manage unexpected errors and exceptions, ensuring uninterrupted app functionality.
- **Backup and Recovery:** Utilize Firebase's built-in backup and recovery features to safeguard data integrity and facilitate quick recovery in case of data loss or corruption.

2.4.2 Availability Requirements

Ensuring high availability is essential to minimize service disruptions and maximize user accessibility.

- **Uptime Guarantee:** Aim for a minimum uptime of 99.9% to ensure the app is available to users almost continuously.
- **Scalability:** Design the app architecture to scale horizontally and vertically to handle increased user demand during peak periods without degradation in performance.

2.4.3 Robustness Requirements

DaFoodies must exhibit robustness to handle various edge cases and adverse conditions effectively.

- **Input Validation:** Implement robust input validation to prevent invalid data inputs from causing app crashes or security vulnerabilities.
- **Graceful Degradation:** Design fallback mechanisms and alternative workflows to maintain core functionalities in case of backend service disruptions or network failures.

2.4.4 Safety-Critical Requirements

While DaFoodies primarily focuses on convenience and efficiency, ensuring safety-critical aspects is paramount.

- **Secure Transactions:** Implement encryption and secure protocols to protect user data during transactions, ensuring confidentiality and integrity.
- **Emergency Features:** Include emergency contact information or features for delivery personnel to handle unforeseen situations during deliveries, prioritizing user safety.

2.5 Maintainability and Supportability Requirements

2.5.1 Maintenance Requirement

DaFoodies prioritizes maintainability to facilitate efficient updates, bug fixes, and enhancements over time.

- **Modular Codebase:** Maintain a modular code architecture using Flutter to facilitate easier updates and code maintenance.
- **Version Control:** Utilize version control (e.g., Git) to track changes and manage codebase versions systematically.

- **Documentation:** Maintain comprehensive documentation for code structure, APIs, and deployment procedures to aid future maintenance and onboarding of new developers.

2.5.2 Supportability Requirements

Supportability focuses on providing effective support to users and resolving issues promptly.

- **Customer Support:** Offer responsive customer support through multiple channels (e.g., in-app chat, email) to address user queries and issues effectively.
- **Bug Reporting:** Implement a user-friendly mechanism for reporting bugs within the app, ensuring prompt resolution by the development team.
- **Feedback Integration:** Integrate user feedback mechanisms to gather insights for continuous improvement of app features and usability.

2.5.3 Adaptability Requirements

DaFoodies should be adaptable to evolving technological and user requirements.

- **Technology Updates:** Ensure compatibility with new Android versions and Flutter updates, adapting the app to leverage latest features and improvements.
- **Feature Flexibility:** Design the app architecture to easily incorporate new features and functionalities based on user feedback and market trends.

2.5.4 Scalability Requirements

Scalability ensures DaFoodies can handle growth in user base and transaction volume seamlessly.

- **Horizontal Scalability:** Design backend services (Firebase) to scale horizontally, accommodating increased user demand without compromising performance.
- **Load Balancing:** Implement load balancing techniques to distribute traffic evenly across servers and ensure consistent app performance during peak usage periods.
- **Performance Monitoring:** Continuously monitor app performance metrics (e.g., response times, server load) to proactively address scalability issues and optimize resource allocation.

2.6 Security Requirements

2.6.1 Access Requirements

Ensuring secure access to DaFoodies is crucial to protect user data and prevent unauthorized access.

- **Authentication:** Implement Firebase Authentication to securely manage user login and access control based on user roles (customer, shopkeeper, delivery personnel, administrator).
- **Authorization:** Enforce role-based access control (RBAC) to restrict access to sensitive functionalities and data based on user roles and permissions.
- **Session Management:** Implement secure session management to prevent session hijacking and unauthorized access to user accounts.

2.6.2 Integrity Requirements

Maintaining data integrity within DaFoodies is essential to ensure the accuracy and reliability of information.

- **Data Validation:** Implement input validation to prevent malicious data inputs that could compromise system integrity.
- **Encryption:** Use encryption mechanisms (e.g., TLS/SSL) to protect data during transmission between the app and backend servers, ensuring data integrity and confidentiality.
- **Checksums and Hashing:** Utilize checksums and hashing algorithms to verify data integrity and detect tampering or unauthorized modifications.

2.6.3 Privacy Requirements

Protecting user privacy is paramount to build trust and comply with data protection regulations.

- **Data Minimization:** Collect and store only necessary user information required for app functionalities, adhering to the principle of data minimization.
- **Privacy Policy:** Provide a clear and accessible privacy policy outlining how user data is collected, used, and protected within DaFoodies.
- **User Consent:** Obtain explicit consent from users before collecting any personal information, ensuring transparency and compliance with privacy regulations.

2.7 Usability and Human-Interaction Requirements

2.7.1 Ease of Use Requirements

- **Intuitive Navigation:** Design an intuitive user interface (UI) with clear navigation paths and consistent layout across screens.
- **Simplified Ordering:** Streamline the order placement process with minimal steps and intuitive controls for selecting items, quantities, and preferences.
- **Feedback Mechanism:** Implement visual and auditory feedback to confirm user actions (e.g., order confirmation) and provide status updates.

2.7.2 Personalization and Internationalization Requirements

- **Personalized Recommendations:** Offer personalized recommendations based on user preferences and order history to enhance user experience.
- **Language Support:** Support multiple languages within the app to cater to international users and ensure content localization.
- **Cultural Sensitivity:** Design UI elements and content that respect cultural norms and preferences of users from different backgrounds.

2.7.3 Understandability and Politeness Requirements

- **Clear Instructions:** Provide clear and concise instructions throughout the app to guide users in performing actions (e.g., placing orders, updating profiles).
- **Polite Notifications:** Use polite and respectful language in notifications and user interactions to maintain a positive user experience.
- **Error Handling:** Communicate errors and issues to users in a non-technical and understandable manner, offering guidance on how to resolve or mitigate them.

2.7.4 Accessibility Requirements

- **Accessibility Features:** Implement accessibility features such as screen reader compatibility, voice control support, and text resizing options.
- **Color Contrast:** Ensure sufficient color contrast and use of accessible color schemes to aid users with visual impairments.
- **Keyboard Navigation:** Support keyboard navigation and shortcuts to facilitate seamless interaction for users with mobility impairments.

2.7.5 User Documentation Requirements

Providing comprehensive and accessible user documentation is essential for user empowerment and support.

- **Online Help:** Include in-app help sections with searchable FAQs, tutorials, and troubleshooting guides to assist users in navigating and using DaFoodies effectively.
- **Contextual Help:** Offer contextual help tips and tooltips throughout the app to provide immediate assistance on specific features or actions.
- **User Guides:** Provide downloadable user guides or manuals covering app functionalities, settings, and best practices for users' reference.

2.7.6 Training Requirements

- **Onboarding Process:** Design a user-friendly onboarding process with interactive tutorials and walkthroughs to introduce new users to app features and functionalities.
- **Training Materials:** Develop training materials and resources (e.g., video tutorials, webinars) for shopkeepers and delivery personnel on using DaFoodies for managing orders and deliveries effectively.

- **Continuous Learning:** Offer ongoing training and updates to keep users informed about new features, app updates, and best practices.

2.8 Look and Feel Requirements

2.8.1 Appearance Requirements

- **Clean and Modern Design:** Use a clean and modern design language throughout the app interface to provide a visually pleasing experience.
- **Consistent UI Elements:** Maintain consistency in UI elements such as colors, typography, and iconography across all screens and interactions.
- **Visual Hierarchy:** Implement a clear visual hierarchy to prioritize important information and actions, guiding users effectively through the app.

2.8.2 Style Requirements

- **Brand Colors:** Use a consistent color palette that aligns with DaFoodies' branding guidelines, ensuring colors evoke positive emotions and enhance usability.
- **Typography:** Choose readable fonts and maintain consistent typography styles (e.g., font size, weight) to enhance readability and accessibility.
- **Iconography:** Utilize intuitive and meaningful icons that support navigation and reinforce actions, maintaining clarity and usability.

2.9 Operational and Environmental Requirements

2.9.1 Expected Physical Environment

DaFoodies is designed to operate in various physical environments (Android, IOS) typically found on university campuses.

- **Mobile Devices:** Support Android smartphones and tablets with varying screen sizes and resolutions commonly used by students, teachers, and staff.
- **Internet Connectivity:** Require stable internet connectivity (Wi-Fi or mobile data) for real-time updates, order processing, and communication between users and the backend server.

2.9.2 Requirements for Interfacing with Adjacent Systems

DaFoodies interacts with adjacent systems to ensure seamless functionality and data exchange.

- **Firebase Integration:** Interface securely with Firebase backend services (Realtime Database, Authentication) for user authentication, data storage, and real-time updates.

2.9.3 Projectization Requirements

DaFoodies adheres to project management principles to ensure organized development and deployment processes.

- **Task Management:** Utilize project management tools (e.g., Jira, Trello) for task assignment, tracking progress, and managing software development lifecycle (SDLC) phases.
- **Agile Methodology:** Follow agile practices such as iterative development, regular sprint planning, and continuous integration to adapt to changing requirements and deliver incremental updates.

2.9.4 Release Requirements

Efficient release management is essential for deploying updates and new features to users.

- **Staged Rollouts:** Conduct staged rollouts of app updates to gradually release new features and updates to users while monitoring performance and user feedback.
- **Version Control:** Manage app versions and track changes using version control systems (e.g., Git) to ensure codebase integrity and facilitate rollback if necessary.

2.10 Legal Requirements

These legal and standards requirements are crucial for DaFoodies to operate legally and responsibly, ensuring user trust, data protection, and compliance with local regulations and industry standards.

2.10.1 Compliance Requirements

DaFoodies must comply with legal regulations and standards to operate lawfully in Bangladesh.

- **Data Protection:** Adhere to Bangladesh's Data Protection Act to ensure proper collection, storage, and handling of user data, including obtaining explicit consent for data processing activities.
- **Consumer Protection:** Comply with consumer protection laws, ensuring transparency in pricing, terms of service, and dispute resolution mechanisms for users.
- **Intellectual Property:** Respect intellectual property rights, including trademarks and copyrights, when using third-party logos, images, and content within the app.

CHAPTER 03

System Analysis

3.1 Use Case Diagram

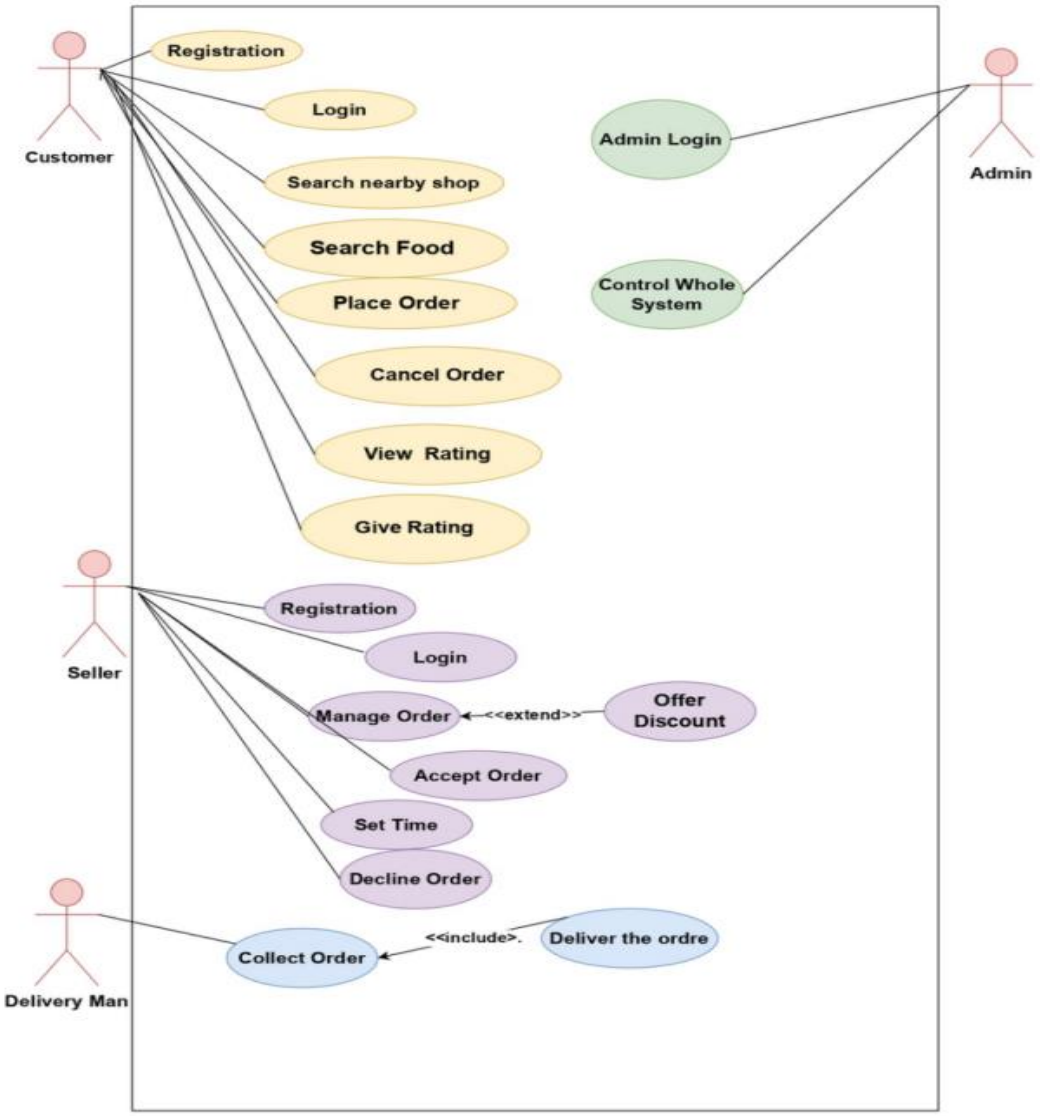


Fig 3.1.1 Use Case Diagram

3.2 Use Case Description

Use Case: Managing Shops and Orders

Actors:

- **Customer:** Teacher/ Student
- **Shopkeeper**
- **Delivery Man**
- **Admin**

Preconditions:

- All actors have installed the DaFoodies app and logged into their respective accounts.

Use Case Flow:

1. **Customer Actions:**
    - **Login and View Shops:**
        - Customer logs into the DaFoodies app.
        - They browse nearby shops or search for specific ones.
    - **Select Shop and View Details:**
        - Customer selects a shop from the list.
        - They view shop details including menu items, ratings, and delivery options if available.
    - **Browse Foods and Place Order:**
        - Within the selected shop, customer browses food items.
        - They add desired items to the cart and proceed to checkout.
        - Customer selects delivery if offered by the shop.
    - **View Order History:**
        - Customer can view past orders in the Order History section.
    - **Cancel Order:**
        - If the shop hasn't accepted the order, customer can cancel it.
    - **Rate and Review:**
        - After receiving the order, customer can rate and review both the food items and the shop.
2. **Shopkeeper Actions:**
    - **Manage Shop:**
        - Shopkeeper logs into DaFoodies Shopkeeper Dashboard.
        - They can add, update, or delete shop information and food items.
    - **Accept or Decline Orders:**
        - Shopkeeper receives order notifications and can accept or decline orders based on availability.
    - **Set Preparation Time:**
        - They specify preparation time for orders to manage customer expectations.
3. **Delivery Man Actions:**

- **Accept Delivery Request:**
  - Delivery man receives delivery requests from shops.
- **Deliver Order:**
  - They pick up orders from shops and deliver them to customers.

4. **Admin Actions:**
   - **Verify Shop:**
     - Admin verifies new shop registrations to ensure legitimacy.
   - **Manage Shops:**
     - They can add or remove shops based on policy compliance and user feedback.
   - **Monitor App Operations:**
     - Admin oversees all app functions to ensure smooth operation and user satisfaction.

Postconditions:

- Customer successfully places and receives orders.
- Shopkeeper manages shop efficiently and fulfills orders.
- Delivery man completes deliveries promptly.
- Admin verifies shops and maintains app integrity.

Exceptions:

- **Order Rejection:** If shopkeeper declines an order, customer is notified.
- **Technical Issues:** Users may need to restart the app or contact support for assistance.

Use Case End

This structured use case description outlines how DaFoodies facilitates interactions among customers, shopkeepers, delivery personnel, and admin roles, ensuring efficient management of shops, orders, and app functionalities within a university campus setting.

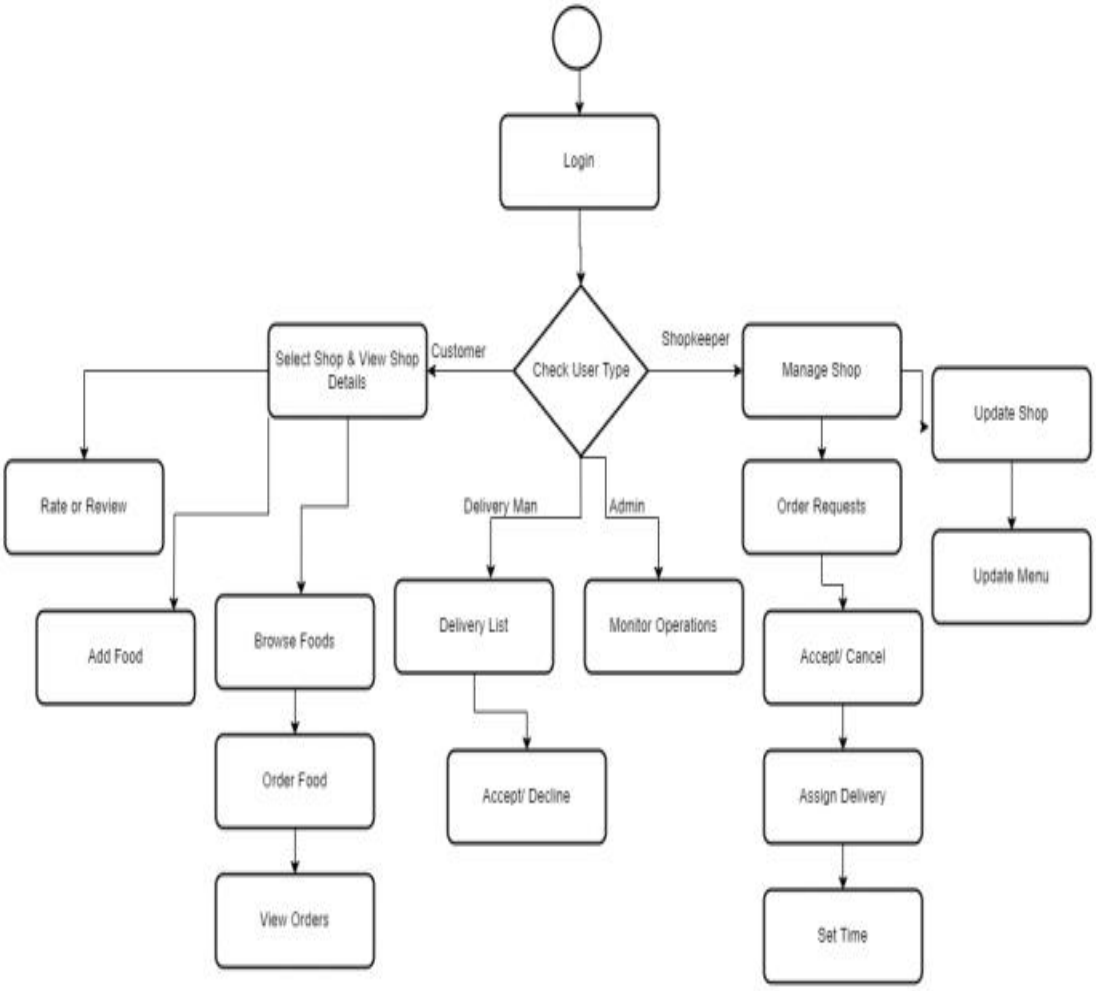


Fig 3.3.1 Activity Diagram

3.4 System Sequence Diagram

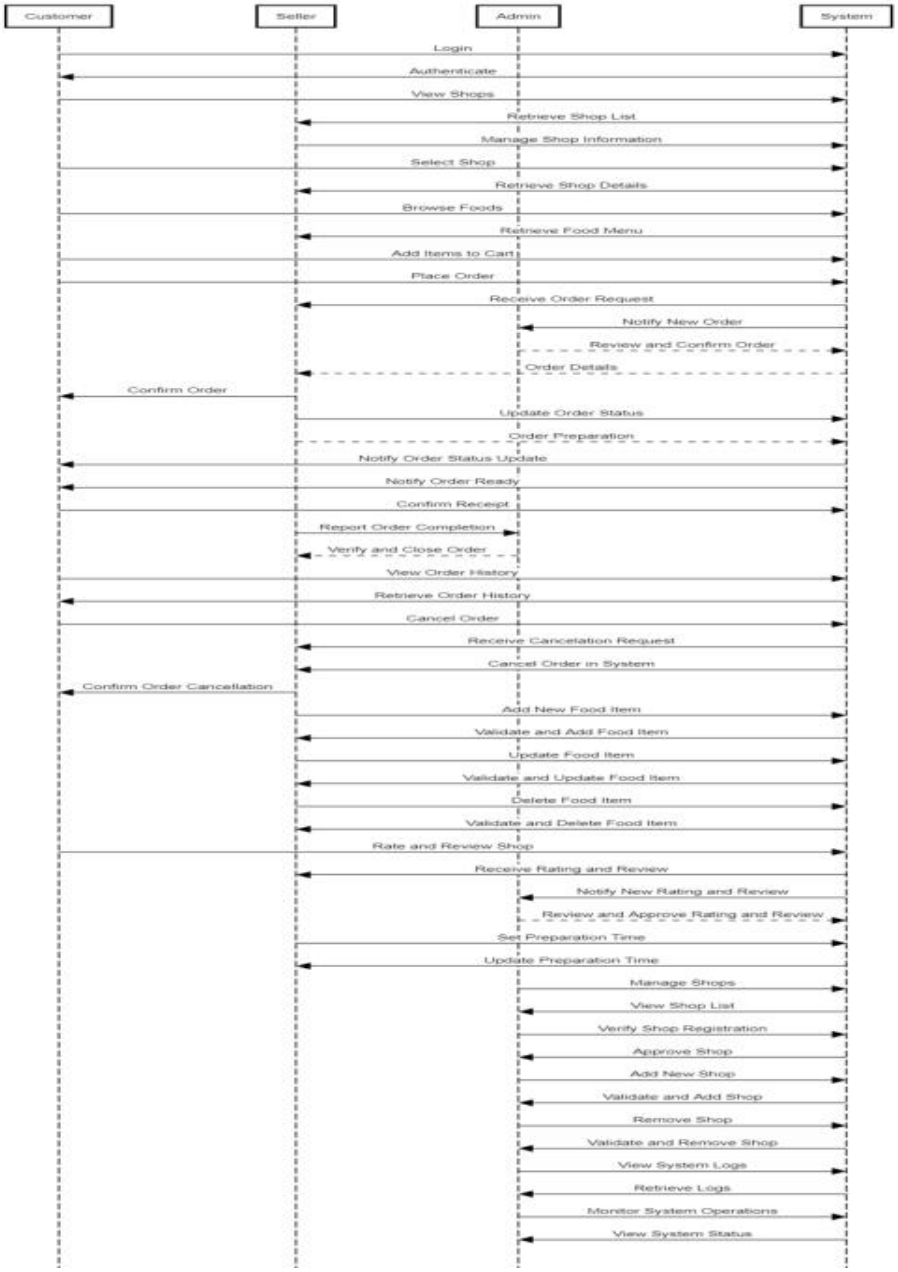


Fig 3.4.1 System Sequence Diagram

3.5 ER Diagram

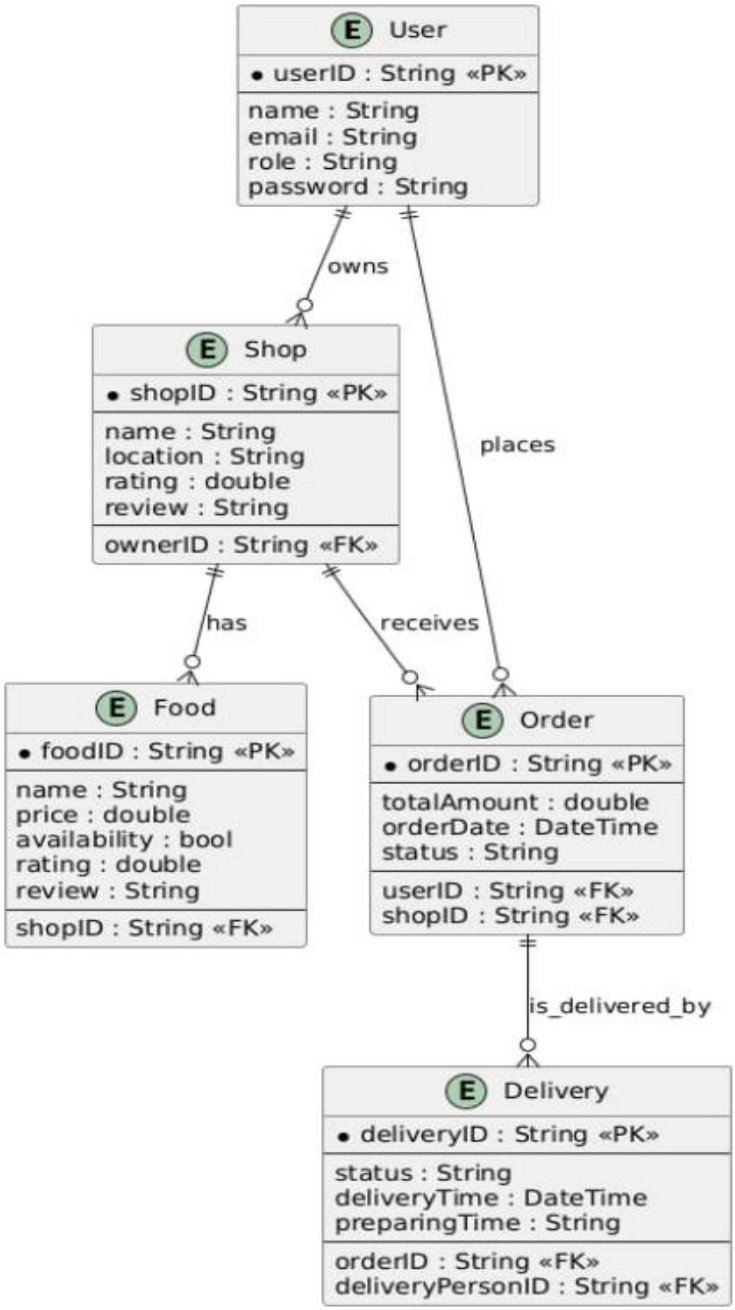


Fig 3.5.1 Entity Relationship Diagram

CHAPTER 04

System Design Specification

4.1 Class Responsibilities Collaboration (CRC) Cards

Class: User Model

Responsibilities:

- Manage user profile information (name, email, address, phone, image)
- Store user's favorite shops and foods

Collaborations:

- Collaborates with Restaurant to store favorite shop
- Collaborates with Food to store favorite food
- Places orders via Order Model

Class: Restaurant

Responsibilities:

- Manage restaurant details (name, email, address, ownerId, image, location)
- Manage restaurant's menu (list of Food items)
- Handle reviews and ratings
- Track delivery availability and list of delivery men

Collaborations:

- Contains multiple Food items
- Processes orders via Order Model
- Works with Deliveryman for order delivery
- Managed by Admin

Class: Food

Responsibilities:

- Manage food item details (name, image, delivery availability, ratings, reviews)
- Track order and visit statistics

Collaborations:

- Part of Restaurant menu
- Included in Order Model

Class: Order Model

Responsibilities:

- Manage order details (ordered, userId, shopId, status, address, distance, prepare Time)
- Track order status (accepted, on the way, canceled, delivered)
- Include food item details

Collaborations:

- Linked to User Model for customer information
- Linked to Restaurant for shop information
- Includes Food item
- Delivered by Deliveryman
- Processed by Shopkeeper

Class: Deliveryman

Responsibilities:

- Manage delivery requests
- Track and update delivery status

Collaborations:

- Delivers orders from Restaurant to User Model
- Interacts with Order Model for delivery details

Class: Admin

Responsibilities:

- Verify and manage shops
- Add and remove shops
- View and manage all other functions

Collaborations:

- Manages Restaurant
- Collaborates with Shopkeeper for shop operations

Class: Shopkeeper

Responsibilities:

- Manage shop operations (add, update, delete Food items)
- Accept or decline orders
- Set order preparation time

Collaborations:

- Manages Restaurant details and menu
- Processes orders via Order Model
- Interacts with Admin for shop verification

4.2 Sequence Diagram

The sequence diagram for the DaFoodies app illustrates the interactions between customers, shopkeepers, delivery men, and the admin with the mobile application. Customers can log in, view and search for shops and food items, place orders, manage order history, cancel orders, and add ratings and reviews. Shopkeepers manage shop details and food items, handle orders by accepting or declining them, and set preparation times. Delivery men receive delivery requests, accept or decline them, and update delivery statuses. Admins verify and manage shops, add or remove shops, and access an admin dashboard to view all app functions. This diagram provides a clear overview of the flow of actions and responses within the DaFoodies app, highlighting the key functionalities and user interactions.

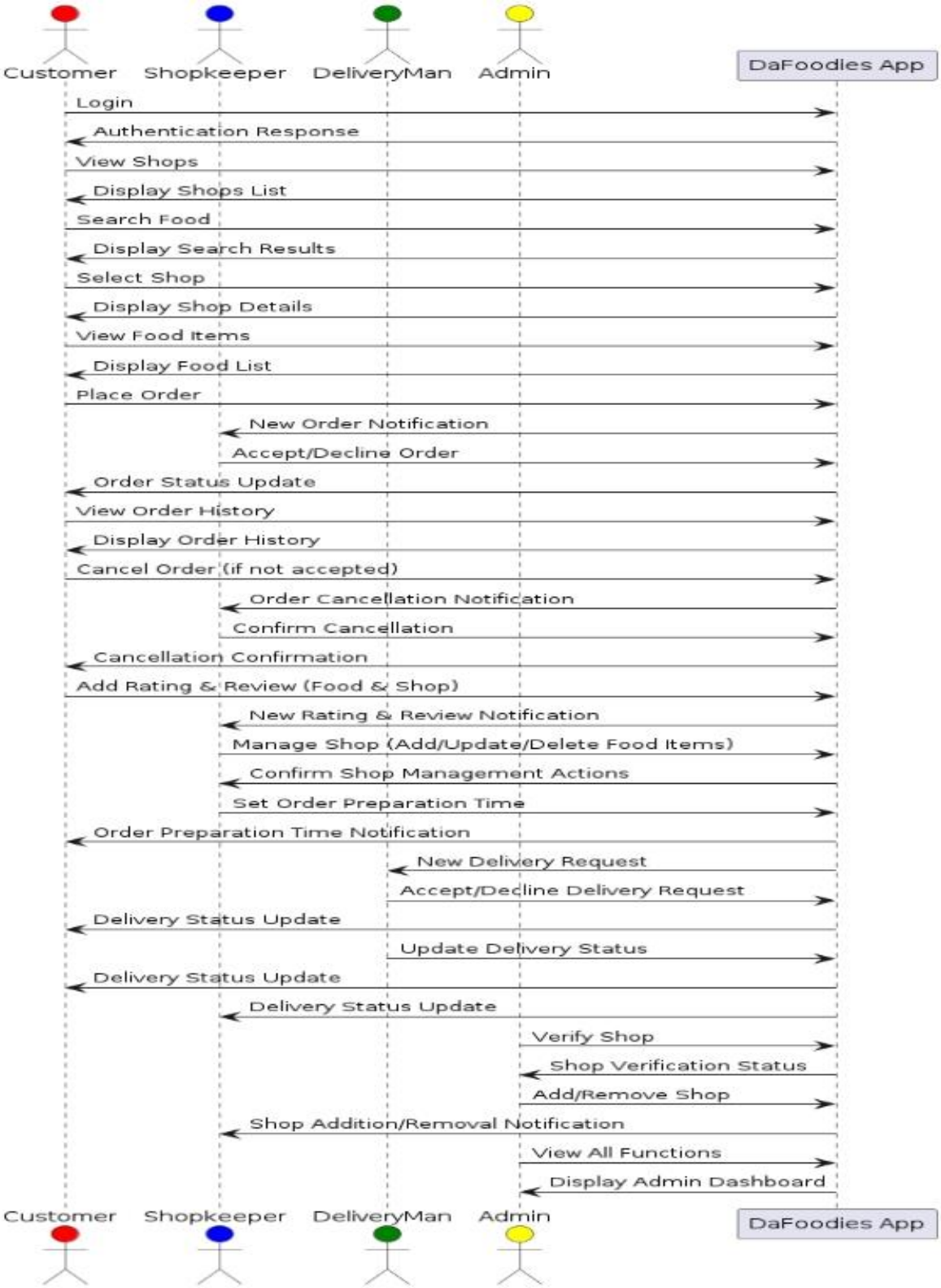


Fig 4.2.1 Sequence Diagram

4.3 Class Diagram

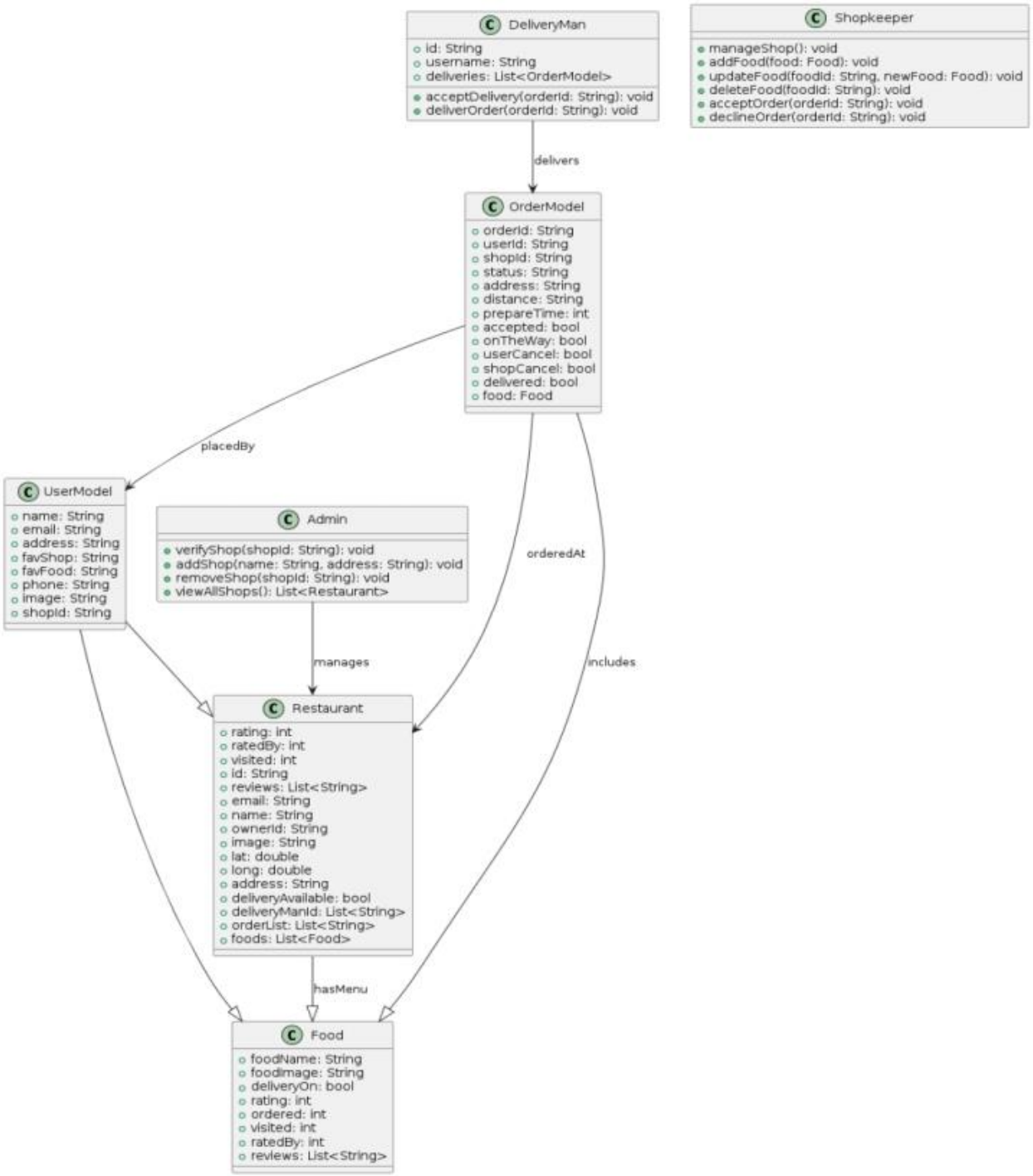


Fig 4.3.1 Class Diagram

4.4 Database Design

Given that the DaFoodies app uses Firebase, the database design would be structured with collections and documents, rather than traditional relational database tables. Here's how the Firebase collections and documents can be structured based on your provided models and requirements:

Firebase Database Structure:

1. **Users Collection**
    - Document ID: auto-generated user ID
        - name: String
        - email: String
        - address: String
        - favShop: String (reference to Shops collection)
        - favFood: String (reference to Foods collection)
        - phone: String
        - image: String
        - shopId: String (reference to Shops collection, nullable for shopkeepers)
2. **Shops Collection**
    - Document ID: auto-generated shop ID
        - name: String
        - email: String
        - address: String
        - ownerId: String (reference to User's collection)
        - image: String
        - lat: Double
        - long: Double
        - rating: Integer
        - ratedBy: Integer
        - visited: Integer
        - delivery Available: Boolean
        - reviews: [String] (array of review IDs)
        - foods: [String] (array of food IDs)
        - deliveryman: [String] (array of user IDs)
        - order List: [String] (array of order IDs)
3. **Shopkeepers Collection**
    - Document ID: auto-generated shopkeeper ID
        - user basic info
        - shopId: String (reference to Shops collection)
4. **Deliveryman Collection**
    - Document ID: auto-generated delivery man ID
        - user basic info
        - shopId: String (reference to Shops collection)

5. **Orders Collection**
    - Document ID: auto-generated order ID
        - userId: String (reference to Users collection)
        - shopId: String (reference to Shops collection)
        - status: String
        - address: String
        - distance: String
        - prepare Time: Integer
        - accepted: Boolean
        - onTheWay: Boolean
        - userCancel: Boolean
        - shopCancel: Boolean
        - delivered: Boolean
        - foodId: String (reference to Foods collection)

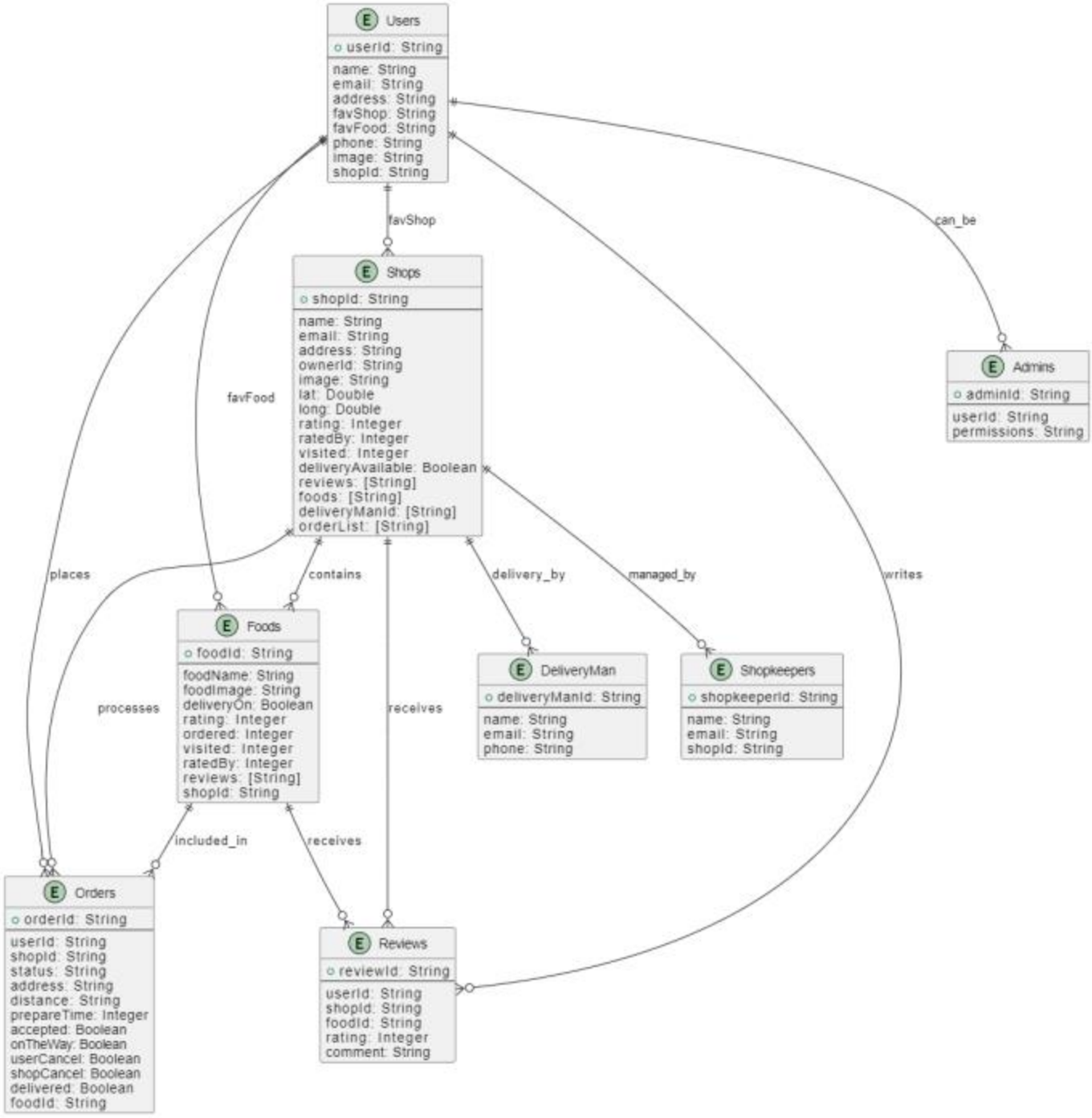


Fig 4.4.1 Database Design

Finance:

1. App Pricing:

The market value of your app will largely depend on its popularity and user base growth. As the app gains traction and attracts more users, it has the potential to evolve into a startup. The increase in user base and engagement can drive up the app's market value, leading to potential investment opportunities or higher revenue.

2. Development Costs:
Firebase Costs:
Free Plan Limits:
1 GiB of stored data
50,000 document reads/day
20,000 document writes/day
20,000 document deletes/day
10 GiB of outbound data transfer/month

Pay-as-You-Go Plan (Blaze Plan): As your user base grows, you'll transition to Firebase's Blaze Plan, which charges based on usage. The costs will depend on the volume of data stored, reads, writes, deletes, and outbound data transfer. You can estimate future costs using Firebase's Blaze Plan Calculator: Firebase Pricing Calculator.

3. App Revenue Streams:
Delivery Fees:
Deliverymen earn 20 Taka per successful delivery.
Platform Service Charges:
The platform charges 10 Taka per order.
High-Volume Shop Charges:
Shops processing 100+ orders daily will be charged a 70 Taka daily fee.

4. Food Pricing and Admin Monitoring:
Food Prices:
Flexible and set by shopkeepers.
Admin Oversight:
Admins can monitor and intervene to ensure price fairness and address any unusual pricing or reviews.

5. Profitability Timeline:

Initial Costs:

Since the app was self-developed, there are no initial development costs.

Firebase Costs:

Currently operating on the free plan, future charges will be incurred as usage exceeds the free limits. These costs will be based on the pay-as-you-go Blaze Plan.

Revenue Growth:

As user numbers and order volumes increase, platform service charges and high-volume shop charges will boost revenue. This increase in revenue will help cover the costs associated with the Blaze Plan and lead to profitability.

Initial Stage:
Initially, you won't have significant costs apart from Firebase's free plan limitations.

Production Stage:
As the user base and order volumes increase, you'll transition to the Blaze Plan for Firebase, incurring costs based on usage. However, the revenue from delivery fees, platform service charges, and high-volume shop charges will increase, helping to offset these costs and drive profitability.

4.5 Development Tools & Technology

4.5.1 User Interface Technology

Flutter:

- **Framework:** Google's UI toolkit for building native apps across platforms.
- **Language:** Dart, known for its simplicity and efficiency.
- **Cross-Platform:** Develop once, deploy on iOS and Android.
- **Features:** Fast development with hot reload, beautiful UIs with customizable widgets, native performance, and access to device features.
- **DaFoodies Application:**
    - **Mobile App:** Entirely developed in Flutter.
    - **UI Design:** Uses Flutter widgets for user profiles, shop listings, menus, orders, and reviews.
    - **Integration:** Syncs with Firebase for real-time data and authentication.
    - **User Interaction:** Seamless browsing, ordering, and profile management.

4.5.2 Implementation Tools and Platforms

Implementing the DaFoodies mobile app involves using several tools and platforms, primarily focused on Flutter development for Android devices. Here's how each tool and platform contribute to the implementation.

4.5.2.1 Visual Studio Code

- **Role:** Integrated Development Environment (IDE) for coding Flutter apps.
- **Features:**
    - **Flutter Extensions:** Provides essential Flutter extensions for syntax highlighting, code completion, and debugging.
    - **Dart Support:** Built-in support for Dart programming language.
    - **Version Control:** Integrates with Git for version control and collaboration.
    - **Extensions Marketplace:** Access to a wide range of extensions for enhanced productivity.

4.5.2.2 Android Studio

- **Role:** Official IDE for Android app development and Flutter projects.
- **Features:**
    - **Flutter Plugin:** Offers a dedicated Flutter plugin with tools for Flutter development.
    - **Emulator Integration:** Built-in Android Emulator for testing Flutter apps.
    - **Code and UI Design:** Supports both coding and visual design of Flutter UI components.
    - **Performance Profiling:** Tools for debugging and optimizing app performance on Android.

4.5.2.3 Android Emulator

- **Role:** Virtual device for testing Android applications.
- **Features:**
    - **Device Simulation:** Provides a range of Android device configurations for testing.
    - **App Compatibility:** Ensures apps work across different Android versions and screen sizes.
    - **Integration with IDEs:** Seamless integration with Android Studio and Visual Studio Code for easy deployment and testing.
    - **Performance Testing:** Allows testing app responsiveness and behavior under different conditions.

CHAPTER 05

Testing Features

5.1.1 Features to be Tested

1. **User Authentication:**
    - Verify login and registration processes for users, shopkeepers, and delivery personnel.
    - Test authentication flows using different credentials (valid, invalid).
2. **Shop and Food Listings:**
    - Ensure accurate display of shops and food items based on user location.
    - Test sorting and filtering options (by rating, distance, cuisine).
3. **Order Placement and Management:**
    - Verify order placement, status updates (accepted, on the way, delivered), and cancellation.
    - Test order tracking functionality for users and delivery personnel.
4. **Review and Rating System:**
    - Validate the ability to add, view, and update reviews for shops and food items.
    - Test rating submission and display consistency.
5. **User Profile Management:**
    - Ensure users can update their profiles (name, address, phone) and preferences (favorite shops, foods).
    - Test profile synchronization with backend data.
6. **Admin and Shopkeeper Functions:**
    - Verify admin functions like shop verification and management (add, remove shops).
    - Test shopkeeper operations including menu management and order handling.
7. **Performance and Scalability:**
    - Test app performance under varying loads (e.g., simultaneous orders, large datasets).
    - Validate responsiveness and latency in real-time interactions.
8. **Integration with Firebase Backend:**
    - Verify data synchronization between the app and Firebase backend.
    - Test offline capabilities and data integrity.

5.1.2 Features not to be Tested

1. **Physical Device Compatibility:**
    - Testing on specific physical devices (e.g., specific models or hardware configurations) may not be feasible during initial stages unless specified.
2. **Third-Party Services:**
    - External services not directly managed by DaFoodies (e.g., Google Maps integration) should be assumed to function correctly unless integration issues arise.
3. **Platform-Specific Behaviors:**
    - Native behaviors specific to iOS (not supported in current scope) won't be tested.
4. **Legal Compliance:**
    - Compliance with local laws and regulations (e.g., data privacy) should be verified but may not be fully tested within the app's scope.
5. **Edge Cases Beyond Scope:**
    - Extremely rare edge cases or unrealistic scenarios not impacting core functionality may be excluded from testing.

5.2 Testing Strategies

Test Approach

The test approach for DaFoodies involves a combination of manual and automated testing to ensure comprehensive coverage of all critical functionalities and performance metrics. The testing will be conducted in multiple phases, starting from unit testing and progressing through integration, system, and user acceptance testing.

1. **Unit Testing:**
    - Focus on individual components, ensuring each function, method, and module works as expected.
    - Use frameworks like Flutter's built-in testing tools and Dart's test package.
2. **Integration Testing:**
    - Test the interaction between integrated components, such as user authentication with Firebase, order management, and data synchronization.
    - Ensure smooth communication between frontend and backend components.
3. **System Testing:**
    - Perform end-to-end testing of the entire application to verify the complete workflow, from user login to order placement and delivery tracking.
    - Use real device testing and emulators to cover different Android versions and screen sizes.
4. **User Acceptance Testing (UAT):**
    - Involve actual users (teachers, students, shopkeepers, and delivery personnel) to validate the app’s functionality and usability.

    - Collect feedback to identify any usability issues or additional requirements.
5. **Performance Testing:**
    - Test app performance under varying loads to ensure scalability.
    - Measure response times, throughput, and resource utilization.
6. **Regression Testing:**
    - Re-test previously tested functionalities after any code changes to ensure no new issues are introduced.
    - Use automated regression test suites to save time and ensure consistency.

5.2.1 Pass/Fail Criteria

- **Pass Criteria:**
    - All test cases must pass without any critical or major defects.
    - The application must meet all specified requirements and performance benchmarks.
    - User feedback from UAT must be positive, with no significant usability issues.
- **Fail Criteria:**
    - Any critical or major defect that impacts core functionality.
    - Failure to meet specified requirements or performance benchmarks.
    - Negative feedback from users indicating significant usability issues.

5.2.2 Suspension and Resumption

- **Suspension Criteria:**
    - Testing will be suspended if a critical defect is found that blocks further testing.
    - Insufficient resources (e.g., test environment, data) to continue testing.
- **Resumption Criteria:**
    - Testing will resume once the critical defect is resolved and verified.
    - Required resources are available, and the test environment is restored.

5.2.3 Testing Schedule

- **Unit Testing:** Weeks 1-2
- **Integration Testing:** Weeks 3-4
- **System Testing:** Weeks 5-6
- **User Acceptance Testing (UAT):** Weeks 7-8
- **Performance Testing:** Weeks 9-10
- **Regression Testing:** Continuous throughout the testing phases and after any code changes.

5.2.4 Traceability Matrix

The traceability matrix ensures all requirements are covered by test cases, providing a clear mapping between requirements and their corresponding tests.

Requirement ID	Requirement Description	Test Case ID(s)
R1	User login functionality	TC1, TC2
R2	User registration	TC3, TC4
R3	Shop listing and search	TC5, TC6, TC7
R4	Order placement and tracking	TC8, TC9, TC10
R5	Review and rating system	TC11, TC12
R6	Admin functionalities	TC13, TC14, TC15
R7	Delivery management	TC16, TC17
R8	Performance benchmarks	TC18, TC19, TC20

5.3 Testing Environment

The testing environment for DaFoodies is set up to simulate real-world conditions as closely as possible, ensuring that the application performs well under various scenarios and conditions. The environment includes hardware, software, network configurations, and data setups necessary for comprehensive testing.

Hardware

- **Devices:**
  - A variety of Android devices with different screen sizes and resolutions.
  - Emulators to simulate different Android versions and device configurations.
- **Servers:**
  - Firebase backend servers for real-time database interactions and authentication.

Software

- **Operating Systems:**
  - Android OS versions 8.0 (Oreo) to the latest version (ensure compatibility across a wide range of devices).
- **Development Tools:**
  - **Visual Studio Code:** For code editing and running unit tests.
  - **Android Studio:** For emulating Android devices, debugging, and running integration and system tests.
  - **Android Emulator:** To test the app on various virtual devices.
- **Testing Frameworks:**
  - **Flutter's built-in testing tools:** For unit and widget testing.

    - **Dart's test package:** For unit testing and assertions.
    - **Firebase Test Lab:** For automated testing on a wide range of real devices hosted by Google.
- **Continuous Integration Tools:**
    - **GitHub Actions:** For automated build and test processes.

Network

- **Wi-Fi and Mobile Data:**
    - Test the application under different network conditions (Wi-Fi, 4G, and 5G).
    - Simulate network latency and intermittent connectivity to test the app's resilience.
- **VPN:**
    - Use VPN to simulate different geographic locations and ensure the app works globally.

Data

- **Test Data:**
    - A set of pre-populated data in the Firebase database, including user accounts, shops, food items, orders, reviews, and ratings.
    - Mock data for testing edge cases and error conditions.
- **Data Privacy:**
    - Ensure that test data complies with data privacy regulations and does not include any real user data.

Configuration Management

- **Version Control:**
    - Use Git for managing source code and configuration files.
- **Environment Variables:**
    - Configure environment-specific variables for accessing different Firebase projects (development, testing, production).
- **Build Configuration:**
    - Separate build configurations for development and production environments to prevent accidental deployment of debug builds.

Security

- **Authentication:**
    - Use Firebase Authentication to manage user logins during testing.
- **Access Control:**
    - Ensure only authorized testers have access to the test environment.
- **Data Encryption:**
    - Encrypt sensitive test data to prevent unauthorized access.

Logging and Monitoring

- **Firebase Crashlytics:**
  - For monitoring app crashes and non-fatal errors during testing.
- **Logcat:**
  - Use Android Logcat for real-time logging and debugging.

5.4 Test Cases

Test Case 1: User Registration

Test Case ID: TC001
Description: Verify that a new user can register successfully.
Preconditions: The user is not already registered.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the registration screen.
3. Enter valid details (name, email, address, phone, password).
4. Tap the 'Register' button.
5. Verify email if email verification is required. **Expected Results:**

- The user should be registered successfully and redirected to the login screen.
- A confirmation email is sent if email verification is required.

Test Case 2: User Login

Test Case ID: TC002
Description: Verify that a registered user can log in successfully.
Preconditions: The user is registered and email verified (if applicable).
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the login screen.
3. Enter valid email and password.
4. Tap the 'Login' button. **Expected Results:**

- The user should be logged in and redirected to the home screen.

Test Case 3: View Shops

Test Case ID: TC003
Description: Verify that a logged-in user can view a list of shops.
Preconditions: The user is logged in.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the 'Shops' section. **Expected Results:**

- The app should display a list of shops available nearby.

Test Case 4: Search for Shops

Test Case ID: TC004
Description: Verify that a user can search for shops using keywords.
Preconditions: The user is logged in.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the 'Shops' section.
3. Enter a search keyword (e.g., "Pizza").
4. Tap the 'Search' button. **Expected Results:**

- The app should display a list of shops matching the search criteria.

Test Case 5: View Shop Details

Test Case ID: TC005
Description: Verify that a user can view details of a selected shop.
Preconditions: The user is logged in.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the 'Shops' section.
3. Tap on a shop from the list. **Expected Results:**

- The app should display the details of the selected shop, including its menu, reviews, and ratings.

Test Case 6: Place an Order

Test Case ID: TC006
Description: Verify that a user can place an order from a shop.
Preconditions: The user is logged in, and the selected shop offers delivery.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the 'Shops' section.
3. Select a shop from the list.
4. Choose a food item from the shop's menu.
5. Tap the 'Order' button.
6. Enter delivery address and confirm the order. **Expected Results:**

- The order should be placed successfully, and the user should receive an order confirmation.

Test Case 7: Order History

Test Case ID: TC007
Description: Verify that a user can view their order history.
Preconditions: The user is logged in and has placed orders.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the 'Order History' section. **Expected Results:**

- The app should display a list of the user's past orders.

Test Case 8: Cancel an Order

Test Case ID: TC008
Description: Verify that a user can cancel an order before it is accepted by the shop.
Preconditions: The user is logged in, and the order is not yet accepted.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the 'Order History' section.
3. Select a pending order.
4. Tap the 'Cancel Order' button. **Expected Results:**

- The order should be canceled, and the user should receive a cancellation confirmation.

Test Case 9: Rate and Review Food

Test Case ID: TC009
Description: Verify that a user can rate and review a food item after delivery.
Preconditions: The user is logged in, and the order is delivered.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the 'Order History' section.
3. Select a delivered order.
4. Tap on the food item to rate and review.
5. Submit the rating and review. **Expected Results:**

- The rating and review should be submitted successfully and displayed under the food item.

Test Case 10: Shopkeeper Manages Shop

Test Case ID: TC010
Description: Verify that a shopkeeper can add, update, and delete food items.
Preconditions: The shopkeeper is logged in.
Test Steps:

1. Open the DaFoodies app.
2. Navigate to the 'Manage Shop' section.
3. Add a new food item with details.
4. Update an existing food item.
5. Delete a food item. **Expected Results:**

- The food items should be added, updated, and deleted successfully.

CHAPTER 06

User Manual

6.1 User Manual (Seller)

1. Introduction Welcome to DaFoodies! As a seller, you have the ability to manage your shop, add and update food items, and process orders. This guide will help you navigate through your features and functionalities.

2. Getting Started

- **Login:**

    1. Open the DaFoodies app.
    2. Enter your email and password.
    3. Tap the 'Login' button.

3. Managing Your Shop

- **Add a New Food Item:**

    1. Navigate to the 'Manage Shop' section.
    2. Tap the 'Add Food Item' button.
    3. Enter food item details (name, image, price, description, etc.).
    4. Tap 'Save' to add the food item to your menu.
- **Update an Existing Food Item:**
    1. Navigate to the 'Manage Shop' section.
    2. Select the food item you want to update.
    3. Make the necessary changes.
    4. Tap 'Save' to update the food item.
- **Delete a Food Item:**
    1. Navigate to the 'Manage Shop' section.
    2. Select the food item you want to delete.
    3. Tap the 'Delete' button.

4. Processing Orders

- **Accept or Decline Orders:**

    1. Navigate to the 'Orders' section.
    2. Select a pending order.

3. Tap 'Accept' to confirm the order or 'Decline' to reject it.

- **Set Preparation Time:**
-
    1. After accepting an order, set the preparation time based on the complexity of the order.
    2. Tap 'Save' to confirm the preparation time.

5. Viewing Reviews and Ratings

- Navigate to the 'Reviews' section.
- View customer feedback and ratings for your shop and food items.

6. Support

- For any issues or questions, contact support via the 'Help' section in the app.

6.2 User Manual (Customer)

1. Introduction Welcome to DaFoodies! This guide will help you find and order the best-reviewed food options nearby with ease.

2. Getting Started

- **Registration:**

    1. Open the DaFoodies app.
    2. Tap the 'Register' button.
    3. Enter your details (name, email, address, phone, password).
    4. Tap 'Register' to create your account.
- **Login:**
    1. Enter your email and password.
    2. Tap 'Login' to access your account.

3. Finding Food and Shops

- **View Shops:**

    1. Navigate to the Home Screen.
    2. Browse the list of available shops.
    3. For trending go to the trending section.
- **Search for Shops:**

1. Use the search bar to find specific shops or cuisines.
2. Enter a keyword (e.g., "Pizza") and tap 'Search'.

4. Placing an Order

- **Select a Shop:**
    1. Tap on a shop to view its menu.
- **Choose a Food Item:**
    1. Select the food item you want to order.
    2. Tap 'Order'.
- **Enter Delivery Details:**
    1. Enter your delivery address.
    2. Confirm the order.

5. Viewing Order History

- Navigate to the 'Order History' section to view your past orders.

6. Rating and Reviewing

- **Rate and Review Food:**
    1. Go to 'Order History'.
    2. Select a delivered order.
    3. Tap on the food item to submit a rating and review.

7. Support

- For any issues or questions, contact support via the 'Help' section in the app.

6.3 User Manual (Delivery Man)

User Manual (Delivery Man)

1. Introduction Welcome to DaFoodies! As a delivery man, you can manage delivery requests and update delivery statuses. This guide will help you navigate through your features and functionalities.

2. Getting Started

- **Login:**

    1. Open the DaFoodies app.
    2. Enter your email and password.
    3. Tap the 'Login' button.

3. Managing Deliveries

- **Accepting Delivery Requests:**
    1. Navigate to the 'Deliveries' section.
    2. View the list of pending delivery requests.
    3. Tap 'Accept' to accept a delivery request.
- **Updating Delivery Status:**
    1. Once you have picked up the order, update the status to 'On The Way'.
    2. After delivering the order, update the status to 'Delivered'.

4. Viewing Delivery History

- Navigate to the 'Delivery History' section to view your past deliveries.

5. Support

- For any issues or questions, contact support via the 'Help' section in the app.

6.4 User Manual (Admin)

1. Introduction Welcome to DaFoodies! As an admin, you have the ability to verify and manage shops, as well as oversee all functionalities. This guide will help you navigate through your features and functionalities.

2. Getting Started

- **Login:**
    1. Open the DaFoodies app.
    2. Enter your admin email and password.
    3. Tap the 'Login' button.

3. Managing Shops

- **Verify Shops:**
    1. Navigate to the Home Screen.
    2. View the list of shops awaiting verification.
    3. Tap 'Verify' to approve a shop.
- **Add or Remove Shops:**
    1. To add a shop, navigate to the 'Add Shop' section and enter shop details.
    2. To remove a shop, select the shop from the list and tap 'Remove'.

4. Overseeing Functions

- **View All Functions:**
    1. Same as Customer and shopkeeper.

CHAPTER 07

Project Summery

7.1 Github Link

https://github.com/MaishaFahmida16/defence

7.2 Critical Evolution

Strengths

1. User-Friendly Interface DaFoodies offers a simple and intuitive user interface that makes it easy for users to navigate through the app. The design ensures that even users with minimal technical skills can easily search for food, place orders, and manage their profiles.

2. Comprehensive User Roles The app supports four types of users: customers (teachers and students), shopkeepers, delivery men, and admins. Each role has distinct functionalities tailored to their specific needs, enhancing the app's usability and efficiency.

3. Efficient Order Management The order management system is well-designed, allowing customers to place orders, shopkeepers to manage orders, and delivery men to track and deliver orders. This seamless integration ensures that the entire order process is smooth and efficient.

4. Firebase Integration Using Firebase as the backend ensures real-time updates and secure data management. This integration supports features like authentication, database management, and notifications, providing a reliable backend infrastructure for the app.

Weaknesses

1. Lack of Payment Integration Currently, DaFoodies only supports cash on delivery, which might limit its appeal to users who prefer online payment methods. Integrating payment gateways could enhance user convenience and expand the app's user base.

2. Absence of Map Integration While the app plans to include map integration in the future, the current absence of this feature might inconvenience users who want to visually locate food shops and track deliveries.

3. Limited Customization The app lacks advanced customization options for users, such as personalizing the app's appearance or setting preferences for food recommendations. Adding these features could improve user engagement and satisfaction.

4. Dependency on Internet Connection as a Firebase-based app, DaFoodies requires a stable internet connection for real-time updates and functionality. Users with poor internet connectivity might experience disruptions and delays, affecting their overall experience.

Feature Effectiveness

1. Rating and Review System The rating and review system effectively allows users to provide feedback on food and shops, helping others make informed decisions. This feature promotes transparency and encourages food shops to maintain high-quality standards.

2. Role-Based Access Control The role-based access control ensures that each user type has access to relevant features and functionalities, preventing unauthorized access and maintaining data security.

3. Order Tracking The ability to track orders in real-time enhances the user experience by keeping customers informed about the status of their orders. This feature is particularly beneficial for managing expectations and improving customer satisfaction.

Future Improvements

1. Implementing Payment Gateways Integrating payment gateways like PayPal, Stripe, or local payment methods could offer users more flexibility and convenience, making the app more competitive in the market.

2. Adding Map Integration Incorporating map features for shop locations and delivery tracking would significantly enhance the user experience, providing a visual representation of food shops and real-time delivery updates.

3. Enhancing Customization Options Introducing customization options, such as theme selection and personalized food recommendations, could increase user engagement and make the app more appealing to a broader audience.

4. Offline Functionality Developing offline functionalities, such as viewing previously loaded food shops and menus, could help users with intermittent internet access continue using the app without interruptions

Conclusion

DaFoodies is a well-constructed app with several strengths, including a user-friendly interface, comprehensive user roles, efficient order management, and robust backend integration. However, there are areas for improvement, such as integrating payment options, adding map features, enhancing customization, and ensuring offline functionality. By addressing these weaknesses and implementing the suggested improvements, DaFoodies has the potential to become a highly popular and widely used app in university campuses and beyond.

7.3 Limitations

Limitations of DaFoodies

1. Limited Payment Options

Currently, DaFoodies only supports cash on delivery, which restricts users who prefer online payment methods. This limitation may deter potential users who are accustomed to the convenience of digital payments.

2. Absence of Map Integration

The lack of integrated maps means users cannot visually locate food shops or track delivery routes within the app. This could inconvenience users who rely on visual navigation and want to see real-time delivery progress.

3. Internet Dependency

DaFoodies relies heavily on a stable internet connection due to its integration with Firebase for real-time updates. Users with poor internet connectivity may face disruptions, affecting their ability to place orders or receive timely notifications.

4. Limited Customization and Personalization

The app currently lacks advanced customization options, such as theme changes or personalized food recommendations. This limitation may reduce user engagement and satisfaction, as users cannot tailor their experience to their preferences.

5. Scalability Concerns

While the current infrastructure supports the app's functionalities, rapid growth in user numbers could strain the backend system. This potential scalability issue might lead to slower response times or system crashes under heavy load.

6. Minimal Offline Functionality

DaFoodies does not support offline functionality. Users cannot browse menus or place orders without an internet connection, which can be a significant drawback in areas with unstable connectivity.

7. Limited Geographic Scope

Designed for university campuses, DaFoodies has a limited geographic scope. This focus restricts its usability to specific locations, limiting its market potential and user base.

8. Basic Delivery Management

The current delivery management system is basic and may not efficiently handle a high volume of orders or complex delivery logistics. Enhancing this system is crucial for maintaining service quality as the user base grows.

9. No Advanced Analytics

The app lacks advanced analytics and reporting features for shopkeepers and admins. These features are essential for understanding user behavior, optimizing operations, and improving service offerings based on data insights.

10. Security and Privacy Risks

While Firebase provides a secure backend, the app needs to implement comprehensive security measures to protect user data and privacy. Without rigorous security protocols, user data could be vulnerable to breaches and misuse

7.4 Obstacles and Achievements

Obstacles

1. Development Complexity Developing DaFoodies required integrating multiple functionalities, such as user management, order processing, and real-time updates, which posed significant challenges. Managing these complexities to ensure smooth and efficient operation was a critical hurdle.

2. Resource Constraints As a solo developer, managing time and resources efficiently was challenging. Balancing various development tasks, testing, and bug fixing required meticulous planning and prioritization.

3. Firebase Integration While Firebase provides a robust backend solution, integrating its various services (like authentication, database, and real-time updates) and ensuring seamless communication between them required extensive learning and troubleshooting.

4. User Experience Design Creating an intuitive and user-friendly interface for four distinct user roles (customers, shopkeepers, delivery men, and admins) was challenging. Ensuring that each user type had access to the necessary features without overwhelming the interface required careful design considerations.

5. Real-Time Order Management Implementing real-time order management to handle dynamic updates for order status and notifications involved dealing with complex asynchronous operations and ensuring data consistency across the system.

6. Security Concerns Ensuring the security of user data and protecting against potential breaches was a significant concern. Implementing robust authentication and authorization mechanisms was essential but challenging.

7. Testing and Debugging Comprehensive testing to identify and fix bugs, ensure compatibility across different devices, and verify the functionality of all features was time-consuming and required a structured approach.

Achievements

1. Successful Development and Deployment Despite the challenges, DaFoodies was successfully developed and deployed, offering a fully functional app that meets its core objectives. This accomplishment demonstrates the effectiveness of the chosen technologies and development strategies.

2. Intuitive User Interface The app features a user-friendly interface that allows seamless navigation for all user types. This achievement ensures that even users with minimal technical knowledge can easily use the app to find food, place orders, and manage their activities.

3. Real-Time Order Processing DaFoodies efficiently manages real-time order processing, providing users with immediate updates on their order status. This feature enhances the user experience by keeping customers informed and engaged.

4. Role-Based Access Control Implementing a comprehensive role-based access control system was a significant achievement. This system ensures that each user type has access to the relevant features, enhancing security and usability.

5. Robust Backend with Firebase Successfully integrating Firebase for authentication, database management, and real-time updates was a major milestone. This integration provides a reliable and scalable backend infrastructure for the app.

6. Foundation for Future Enhancements The current version of DaFoodies provides a solid foundation for future enhancements. Features such as payment integration, map functionality, and advanced customization options can be added to further improve the app.

7. Learning and Skill Development The development process provided valuable learning opportunities and helped enhance skills in Flutter, Firebase, and mobile app development. These skills will be beneficial for future projects and professional growth.

7.5 Future Scope

1. Payment Integration

A key area for future enhancement is the integration of various online payment methods such as credit/debit cards, mobile wallets, and digital banking. This feature will offer users greater convenience and flexibility, catering to their diverse payment preferences and enhancing the overall user experience.

2. Map Integration

Adding map functionality will enable users to visually locate nearby food shops and track their delivery in real-time. This feature can significantly improve user satisfaction by providing them with a visual representation of their order status and the shop locations.

3. Advanced Analytics and Reporting

Developing advanced analytics and reporting tools for shopkeepers and admins will help them gain valuable insights into user behavior, sales trends, and operational performance. These insights can drive data-driven decision-making and help optimize the business processes.

4. Enhanced Customization and Personalization

Introducing advanced customization options and personalized recommendations can enhance user engagement. Features like theme changes, personalized food suggestions based on user preferences, and tailored notifications will create a more personalized experience for users.

5. Offline Functionality

Implementing offline functionality will allow users to browse menus and add items to their cart without an internet connection. This capability will ensure a seamless experience even in areas with unstable connectivity, improving accessibility and user retention.

6. Expanded Geographic Scope

While DaFoodies is currently focused on university campuses, expanding its scope to include nearby neighborhoods, cities, or even other educational institutions can broaden its user base and market reach. This expansion will also involve scaling the backend infrastructure to support the increased user load.

7. Improved Delivery Management

Enhancing the delivery management system to handle higher order volumes and complex logistics can improve efficiency. Features like route optimization, real-time delivery tracking, and better coordination between delivery personnel and restaurants will be crucial for scaling operations.

8. Loyalty Programs and Discounts

Introducing loyalty programs and discounts can attract and retain customers. Features like reward points, special offers, and referral bonuses will encourage repeat orders and foster user loyalty.

9. Multi-Platform Support

Expanding DaFoodies to support multiple platforms, such as iOS and web applications, can increase its accessibility and user base. Ensuring a consistent experience across different platforms will be crucial for maintaining user satisfaction.

10. Enhanced Security Measures

Continuously improving security measures to protect user data and privacy is essential. Implementing advanced authentication methods, encryption protocols, and regular security audits will help safeguard sensitive information and build user trust.

11. Voice Command Integration

Integrating voice command features can make the app more accessible and user-friendly. Users will be able to search for food, place orders, and navigate the app using voice commands, adding a new level of convenience.

12. AI and Machine Learning

Leveraging AI and machine learning can enhance various aspects of the app, from personalized recommendations to predictive analytics. Implementing AI-driven features can help improve user experience, operational efficiency, and business insights

The future scope of DaFoodies is vast, with numerous opportunities for enhancement and expansion. By focusing on payment integration, map functionality, advanced analytics, customization, offline capabilities, geographic expansion, improved delivery management, loyalty programs, multi-platform support, security, voice command integration.