Codewave pty ltd

FOOD DELIVERY APP PROJECT PROPOSAL

Prepared By:

LETHOKUHLE MNDZEBELE 202204046
SANDZA LUKHELE 202201121
ZINHLE MATFONSI 202202481
DLAMINI TANDZILE 202201887

+26876350512
www.codewave.com
Matsapha, Eswatini

Contents

Introduction

The food industry has witnessed a significant shift toward digital platforms, allowing customers to order meals online conveniently. With the growing demand for fast and efficient food delivery services, local food outlets often struggle to compete with larger chains due to the lack of resources and digital infrastructure. A well-integrated food delivery system can bridge this gap, providing a seamless solution for local restaurants to reach a wider customer base. Eswatini has a growing need for convenient food delivery services, particularly in urban areas like Mbabane and Manzini. Existing options may be limited, or not technologically advanced. This project aims to address this gap.

History

Traditional food delivery services have been in existence for decades, relying on phone orders and manual delivery coordination. However, with advancements in technology, app-based food delivery services have revolutionized the industry. Food delivery in Eswatini has traditionally been informal and limited, primarily operating through phone in orders and personal delivery arrangement by individual restaurants. The introduction of mobile technology and increased internet penetration in Eswatini opened new possibilities of digital services including online food ordering, the demand of convenience formalized food delivery platforms.

Objectives

The primary goal of this project is to develop a user-friendly and efficient food delivery system that connects local food outlets with customers. The system will include a mobile and web-based application where users can browse menus, place orders, and track deliveries in real-time. It will also integrate features such as secure payment options, customer reviews, and loyalty programs to enhance user experience and engagement.

The primary objectives of a food delivery app revolve around enhancing user experience, ensuring efficiency, and driving business growth.

- For Customers:
    - Provide an easy-to-use interface for browsing menus, placing orders, and tracking deliveries.
    - Ensure timely deliveries with real-time tracking and estimated delivery times.
    - Partner with multiple restaurants to offer diverse food choices.

- For Restaurants & Food outlets:
    - Help restaurants reach a larger customer base without heavy marketing costs.
    - Provide tools for order tracking, inventory management, and menu updates.
    - Enable restaurants to earn more through delivery and dine-in promotions.
    - Offer analytics on customer behavior, peak hours, and popular dishes.

- For the Business (App Owners):
    - get money or revenue that is earned through commissions, delivery fees, ads, and subscription models
    - Scale operations to new cities and regions for market expansions
    - Stay ahead with unique features (live tracking, group ordering, eco-friendly packaging).

Aims

This project will serve as a bridge between local restaurants and their customers, ensuring that small businesses thrive in an increasingly digital world.

- Develop a user-friendly mobile app for ordering food from local restaurants in Eswatini that will provide a reliable and efficient delivery service.
- Support local businesses by increasing their customer reach.
- Generate employment opportunities for delivery personnel.
- Offer a secure and convenient payment platform.

User/Client involvement

When developing a food delivery app, proper understanding of users' needs is crucial because they are the ones who will determine the success or failure of the app. User and client involvement is key in order to ensure that the app is tailored to the specific preferences, challenges, and opportunities in Eswatini. This section of the project plan will explore how users (restaurants, customers) will be involved in the development process.

- Research and Discovery Phase: From the beginning of the project users will be involved because it is key to understand their needs. This means engaging with local customers and the restaurants at Manzini Lifestyle to identify their needs. We will conduct surveys, questionnaires or interviews with local residents who are regulars in these restaurants. Questions could explore frequency of eating out, and willingness to use a digital platform. We will engage with the eateries in Manzini lifestyle to understand their capacity to handle online orders, their current delivery methods (if any), and their willingness to partner with an app.

- Design Phase: Once initial insights have been gathered, users and clients will help co-create the app's features and interface to ensure it resonates with their needs. Organize workshops or participatory design sessions where users brainstorm features. For example, they might request an interface that has both SiSwati and English languages, simple navigation for less tech-savvy users, or offline ordering capabilities given potential internet disruptions. We intend to collaborate with restaurant owners to design their side of the app. For example, an easy-to-use dashboard for managing orders. Simplicity and training needs is vital as users may lack digital tools. We will also work with delivery workers to map out a tracking system that accounts for Eswatini’s geography paved urban roads versus rural dirt tracks to ensure fair compensation based on distance or time.

- Development and Testing Phase: During the development phase, iterative testing with users and clients to ensure that the app meets real-world needs and functions smoothly. We will test how the app handles payments with a sample of customers. For instance, test how well the app handles payments whether paid using mobile money or over card payments. Still on the testing phase we will gather feedback on delivery times and food quality. Lastly we will test the app with restaurants to ensure order notifications are reliable, especially if internet connectivity is patchy.

Involving users and clients will help us to ensure the app is a solution for Eswatini. Customers get a convenient service from the comfort of their homes and restaurants gain new revenue streams. This collaborative approach also builds trust and loyalty because users will feel like they are valued if they are involved from the get go.

Risks

Technical risks such as software bugs and glitches may pose a great threat to the success of the software as these errors may give incorrect, unexpected crashes or cause software to malfunction temporarily. These can lead to frustration for users, potentially causing them to abandon the app and the frequent issues can negatively affect the app's reputation and result in poor reviews. Bugs can interfere with order processing, delivery tracking, and payment processing, impacting overall operations. If users cannot place orders due to app failures, the service could lose revenue. The mitigation strategies would involve validating individual components of the application to ensure they work correctly in isolation. Also, test interactions between different components and third-party services to identify interface issues. Moreover, we will assess the entire system’s compliance with specified requirements through end-to-end testing. To add on that, we will gather feedback from real users on the app’s functionality and usability before the official launch and maintain a routine for updating the app and patching any identified bugs or issues. Keeping users informed about updates to enhance their confidence in the app's reliability.

Regulatory Risks such as non-compliance with Food Safety Regulations is caused by **l**imited understanding or awareness of relevant food safety regulations among vendors and delivery personnel can lead to non-compliance. Also, inadequate communication regarding safety standards and procedures between restaurants, delivery partners, and the food delivery platform. To add on that, insufficient training for staff involved in food handling, preparation, and delivery may result in improper practices that violate safety standards. This negligence’s may result to fines, sanctions, or legal action from regulatory bodies for failing to comply with food safety standards and can impose significant costs on the business. Moreover, non-compliance may result in the suspension or revocation of business licenses, restricting the ability to operate legally. Furthermore, the negative publicity resulting from safety violations or health risks can harm the brand's reputation and erode customer trust.

To mitigation these risks, we will implement training programs for all staff involved in food handling, preparation, and delivery to ensure they understand food safety practices and regulations. We will then develop and document clear policies and procedures for food safety compliance that all partners and employees must follow. After that, we will utilize technology to monitor food temperatures, track food origins, and record compliance with safety standards in real time. To add on that, we will perform periodic compliance audits of vendors and delivery processes to identify and address any non-compliance issues.

Reputational Risks such as negative Customer Feedback which may be caused by frequent delivery delays or inaccuracies in order fulfilment which may frustrate customers. Technical glitches in the app, such as slow loading times, crashes, or inaccurate tracking, can lead to dissatisfaction. Also, delivering food that is not fresh, has poor packaging, or does not meet customer expectations can result in negative reviews. Consistent negative feedback can harm the overall perception of the brand in the market, and a damaged reputation can take time to repair, requiring significant efforts in marketing and customer engagement. To mitigate these risks, we will implement performance monitoring systems to track delivery times and order accuracy. This will establish clear quality standards for food and delivery services, and regularly communicate them to partners. We will also provide multiple channels for customer support (e.g., chat, phone, email) and ensure timely responses. To add on that, we will train customer service representatives to handle complaints effectively and empathetically. To ensure all these, we will design an intuitive user interface for the application that simplifies navigation and enhances the checkout experience. Lastly, we will regularly gather user feedback to inform continuous improvements to the app's design and functionality.

Cybersecurity Risks is another risk that the software may face, these risks includes data breaches which may pose a great threat to customers as it may expose sensitive customer data since food delivery apps collect information from users. The personal information include, names, addresses, payment information, and phone numbers. Breaches can lead to unauthorized access to this sensitive data, resulting in identity theft or financial fraud. Also, payment card information is often stored for convenience and if not secured properly, it can be a target for cybercriminals. The mismanagement of payment processing can lead to a breach of payment card industry (PCI) compliance, resulting in fines. To mitigate these risks, we will encrypt sensitive data both at rest and in transit using strong encryption standards (e.g., AES-256). This encryption will protect data from unauthorized access, making it unreadable without the proper decryption keys. Also, we will require users to verify their identity through password verification and also send them OTP to their phone. This will provide an extra layer of security, reducing the risk of unauthorized access even if credentials are compromised. To add on that, we will conduct periodic audits and penetration tests to identify vulnerabilities in the application. This will help in proactive identification and remediation of security weaknesses before they can be exploited. Collect only the data that is necessary for the application to function effectively and properly anonymize sensitive information. This reduces the volume of sensitive data that could be compromised in the event of a breach.

Standards and procedures

Effective development processes are critical to the success of projects. This Software Development Standard provides a clear set of procures that will help the development of the software. These requirements are mandatory and must be adhered to by all employees' consultants and contractors involved in the development or modification of the software.

All software development projects, including maintenance projects, must follow these standards.

General guidelines:

- Simplicity of use
- Ruggedness (difficult to misuse, kind to errors
- encountered)
- Delivered on time and when needed
- Reliability
- Efficiency (fast enough for the purpose it was created)
- Minimum development cost
- Conform to standards
- Clear, accurate and precise user documentation, clear, accurate and precise technical documentation

A comprehensive Standards document is available that will outline:

- Analysis procedures
- Data protection procedure
- Implementation procedure
- Coding procedure
- Design procedures
- SQL procedures
- Testing procedures
- Post implementation procedure

Organization of the Project

Relation to Other Projects

This food delivery system is closely related to other digital transformation initiatives within the food service industry. It aligns with projects aimed at enhancing local business digital presence, improving supply chain logistics, and integrating cashless payment solutions. Additionally, the system can be extended to include AI-driven recommendations, partnerships with local grocery stores, and integration with ride-sharing services for optimized delivery.

Project Organization

The project follows a structured approach with distinct phases:

1. Planning Phase

Defining project scope, requirements, and feasibility. The project scope outlines the boundaries of the project by specifying what will be included and what will not. It helps prevent scope creep and ensures all stakeholders have a common understanding. It includes objectives, deliverables, milestones and constraints. Project requirements detail the specific needs that must be met for the project to be successful. These can be functional or non-functional. The project's feasibility must be evaluated to determine if it is viable technically, financially, and operationally.

2. Design Phase

The Design Phase lays the groundwork for the entire system by defining its structure, appearance, and data organization. This stage begins with establishing the system architecture, where developers outline the software's core components, their interactions, and the technology stack to be used. Key considerations include scalability, security, and efficient data flow. Next, the UI/UX design process involves creating wireframes and interactive prototypes to visualize the user interface. Designers focus on usability, accessibility, and aesthetic consistency, refining layouts based on stakeholder and user feedback. Simultaneously, the database structure is meticulously planned, with tables, relationships, and indexing strategies designed to ensure optimal performance, data integrity, and efficient storage.

3. Development Phase

With the design finalized, the Development Phase brings the system to life through coding and integration. Developers start by writing clean, modular code for both frontend (user-facing interfaces) and backend (server logic, APIs, and authentication systems), following best practices and version control protocols. The integration process connects these components, ensuring seamless communication between the frontend, backend, and any third-party services (e.g., payment gateways or cloud APIs). Finally, the backend setup involves deploying and configuring servers, databases, and security measures (like encryption and access controls) to create a stable foundation for the system.

4. Testing Phase

Before launch, the system undergoes rigorous evaluation in the Testing Phase to identify and resolve issues. Quality assurance (QA) teams conduct various tests—including unit, integration, and system testing—to verify functionality and compatibility. Any bugs discovered are logged and prioritized for bug fixing, with critical issues addressed immediately to prevent disruptions. Additionally, performance testing simulates high-traffic scenarios and stress conditions to evaluate the system's speed, stability, and resource usage, allowing teams to optimize responsiveness before deployment.

5. Deployment Phase

The Deployment Phase marks the system's transition to a live environment. Developers launch the system on production servers, configuring domains, SSL certificates, and hosting setups to ensure secure and reliable access. During user onboarding, training materials, tutorials, and support channels (e.g., help desks or chatbots) are rolled out to facilitate a smooth transition for end-users. Early feedback is collected to make quick adjustments and ensure user satisfaction from day one.

6. Maintenance and Upgrades Phase

Post-launch, the system enters the Maintenance & Upgrades Phase, where continuous monitoring tracks performance, errors, and security threats using tools like log analyzers and alerts. Regular updates are released to patch vulnerabilities, introduce new features, and improve efficiency based on user feedback and technological advancements. This phase ensures the system remains scalable, secure, and aligned with evolving business needs over time.

Schedule and Training Plan for Food Delivery System Development

To ensure the successful development of our food delivery platform, the team will follow a structured training and implementation timeline aligned with each phase of the project. In Month 1, the Project Manager will complete Agile/Scrum and risk management training to establish efficient workflows for coordinating restaurant partnerships, delivery logistics, and customer service processes. Simultaneously, the Business Analyst will focus on requirement gathering techniques specific to food ordering workflows, payment integrations, and stakeholder management with restaurants and delivery personnel.

As we move into Month 2, the Software Architect will undergo training on system design and security best practices, with special emphasis on high-traffic handling during peak meal times and secure payment processing. During this same period, Developers will train on programming frameworks and API integrations critical for our food delivery system - including third-party payment gateways, geo-location services for delivery tracking, and restaurant menu management systems. Our UI/UX Designer will refine skills in design tools and user research methods to create an intuitive ordering interface that accommodates diverse customer needs, from quick meal selection to special dietary filters.

Month 3 sees our QA Engineer training on both automated and manual testing tools, with particular focus on order flow validation, real-time delivery status updates, and multi-platform compatibility testing (web, iOS, Android). The Database Administrator will concurrently specialize in data security and performance tuning to handle high volumes of simultaneous orders while protecting sensitive customer and payment information.

Finally, in Month 4, our Marketing Team will complete training in digital marketing strategies tailored for food delivery platforms, including geo-targeted promotions, loyalty program implementation, and crisis management for delivery delays. They'll also master customer engagement techniques specific to the food industry, ensuring smooth onboarding for both restaurants and end-users when we launch.

This phased training approach ensures each team member develops precisely the skills needed at the right moment, keeping our food delivery platform development on schedule while addressing the unique challenges of the on-demand meal delivery market. The schedule allows for overlapping competencies where needed, particularly in integrating restaurant backend systems with our customer-facing platform.

Training will be conducted via:

- Internal Workshops (led by senior team members)
- External Certification (if required, e.g., cloud platforms)
- Self-Paced Online Courses (for flexible upskilling)

Governance and Reporting Structure

- Weekly Standups: Daily for development teams, weekly for stakeholders.
- Sprint Reviews: Every 2 weeks (Agile methodology).
- Steering Committee Meetings: Monthly (for high-level updates).

Project phases

The process of creating a system like this requires a software development model that is suitable for our project. So for this project we will use the Waterfall model. Below is an outline of the project phases for developing a food delivery app in Eswatini. For each phase the tasks will be discussed as well as potential milestones and how to ascertain milestone completion.

Project Phases in the Waterfall Model

- Requirements Analysis: In this phase the system's services, constraints, and goals are established by consultation with system users, we will gather and document all requirements for the app based on user and client input. We will conduct surveys and interviews with customers and restaurant owners. App features will be defined (e.g. order placement, restaurant menus, delivery tracking). We will further document requirements in a Software Requirements Specification (SRS) and have stakeholder approval of requirements. This document will address all identified needs (e.g., SiSwati language support, affordability features).

- Design and Development: Once we are done with the first phase we will develop the application using a service-oriented architecture (SOA) approach to ensure integration with existing systems and support core business objectives. Create wireframes and UI/UX mockups tailored to Eswatini users (simple, multilingual), plan database structure and specify hardware/software needs. Lastly a detailed project timeline and resource plan will be developed, this will help to track projects operations, budgeting and progress.

- System Implementation (Coding): This stage is for our programmers whereby they build the app based on the design specifications listed in phase one. Develop front-end, mobile app for users and dashboard for restaurants. Build back-end which is the server, database and APIs for order processing and tracking. Then integrate payment systems which will be MTN Mobile Money as well as card payments. Lastly perform unit testing on individual components.

- Testing: Here as a team we ensure the app meets requirements and works in Eswatini's real-world conditions. Conduct integration testing for example make sure app connects to payment gateways. Perform system testing such as end-to-end functionality, order placement to delivery. Run user acceptance testing with the customers, restaurants, and delivery workers. Test if app does not malfunction during peak order times. Then lastly we fix bugs identified during the testing process.

- Maintenance: Use project management tools to monitor project progress, including modules for planning, monitoring, and data collection. This ensures that the project stays on track and any deviations are promptly addressed. Monitor app performance like crash reports, server uptime. Release updates for new features or bug fixes. Respond to user and client feedback via support channels.

This outlined plan provides a robust framework for managing app design projects, however it is essential to remain adaptable to changes and challenges that may arise. Flexibility in project management tools and methodologies can help accommodate evolving project requirements and ensure successful project completion. This structured Waterfall approach ensures clarity and control, ideal for a first-of-its-kind app in Eswatini, but requires diligent milestone tracking to avoid delays along the critical path.

Requirements analysis and design

Requirements analysis and design are critical phases in the software development lifecycle. For the "Dial a Delivery" food delivery app, employing effective methods and techniques will ensure that the final product meets user needs and business objectives. This combination of comprehensive requirements analysis techniques and robust design methods, supported by appropriate tools and resources, will ensure that "Dial a Delivery" effectively meets the needs of its users and stakeholders. This structured approach facilitates clear communication and collaboration across all phases of the project, paving the way for a successful food delivery service application. Regular review and adaptation of these methods throughout the project lifecycle will further enhance the outcomes.

The first phase is the requirements analysis which focuses on the methods and techniques of how the project will be executed. These methods include conduct interviews with stakeholders such as potential users, restaurant partners, and delivery personnel to gather insights on their needs and expectations. The resources needed to accomplish this method includes interview questions, a list of stakeholders, note-taking tools, and possibly recording equipment (with consent). Another method is using surveys and questionnaires where we will use surveys to collect quantitative and qualitative data from a larger audience to understand user preferences and pain points. In order to accomplish this method, we will use survey creation tools such as google forms, survey monkey and distribution channels such as email, social media.

Also, we will make use of workshops and focus groups where will organize workshops and focus groups to encourage discussion among users and stakeholders, allowing for collaborative idea generation and prioritization. The will use facilitation materials such as whiteboards, post-it notes, and a capable facilitator. In addition, we will define use cases to represent the interactions between users and the system, helping to clarify functional requirements. We will make use of case templates, UML tools like Lucid chart. In addition to that, we will create detailed user personas to represent different user types and their specific needs, preferences, and behaviours. We will use persona templates, demographic data, research materials to accomplish this method.

The second phase would be requirements design which also has its methods and techniques which outlines the requirements for designing the software. These methods include Wire framing where we will create wireframes to outline the basic structure and layout of the app, focusing on user interface design and navigation. The tools to do wire framing include Sketch. Also, another method will be developing flow diagrams to visualize the user's journey through the app, identifying key interactions and decision points. We will use diagramming software tools such as Lucidchart. To add on that, we will document the functional requirements (specific behaviours, features) and non-functional requirements (performance, security, usability). To do this method we will use requirements specification templates, collaborative documentation tools such as confluence and Google Docs.

Also, we will make use of Architecture Design where we will define the system architecture, including database designs, server-side structures, and client-side components. The resources needed to accomplish this task would be Architecture design tools such as Draw.io, and database design software such as MySQL. In addition, we utilize established design patterns such as MODEL View Controller (MVC) to create scalable and maintainable application structures. We will make use of design pattern reference materials and development frameworks. Documentation Tools such as Google Docs, Confluence, or Notion for collaborative documentation and requirements gathering, is another technique we are going to make use of. Moreover, we will use GitHub which is already setup, for effective communication among team members and stakeholders to make sure all needs are catered for.

Design and Prototyping Tools are tools we will employ that we meet the demands of stake holding by designing the product they desire and paid for. These tools include Figma for UI/UX design and prototyping, and Sketch for wireframing. To add on that, we will use testing tools which will ensure that our product stakeholders needs as well as end user's needs. Among these tools is UserTesting for gathering user feedback on prototypes and usability testing. Another technique we will use is the Version Control Systems where we will use GitHub which will helps in keeping track of changes made during the design and development phases.

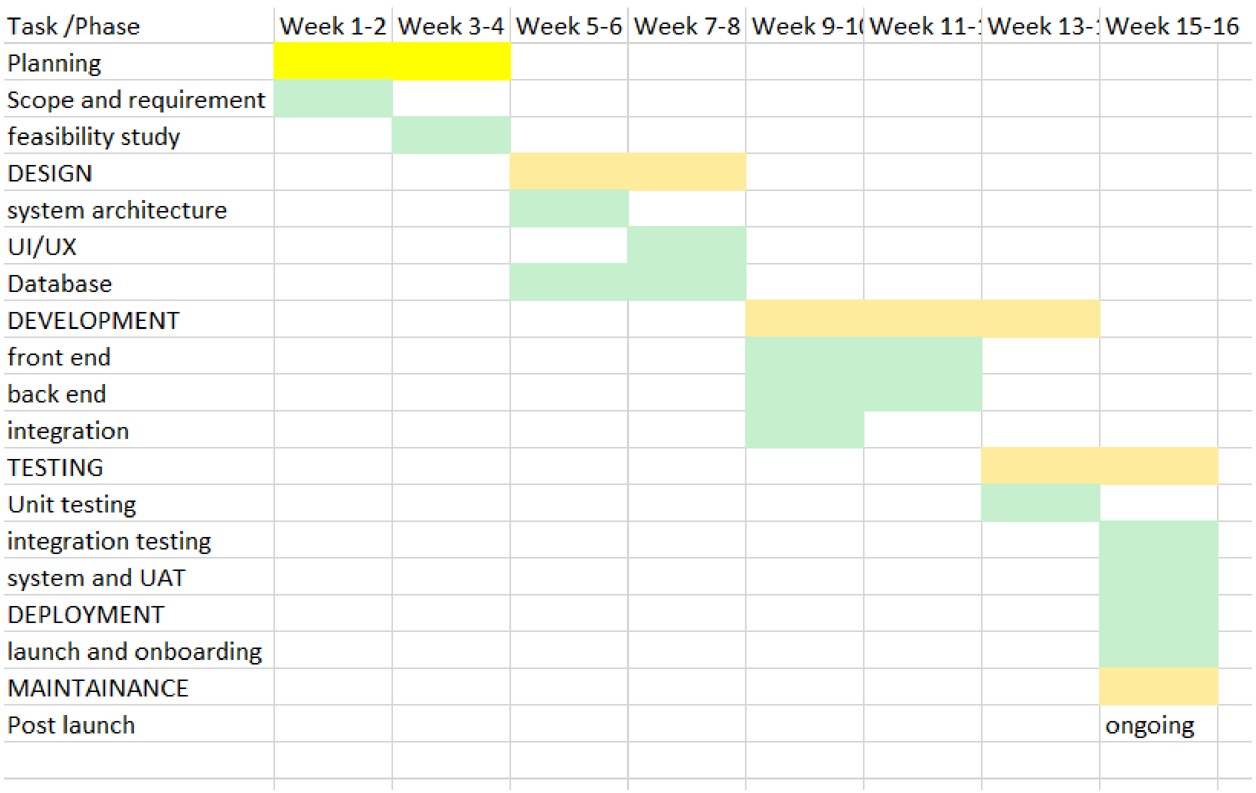
Technology stack (implementation)

The stack addresses different aspects of the system, including front-end, back-end, databases, payment processing and logistics.
The front-end is where users (customers, restaurants, and drivers) will interact with the system. The front-end will be mobile-centric, as mobile adoption is high in Eswatini. The back-end powers the application, handling business logic, user management, orders, payments. Payment integration is a critical part of the system. Eswatini's relatively high mobile money usage, mobile payment gateways will be prioritized. The delivery system will require real-time tracking, route optimization, and real-time communication between drivers and customers. To monitor performance and make data-driven decisions, the system will require analytics tools. The cloud infrastructure will be scalable, reliable, and able to handle sudden traffic spikes, especially during peak hours. Providing excellent customer support and engaging with customers is key to retaining users.

Tools and resources

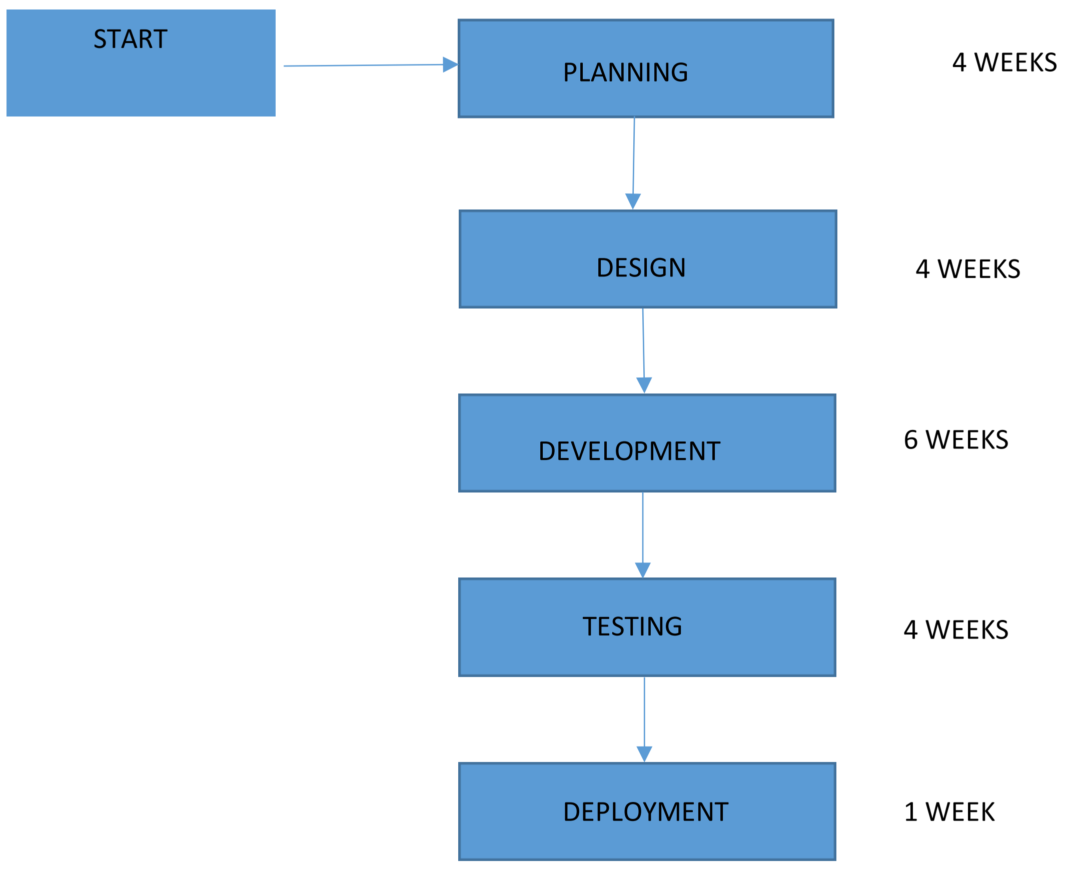
React Native: For cross-platform mobile app development (iOS and Android).
React.js or Vue.js: For building a responsive, dynamic web interface for restaurants and the admin panel.
App Development Tools: Android Studio, Xcode for iOS.
UI/UX Design Tools: Figma for wireframes and design mockups.
Django: A robust, scalable back-end framework if you prefer Python.
MongoDB: A NoSQL database for storing unstructured data like user preferences or delivery status logs.
SSL: For securing communication between the client and server.
Mobile Money Integration (MTN Mobile Money or Eswatini's eMali): Integration for local mobile payment systems.
Onfleet: for managing driver schedules, deliveries, and performance.
Google Analytics: For tracking website and mobile app traffic.
AWS: for storage and analysis.
SurveyMonkey: to gather feedback from users.
Chatbot platforms: for automated customer interactions.

Gantt Chart



Task /Phase	Week 1-2	Week 3-4	Week 5-6	Week 7-8	Week 9-1	Week 11-	Week 13-	Week 15-16
Planning	🟨	🟨						
Scope and requirement	🟩							
feasibility study		🟩						
DESIGN			🟨	🟨				
system architecture			🟩					
UI/UX				🟩				
Database			🟩	🟩				
DEVELOPMENT					🟨	🟨	🟨	
front end					🟩	🟩		
back end					🟩	🟩		
integration					🟩			
TESTING							🟨	🟨
Unit testing							🟩	
integration testing								🟩
system and UAT								🟩
DEPLOYMENT								🟩
launch and onboarding								🟩
MAINTAINANCE								🟨
Post launch								ongoing

PERT DIAGRAM



Testing Strategy Document for Food Delivery System

1. Test Environment & Equipment

The testing environment for our food delivery platform has been carefully designed to replicate real-world operational conditions, ensuring comprehensive validation of all system components before launch. For hardware requirements, we will utilize both cloud-based servers (AWS, Azure) configured for scalability testing and dedicated servers specifically for stress testing critical systems like order processing and payments. A diverse range of end-user devices will be employed, including 10+ smartphone and tablet models covering various iOS and Android versions to ensure full compatibility, along with GPS-enabled devices for accurate delivery route simulations and desktop browsers for web platform validation.

Our software testing toolkit includes robust solutions for every aspect of the platform. Mobile testing will leverage Android Studio and Xcode for emulation, complemented by Appium for cross-platform automated testing. To evaluate system performance under realistic loads, we'll use JMeter to simulate peak-order scenarios typical during meal rush hours, while Locust will handle

API load testing for payment and order systems. The automation framework includes Selenium WebDriver for web portal regression testing, with JIRA and TestRail integrated for comprehensive defect tracking and test case management. API endpoints will be rigorously validated using Postman.

Specialized testing setups address the unique requirements of food delivery operations. For location services, we've incorporated mock GPS tools to accurately simulate delivery personnel movement. The payment system testing utilizes sandbox environments from Stripe and PayPal to safely simulate transactions, including edge cases like failed payments and refund processing. Restaurant integration testing includes validation of POS system connectivity and thorough menu synchronization checks across all platform interfaces.

The staging environment has been configured to precisely mirror our production setup while using dummy data, completely isolated from live systems to prevent any potential data leaks. Network condition testing will simulate various real-world scenarios that delivery personnel might encounter, including 3G/4G/Wi-Fi connectivity issues and bandwidth limitations in different geographical areas. This comprehensive testing environment allows us to validate every aspect of the system, from individual API calls to full-scale delivery operations, under conditions that closely match actual usage scenarios.

2. Testing Phases & Procedures

Building on this robust testing infrastructure, we will implement a rigorous phased testing methodology. The process begins with unit testing of individual modules such as ordering, payments, and GPS tracking, where developers will validate each component in isolation. This is followed by component testing, where specific functions like adding items to cart or applying promo codes will be thoroughly examined. The integration phase will then test how these modules interact, particularly focusing on critical connections between frontend and backend API, payment gateway functionality, and the seamless operation of order processing with delivery tracking.

System testing will bring all components together in an environment that accurately mimics real world conditions, evaluating the complete food delivery ecosystem from customer order placement to restaurant fulfillment and final delivery. Performance testing will push the system to its limits, simulating high-traffic scenarios like weekend dinner rushes or special promotional events to ensure stability under peak loads. Security testing will verify all protective measures, including data encryption, prevention of SQL injections, and robust user authentication protocols.

The final phase involves User Acceptance Testing (UAT), where selected end-users including customers, restaurant staff, and delivery personnel will validate the system's usability and functionality in real operational scenarios. Throughout all phases, we will maintain meticulous testing procedures including detailed test case creation, systematic execution and logging of results, comprehensive bug reporting and resolution processes, and thorough retesting with regression analysis to ensure fixes don't introduce new issues. Only after successful completion of all these validation steps will the system be cleared for deployment, ensuring a reliable, secure, and high-performing food delivery platform at launch. The testing environment must mirror the production setup as closely as possible to ensure reliability.

Component	Specifications
Hardware	Servers: 2x AWS EC2 instances (Linux/Windows as needed) - Test Machines: Minimum 4GB RAM, x64 CPU
Software	OS: Windows 11/Linux (Ubuntu) - Databases: MySQL/PostgreSQL - Browsers: Chrome, Firefox, Edge (latest versions)
Test Tools	Unit Testing: JUnit (Java), PyTest (Python) - API Testing: Postman, Swagger - UI Testing: Selenium, Cypress - Performance: JMeter, LoadRunner
Test Data	Synthetic datasets (mimicking real-world scenarios) - Masked production data (if applicable for UAT)
CI/CD Integration	GitHub Actions for automated test execution - Docker for environment consistency

2. Test Planning

Testing will follow a structured approach across different levels:

Test Level	Objective	Performed By	Entry Criteria	Exit Criteria
Unit Testing	Verify individual modules/functions	Developers	Code committed	90%+ coverage, zero critical defects
| Integration Testing	Check interactions between components	Testers + Developers	Unit tests passed	All APIs/services communicate correctly
System Testing	Validate full system functionality	QA Team	Integration passed	All requirements met, defects logged & prioritized
UAT (User Acceptance Testing	Confirm business readiness	End-Users/Stakeholders	System testing signed off	Business approval, no showstoppers
Performance Testing	Assess speed, scalability, stability	Performance Team	System stable	Meets SLA (response time <2s, 99.9% uptime)

Security Testing	Identify vulnerabilities	Security Team	Code freeze	OWASP Top 10 resolved, pentest passed

- Database → API Layer → Business Logic → Frontend → External Systems

Testing Procedures

A. Test Case Design

- Derived from requirements (use traceability matrix).
- Includes positive, negative, and edge-case scenarios.

B. Defect Management

- Logging: JIRA/TestRail for tracking (severity: Critical/Major/Minor).
- Prioritization: Critical fixes within 24hrs; others per sprint plan.
- Retesting: Fixed defects verified before closure.

C. Automation Strategy

- Unit/API Tests: 100% automated (run on every commit).
- UI Tests: 70% automation (critical paths only).
- Performance/Security: Automated scans scheduled weekly.

D. Sign-off Criteria

- All test levels completed.
- 95% or more requirements covered.
- Zero Critical/High defects open.

➢ Risks & Mitigations

Risk	Mitigation
Test environment instability	Use Docker containers for consistency
Delayed defect resolution	Daily triage meetings with developers/test teams
Insufficient test data	Generate synthetic data scripts upfront.

Resources

During the execution of the food delivery app development, many resources will be needed. For the app to be properly developed using the Waterfall model we will require a combination of human resources, hardware, software tools and other supporting resources. In this section we will briefly look into the hardware and other tools needed for this project to be a success.

Human Resources

- Project manager (Miss Lethokuhle Mndzebele) to ensure design aligns with SRS and oversee timelines
- Designer (Mr Sandza Lukhele) who will develop wireframes and mockups tailored to Eswatini users.
- System Architect (Miss Tandzile Dlamini) who will be responsible for designing app architecture and database structure
- Programmer (Miss Zinhle Matfonsi) who is be responsible for translating ideas and requirements into functional software by writing, testing, debugging, and maintaining code
- Business analysts will conduct surveys and interviews and draft the Software Requirements Specification (SRS)

Technical resources

- IDEs like Visual Studio Code.
- Backend frameworks (Django).
- Database: MySQL for storing user data, orders, and restaurant menus.
- Payment gateways (MTN MOMO, Eswatini E-Mali).
- Version Control tools like Git hosted on GitHub for collaborative coding.
- Figma for UI design and prototyping

Hardware resources

- Laptops for analysts to document findings and draft SRS and for programmers to write code and conduct testing.
- Smartphones for field testing of existing apps or recording interviews.
- Servers for hosting databases

Financial Resources

- App Hosting & Infrastructure – monthly cloud/server costs.
- Third-party APIs/Services – Google Maps API and payment gateways.
- App Store Fees
- Marketing & Promotions – social media, influencer campaigns, ads.
- Legal & Licensing – business registration, food safety compliance.

Quality assurance

Quality assurance (QA) is a critical component in software development, ensuring that the product being developed meets the specified requirements and maintains a high standard of quality. Below is an outline of the organization, procedures, and aspects that will be included in the quality assurance plan for the "Dial a Delivery" app. In establishing a robust QA organization and following structured procedures, the app development team will ensure that the software meets the quality requirements outlined in the initial phases. The focus on collaboration, comprehensive testing, and continuous improvement will contribute significantly to delivering a reliable and high-quality food delivery service application. Regular reviews and adaptations of the quality assurance plan will be crucial for responding to emerging challenges and ensuring ongoing quality throughout the project lifecycle.

The Organization for Quality Assurance involves a QA Team Structure. This structure is made of QA Manager who is responsible for overseeing the QA process, coordinating between development and testing teams, and ensuring adherence to quality standards. It also involves the QA Engineer who is responsible for creating and executing test plans, conducting tests, documenting results, and reporting defects. Also, a QA Architect who is responsible for designing and implementing the QA processes, strategies and frameworks that guide the quality assurance efforts of the project. Lastly, it includes a QA analyst which collaborates with the QA team to ensure that acceptance criteria are clearly defined and understood.

Establishing clear roles and responsibilities within the QA team and other related teams to promote collaboration between teams is important for developing our food delivery app. This is key as it will help us to facilitate the identification and resolution of quality issues. Another thing we will do is to conduct test planning where we will develop a comprehensive test plan outlining the scope, approach, resources, schedule, testing types. These types of testing include functional, performance, security and entry/exit criteria. Also, we will define specific quality objectives and criteria for acceptance.

To add on that, we will conduct requirements review of the requirements documentation to ensure they are clear, complete, and testable. Also, we will involve stakeholders in the review process to gather feedback and identify ambiguities. Moreover, we will create detailed test cases and test scripts based on requirements and use cases, covering positive and negative scenarios, edge cases, and user flows. Also, we will prioritize test cases based on risk assessment and critical functionality. To add on that, we will configure testing environments that mimic production settings, including devices, operating systems, and browsers. Also, we will ensure that all necessary tools and resources are available for testing.

We will then do testing execution where we will perform various types of testing to ensure we have created the right product for our stakeholders. These tests include unit testing which would be conducted by developers to verify individual components. Another test would be integration testing where we would assess the interactions between integrated components. Also, we would conduct system testing, where we would evaluate the complete system against requirements. Moreover, User Acceptance Testing (UAT) is another test we would conduct where we will be validating the app with end users to confirm it meets their needs. To add on that, to assess the performance, we would do a performance test for testing for load, stress, and scalability.

Defect Tracking and Reporting would be another strategy we would use the defect tracking tool to log, track, and prioritize defects. We will also conduct regular status meetings to discuss testing progress, defects found, and any necessary corrective actions. Another strategy we will use is to Implement regression testing following bug fixes or new feature development to ensure existing functionalities remain unaffected. Also, we will automate regression tests where possible to improve efficiency.

Quality Metrics and Reporting: Define and collect key performance indicators (KPIs) related to quality (e.g., defect density, test coverage, test pass rate). Also, we will generate regular quality reports to communicate findings, test results, and overall project quality status to stakeholders. Moreover, we will provide training and continuous improvement for QA team members on best practices, tools, and techniques. After that we will conduct post-project reviews to identify lessons learned and areas for improvement in the QA process.

Lastly, we will compile a comprehensive quality assurance plan which will include separate documents detailing the following aspects:

- Quality Assurance Plan: Outlining the QA strategy, objectives, roles, and responsibilities.
- Test Plan: Detailing specific testing activities, test cases, environments, and schedules.
- Defect Management Plan: Specifying processes for defect identification, tracking, and resolution.
- Test Case Documentation: Listing all test cases with expected results and execution steps.
- Test Metrics Report: Providing statistical insights into testing progress and quality.

Change management plan

To manage changes, we have a configuration management document that outlines how we will manage changes to the software development life cycle. It ensures that changes are tracked, verified, and implemented effectively without negatively impacting the software's functionality, performance, or stability.

This plan applies to:

- Software Code: All changes to source code, libraries, modules, etc.
- Documentation: Changes to project documentation, user guides, specifications, etc.
- Configuration Items: Any hardware, software, or related resources that are part of the project.

The process will be as follows

- Change Identification. A change request is initiated by a stakeholder (developer, engineer, client, etc.) whenever a need for change is identified. A form will be used to capture the reason for the change, description, impact analysis, and proposed solution.

- Change Log: A system will be in place to log the change for monitoring

- Change evaluation: The project manager and designer will evaluate the change request. Not all changes can be implemented without any impact to timeline or budget. The PM will evaluate the impact if the change

- Approval or Rejection: The PM will then choose to approve or reject the proposal.

- Implementation: If the PM approves the change, the developers will then implement the system. All changes will be tracked

- Testing and Validation: The change will undergo unit testing to ensure that it functions as intended without introducing new defects. The change will also be reviewed by the stakeholders to ensure that it meets the business requirements and does not negatively impact user experience.

- Documentation Update: All affected documentation, such as user manuals, API documentation, and configuration files, will be updated to reflect the changes made during the implementation.

This change management plan ensures that changes to the software development process are handled systematically, with clear procedures for logging, evaluating, approving and tracking changes. Establishing a well-defined change management process, ensures that our project teams can mitigate risks, ensure quality, and keep the development process on track.