Minute Burger

Version 1

Version	Date	Author(s)	Description of Change
1.0	2025/ /		
1.1	2025/ /		
1.2	2025/ /		
1.3	2025/ /		
1.4	2025/ /		

Name	Role	Email
Abellanosa Carl	Developer	2201104567@student.buksu.edu.ph
Lagare Dave	ProjectLeader/Documentation	@student.buksu.edu.ph
Adaro Joshua	UI/UX Designer	2201102105@student.buksu.edu.ph

Table of Contents

Document Overview

Scope:

This documentation outlines the features, technical framework, design structure, and deployment strategy of the Minute Burger Inventory System, a mobile system to help branches manage their inventory efficiently.

Out-of-Scope:

This document does not include marketing, post-release enhancements, or long-term maintenance strategies.

Audience:

This document is intended for the following readers involved in the development, deployment, and use of the Minute Burger Inventory System:

Developers – Responsible for building and maintaining the system, both mobile and web platforms.

UI/UX Designers – Ensure the system is user-friendly, accessible, and visually aligned with Minute Burger branding.

Tester – Verify the system's functionality, performance, and reliability through structured testing.

Project Stakeholders – Includes Minute Burger owners, branch supervisors, and decision-makers who require insights into the

system’s features and benefits.

End Users (Store Staff) – The primary users who will manage inventory, monitor stock levels, and generate reports.

Project Overview

Executive Summary:

The Minute Burger Inventory System is a mobile application designed to streamline inventory management across Minute Burger branches. It helps store staff and owners monitor stock levels, reduce losses from overstocking or shortages, and improve operational efficiency through real-time tracking and reporting.

Objectives:

To develop a minute burger mobile application that enables inventory system to improve the accuracy of the inventory management.

Business Goals

Improve accuracy in inventory tracking to minimize waste and stock shortages.

Provide a centralized system for easier management across multiple branches.

Enable faster decision-making through automated reporting.

Technical Goals

Develop a reliable, user-friendly system accessible via mobile devices.

Integrate with cloud storage for real-time data updates and backups.

Ensure secure user authentication and role-based access control.

High-Level Features:

Inventory Dashboard: Real-time overview of current stock levels.

Stock In/Out Logs: Record and track product entries and usage.

Low Stock Alerts: Automatic notifications when items fall below threshold levels.

Reports Generation: Auto-generated reports for stock usage, trends, and audit purposes.

Problem Statement:

Currently, Minute Burger outlets keep their inventories through manual forms - logbooks or spreadsheets - and these take a lot of time, are too human error-prone, and are not efficient in the monitoring of real-time stock levels. All these lead to inaccurate inventory counts, product shortages or over-stocking, delays in restocking, and lack of visibility across multiple branches. So not having a centralized system makes it hard for owners to track the operations with timely reports.

User Needs:

The Minute Burger Inventory System users the branch staffs and owners — would be able to count on the system for:

- Efficiency in updating inventory transactions.
- Real-time visibility of stock levels .
- System accessibility from an easy-to-use mobile.
- Alerts on low stock and critical shortages.

Market Analysis Summary:

While there are various inventory management apps available, many lack key features like real-time stock tracking and seamless integration. Minute Burger's Inventory System aims to address these gaps, providing a more tailored and efficient solution for fast food establishments.

Functional Specifications

Inventory Management

Allows users to add, update, and remove inventory items such as ingredients, packaging, and supplies. Each item has details such as quantity, expiration date, and cost.

- **User Interaction Flow:**
    - User logs in to the system.
    - User will see the Inventory on the main dashboard.
    - User can add, update, or remove items.
    - Confirmation is shown after an update.
- **Use Cases:**
    - **Primary Use Case:** An employee logs in and adds a new item to the inventory.
    - **Alternate Use Case:** The employee is unable to add an item because the necessary data is missing or incorrect**.**

Sales Report Generation

Allows the generation of reports based on sales data, inventory usage, and stock levels to assist in restocking decisions.

- **User Interaction Flow:**
    - User selects "Reports" from the menu.
    - User selects the date range for the report.
    - System generates a detailed report with sales and inventory data.
- **Use Cases:**
    - Primary Use Case: A owner generates a sales report for the week.
    - Alternate Use Case: The report fails to generate due to a server issue.

User Stories & Requirements

User Stories:

- As a owner, I want to view current stock levels so that I can order supplies before they run out.
- As a staff member, I want to update inventory when new stock arrives so that the system stays accurate.

Acceptance Criteria:

- The app must allow staff to add or update inventory items in real-time.

- The app must send notifications when stock levels fall below predefined thresholds.

Non-functional Requirements:

- **Performance**: The app should load within 10 seconds on most devices but depending on the Internet connection.
- **Scalability**: The system should be able to handle up to 30 transactions
- **Reliability**: The app must be available 99% of the time.

Technical Specifications

Architecture

Mobile Client: Android application developed using Java and Android Studio.

Backend:

- **Firebase Firestore**: Real-time NoSQL database for storing and retrieving inventory data.
- **Firebase Authentication**: Manages user registration and login using email and password.

APIs:

- **iTextPdf**: Utilized for generating downloadable and printable PDF reports from inventory or sales data.
- **Cloudinary:** Used for the Images of the inventory

Platform-Specific Considerations

Android Support:

- Minimum supported version: Android 8.0 (API level 26).
- Follows Material Design principles to ensure consistent UI/UX across Android devices.
- Optimized for both smartphones and tablets used in fast-food business operations.

Data Management

- **Data Flow**:
    - User action → Mobile client → Firebase → Real-time database update → UI reflects changes.
- **Database Schema**:
    - **Inventory Table**: ItemID, Name, Quantity, ExpirationDate, Cost.
- **API Endpoints**:
    - **POST /inventory**: Add or update inventory item.
    - **GET /inventory/{item_id}**: Retrieve details for a specific inventory item.

Security & Privacy

- **Authentication**: Firebase Authentication (email & password).
- **Authorization**: Role-based access control (owner, Staff).
- **Encryption**: HTTPS for secure data transmission; Firebase's built-in security for data storage.

Third-Party Integration

Firebase Cloud Messaging (FCM):
Sends real-time push notifications when inventory thresholds are reached or updates occur.

iTextPdf:
Generates PDF-based inventory or sales reports for printing or digital archiving.

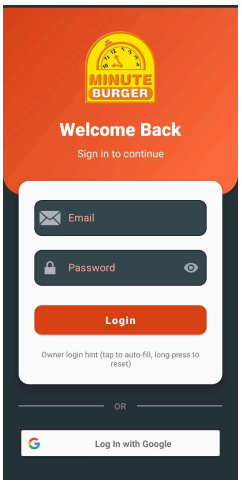
Cloudinary:

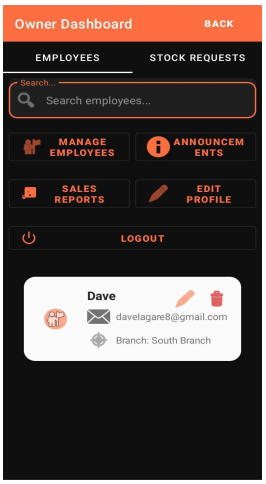
Cloudinary is used to upload and manage images in the Minute Burger Inventory System. This includes product photos and other relevant images that need to be associated with inventory items.

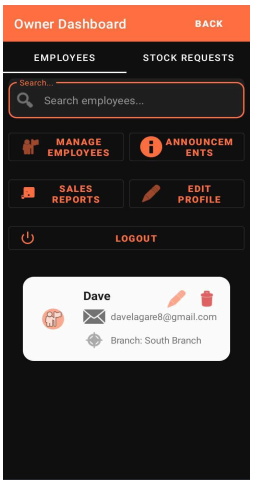
UI/UX Design Specifications]

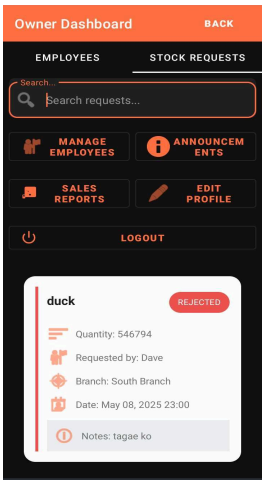
Wireframes

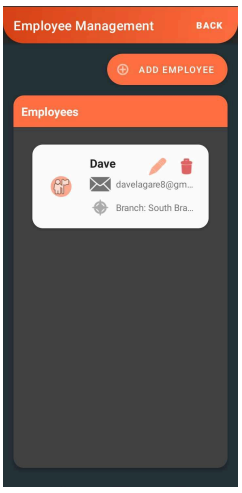
High-Fidelity Designs:

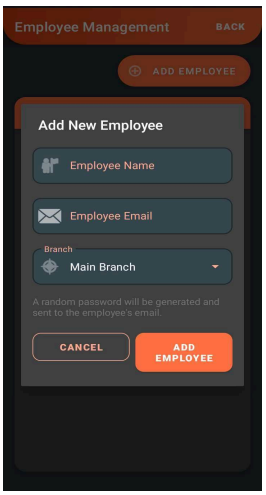


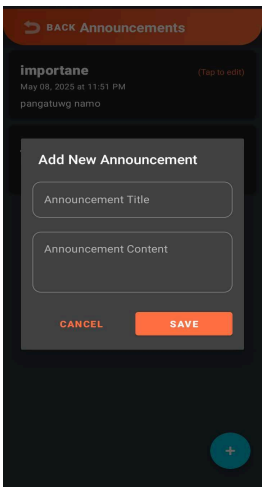


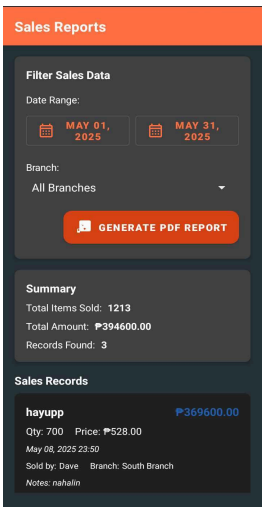


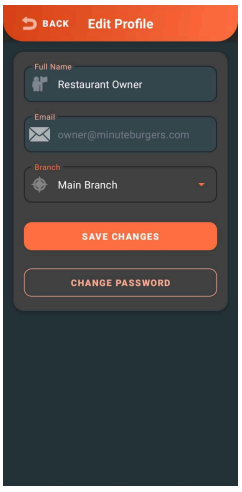


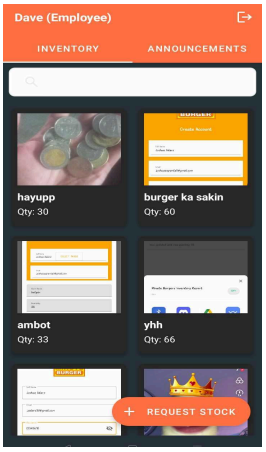


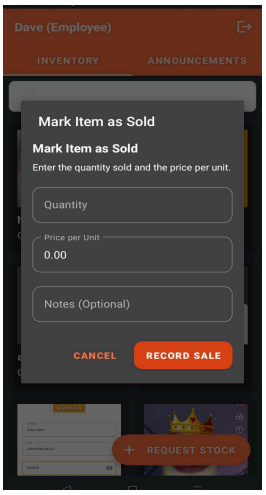






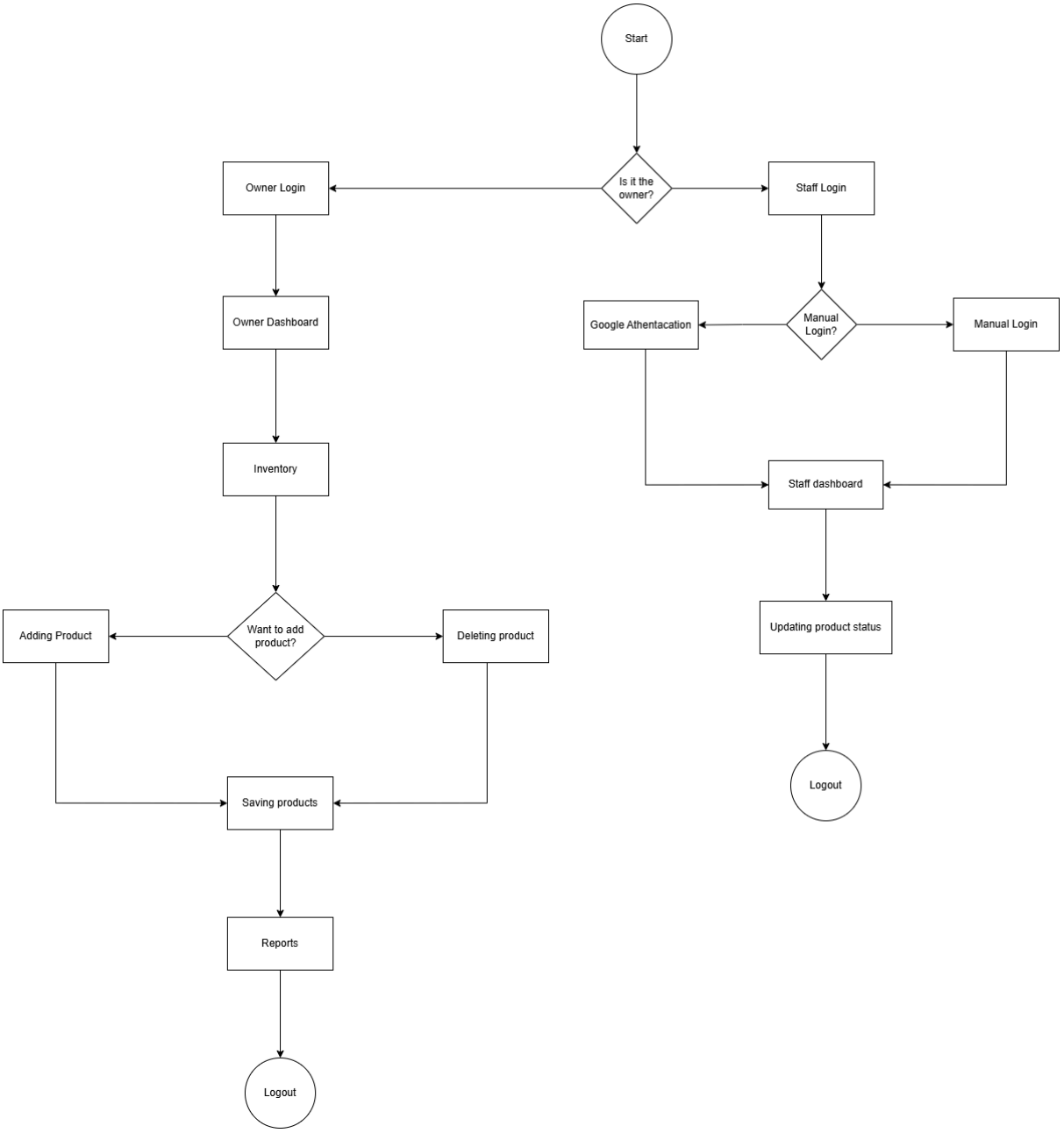






Navigation & Flow

User Flow Diagrams:



Accessibility Guidelines

The Minute Burger Inventory System is made to be easy for everyone to use, including people with vision problems, physical disabilities, or color blindness. We followed basic accessibility standards (WCAG) to make the app more inclusive.

Accessibility Features:

- **Clear Contrast**
  Text is easy to read with strong contrast between text and background colors.

- **Screen Reader Friendly**
  Icons and images have descriptions so screen readers can read them out loud.

- **Mobile-Friendly Design**
  The app works well on the android's screen

- **Consistent**
  The app is consistent in all ways the texts, colors and icons.

Deployment Plan

Release Process

The app will be finalized and built using Android Studio. Below are the steps for releasing the application:

- **Final Build**: Once development is complete, the final version of the app will be exported from Android Studio.

- **Upload to the webpage we created**: The APK will be uploaded to the Google Play Store using our institution's or developer's account.

- **Internal Testing**: Before going live, the app will undergo internal testing through the peer to peer internal testing track.

Rollout Strategy

- **Week 1: Internal Testers**
  The app will be shared with a small group of internal testers. They will verify core features, report bugs, and provide usability feedback.

- **Week 2: Limited Beta Group**
  After resolving initial issues, the app will be released to a limited beta group. This phase allows for broader testing, simulating real-world usage, and identifying any remaining bugs or performance concerns.

- **Week 3: Public Release**
  Once all major issues are resolved and performance is stable, the app will be officially launched to the public. Users will be able to download and install the app via webpage that we created/

Appendices

Glossary

APK (Android Package Kit) – The file format used by Android to distribute and install applications.
Firebase – A backend platform by Google that provides services like real-time databases, authentication, and cloud messaging.
Firestore – A NoSQL real-time cloud database used to store and sync data in Firebase.
Material Design – A design system developed by Google that outlines UI/UX standards for consistent and intuitive Android app interfaces.
iTextPDF – A library used to generate and manipulate PDF documents in Java-based applications.

References & Resources

Related Documentation

Firebase Implementation Guide /https://firebase.google.com/docs

Android Development Checklist/
https://developer.android.com/privacy-and-security/about

Cloudinary Documentation/ https://cloudinary.com/documentation

iText pdf Documentaion/
https://itextpdf.com/resources/api-documentation