PULSEPAY ZIMRA FISCALIZATION MOBILE APPLICATION

By

Collins Kurai Chimbganda

H210254G

HIT400 Capstone Project Submitted in Partial Fulfillment of the

Requirements of the Degree of

Bachelor of Technology

In

Software Engineering

In the

School of Information Sciences and Technology

Harare Institute of Technology

Zimbabwe

Supervisor

Mr. Mukosera

Month/Year

June 2025

ABSTRACT

This project presents the design and development of Pulse Pay, a fiscalized mobile point of sale system integrated with intelligent receipt anomaly detected mechanisms. The core of this project lies on the objective of streamlining tax-compliant transaction processing, at the same time enhancing accountability and fraud detection in real time. Through the fiscalization module, all receipts and credit notes generated by the system are encrypted, transmitted securely to the ZIMRA servers and digitally signed to ensure that there is authenticity and traceability, which are the main talking points of the concept of fiscalization.

The suggested system has these stages:

- Invoice generation
- Data Encryption and digital signature generation
- Payload submission and response pulling
- Fiscal QR Code generation
- Receipt Anomaly detection

The system transforms the users' invoice data into the required JSON payload format. Creates receipt string with a summary of the receipts taxes and invoice data from which a digital signature and hash are generated using RSA keys. The signature and hash are sent along with the receipt JSON payload to ZIMRA. An md5 hash version of the generated signature is then used to come up with a URL used so invoice validation and invoice QR Code generated. Anomaly detection measures are put on the system to detected erroneous receipts and potential cases of fraud. This system has the potential to be widely adopted by many tax payers in various working environment because of its simple usability, diversity and its ability to do some useful tax reporting. The solution empowers businesses to maintain compliance with national tax regulations and provides tax authorities with greater visibility into transaction data. Above all this innovation demonstrates the transformative potential of mobile technologies in modern tax administration and digital commerce

PREFACE

The growing need for transparency, accountability, and efficiency in financial transactions has driven the advancement of fiscal technologies across both public and private sectors. In many developing countries, challenges such as tax evasion, fraudulent receipts, and underreporting of sales continue to undermine revenue collection efforts. This project was initiated in response to such challenges, with the aim of developing a mobile-based solution that not only ensures fiscal compliance but also introduces intelligent mechanisms to detect irregularities in transaction records.

This document presents the journey of designing and implementing a fiscalized mobile application that integrates real-time receipt generation, digital tax validation, and anomaly detection powered by machine learning. The application serves both businesses and tax authorities by automating the fiscalization process and enhancing oversight through automated data analysis.

ACKNOWLEDGEMENT

To everyone who helped me finish this project successfully, I would like to extend my sincere gratitude and appreciation. I want to start by giving thanks to the Lord Almighty, whose grace and presence have been with me from the beginning of this program till now, when I am reaching a significant milestone. I want to express my gratitude to Mr. Mukosera, my supervisor, for his helpful advice, knowledge, and unwavering support over the entire study process. This project has been much influenced and improved by his perceptive comments, helpful critiques, and support. Additionally, I am incredibly appreciative of the Software Engineering department's faculty members, whose guidance and instruction have given me a solid background in my field of study. Their enthusiasm for learning and commitment to teaching have been incredibly motivating. I want to express my sincere gratitude to my mother for supporting me during the trip. For their constant encouragement and faith in me, I am also appreciative of my family and friends. I would like to express my gratitude to the study participants, whose readiness to share their knowledge and perspectives has greatly influenced the research results. Their input has been crucial in determining how this dissertation has turned out.

Finally, I would want to sincerely thank everyone who has indirectly supported this research by their publications, academic work, and earlier research. The theoretical foundation and technique of this dissertation have been greatly influenced by their contributions. Finally, I would want to express my sincere gratitude to everyone who helped to complete this project, no matter how modest their contribution was. Your encouragement, advice, and support have been priceless, and I sincerely appreciate the chance to have embarked on this research adventure with your steadfast assistance.

This is to certify that HIT 400 Project entitled "Pulse Pay**" has** been completed by **Collins K Chimbganda** (H210254G) for partial fulfilment of the requirements for the award of **Bachelor of Technology** degree in **Software Engineering**. This work is carried out by **him** under my supervision and has not been submitted earlier for the award of any other degree or diploma in any university to the best of my knowledge.

Your Supervisor Name	**Approved/Not Approved**
Project Supervisor	Project Coordinator
Signature: ……	**Signature: ……**
Date: ……	**Date: ……**

Certificate of Declaration

This is to certify that work entitled "Pule Pay" is *submitted in partial fulfillment of the requirements for the award of Bachelor of Technology (Hons) in Software Engineering, Harare Institute of Technology. It is further certified that no part of research has been submitted to any university for the award of any other degree.*

(Supervisor)	Signature……..	Date…….
(Mentor)	Signature……..	Date……
(Chairman)	Signature……..	Date……..

Project Documentation Marking Guide

ITEM	TOTAL MARK /%	ACQUIRED/%
PRESENTATION- Format-Times Roman 12 for ordinary text, Main headings Times Roman 14, spacing 1.5. Chapters and sub-chapters, tables and diagrams should be numbered. Document should be in report form. Range of document pages. Between 50 and 100.Work should be clear and neat	**5**	
Pre-Chapter Section Abstract, Preface, Acknowledgements, Dedication & Declaration	**5**	
Chapter One-Introduction Background, Problem Statement, Objectives – smart, clearly measurable from your system. Always start with a TO… Hypothesis, Justification, Proposed Tools Feasibility study: Technical, Economic & Operational Project plan –Time plan, Gantt chart	**10**	
Chapter Two-Literature Review Introduction, Related work & Conclusion	**10**	
Chapter Three –Analysis Information Gathering Tools, Description of system Data analysis –Using UML context diagrams, DFD of existing system Evaluation of Alternatives Systems, Functional Analysis of Proposed System-Functional and Non-functional Requirements, User Case Diagrams	**15**	
Chapter Four –Design Systems Diagrams –Using UML Context diagrams, DFD, Activity diagrams Architectural Design-hardware, networking Database Design –ER diagrams, Normalized Databases Program Design-Class diagrams, Sequence diagrams, Package diagrams, Pseudo code Interface Design-Screenshots of user interface	**20**	
Chapter Five-Implementation & Testing Pseudo code of major modules /Sample of real code can be written here Software Testing-Unit, Module, Integration, System, Database & Acceptance	**20**	
Chapter Six –Conclusions and Recommendations Results and summary, Recommendations & Future Works	**10**	
Bibliography –Proper numbering should be used Appendices –templates of data collection tools, user manual of the working system, sample code, research papers	**5**	
	100	**/100**

Table of Contents

List of Figures

List of tables

Chapter One: Introduction

1.1 Background

In the digital age, fiscalization has become a fundamental part in ensuring compliance with tax regulations. Traditionally, businesses in Zimbabwe have relied heavily on Electronic Cash Registers (ECRs) and fiscalized printers for issuing fiscalized receipts and invoices. However, these are now face significant challenges with the introduction of ZIMRA's FDMS (Fiscal Data Management System), particularly in remote and low-network areas, where connectivity problems are disrupting the fiscalization processes. Another challenge is arising on implementations where a fiscal system work separately from the accounting packages. This implementation lead to a lot of errors in invoice data extraction and tax compliance in general.

With the growth of mobile technology, there is a unique opportunity to overcome these challenges. This project proposes a mobile-based fiscalization application that allows users to generate fiscalized receipts and invoices even in areas with poor network connectivity. Beyond receipt generation, the mobile app also offers detailed tax reporting, receipt anomaly detection, and advanced data analytics for sales and tax predictions — equipping businesses with intelligent tools for compliance and operational insights. This is an implementation that will accommodated individuals and businesses in various industries and of various sizes.

1.2 Problem Statement

As from January 2024, TARMS and FDMS from ZIMRA replaced the already existing ZIMRA tax collection systems with a new one that required the upgrading of already existing tax collection devices and systems to match up with the current implementation. Interaction with FDMS is heavily dependent on stable internet connection to perform the fiscalization tasks and these upgrades on devices that are almost obsolete are posing agonizing network challenges and unexpected device failures. In most cases receipts are taking too much time to be processed and reflect as received on the ZIMRA servers. Implementations that have a fiscal system working separately from the accounting package are mostly facing challenges in due to template changes

that lead to incorrect reading and extraction of invoice data from the PDF invoices. These challenges are not only affecting the tax collection authorities but also inconveniences businesses and customers alike. Therefore, there is an unforeseen need for a mobile fiscalization solution that is not only cheap but can operate reliably given the nation's current conditions while also offering intelligent reporting and anomaly detection features.

1.3 Objectives

The primary objectives of this project are:

- Achieve 100% successful connection to ZIMRA's API gateway and process at least 98% of fiscalization requests
- Ensure 100% of generated invoices are encrypted and digitally signed using an approved cryptographic method, with validation logs showing zero unsigned invoices during monthly audits.
- Implement an anomaly detection module that identifies fraudulent or erroneous invoices with a false positive rate
- Develop a reporting dashboard that provides users with real-time insights covering 100% of submitted invoices, including return summaries, and compliance status, with reports available monthly and on demand.

All objectives are specific, measurable, achievable, relevant, and time-bound (SMART), with the final mobile application being a tangible deliverable evaluated against these goals.

1.4 Hypothesis

The proposed mobile fiscalization application will improve the reliability and accessibility of fiscalized receipt generation in low-network areas, enhance tax compliance, and provide users with actionable sales and tax insights through advanced reporting and predictive analytics.

1.5 Justification

This project addresses a critical gap in the market by targeting users who are currently underserved by traditional fiscalization methods. Mobile devices are more affordable and widespread than other fiscalization devices, and enabling fiscalization on such platforms will:

- Enhance compliance for small to medium-sized businesses.
- Reduce operational costs associated with maintaining traditional ECRs and Printers.
- Provide businesses with critical insights for decision-making through built-in analytics.
- Contribute to national revenue collection by broadening the base of compliant taxpayers

1.6 Proposed Tools

The following tools and technologies will be used in the project:

Tool / Technology	Purpose
RSA Certificates	• To digitally sign fiscal receipts and securely encrypt communication with ZIMRA servers. RSA certificates ensure that data integrity, authenticity, and confidentiality are maintained throughout the fiscalization process.
Flutter	• Front-end mobile application development framework • Will be used from the application's interface development
Dart	• Application programming language used by the Flutter framework • Will be used program the application's interface and part of the backend
SQLite Database	• A lightweight database engine that will be used by the mobile application • Will be used to create the system's database and its

	related tables • Will support SQL queries for data input and extraction
REST APIs (ZIMRA fiscalization)	• Provides the official interface for submitting fiscal receipts, receiving validation signatures, and handling of all fiscal operations. The system interacts with this API to remain legally compliant with tax regulations.
Kotlin modules	• Will be used in the application's backend • It will be used to generate the receipt signature using the locally stored certificates • Will be used to generate invoice MD-5 Hash
Machine Learning (Python)	• Used to implements algorithms to identify suspicious or irregular receipt patterns
PDF libraries	• Used to generate professional invoice documents with embedded fiscal data • PDF libraries allow formatting, styling, and saving receipts for email or printing purposes.
QR code libraries	• Generates QR codes containing ZIMRA-approved fiscal data • These codes are required by law and must be included on all fiscal receipts.

Table 1: Proposed tools

1.7 Feasibility Study

1.7.1 Technical Feasibility

The proposed application can be developed using widely adopted mobile technologies (Flutter, Dart, python and SQLite). Fiscalization protocols can be integrated using available API standards. Data analytics and anomaly detection can be implemented using lightweight models optimized for

mobile devices. Therefore, the project has been deemed as technically feasible as these technologies are readily available and are at disposal to any developer

1.7.2 Economic Feasibility

Compared to the high costs of setting up and maintaining ECR systems, a mobile application significantly reduces costs for businesses. Development and maintenance costs are minimal, here is why the project is regarded as economically feasible:

1. Low development and maintenance costs – the use of open source technologies like Flutter and python ensures that there is zero to little licensing costs involved in the project. This proves that a small development team can develop and maintain the system , especially when leveraging cross platform tools that minimize duplication effort
2. Potential for monetization – the application can be used to generate income through subscription plans, usage bases pricing, transaction commissions or enterprise licensing. Integration with government compliance programs could also lead to public-private partnerships or subsidies that reduce costs for users.

1.7.3 Operational Feasibility

Users are already familiar with mobile apps, reducing training requirements. The app will feature a user-friendly interface designed for simplicity. Businesses will be able to continue operations even during network outages, and the app's reporting features will offer additional operational benefits. Points that show that there is operational feasibility are as follows:

1. User friend mobile interface – the mobile application is designed aiming at simplicity and usability. This ensures that both technical and non-technical system users will easily grasps the system's work flows, making sure that they can easily navigate features such as receipt generation, fiscalization, and report viewing without extensive training.
2. Alignment with existing business processes - The app integrates seamlessly into daily sales operations. It automates fiscalization in the background as users issue receipts, eliminating the need for additional steps or new workflows. This reduces resistance to adoption and ensures continuity in business operations.

3. 24/7 availability and real time processing - Since the solution is mobile and cloud-based, users can access it anytime, from anywhere, as long as they have internet connectivity. Fiscalization and anomaly detection run in real-time or near-real-time, supporting responsive and continuous operations.
4. Scalability Across Different Business Sizes - Whether used by small shops or larger enterprises, the system adapts to varying transaction volumes and user roles. It supports multi-user environments, branch-level reporting, and centralized monitoring, making it operable in diverse business settings.

1.8 Project Plan

1.8.1 Time Plan

The project will follow a four-phase agile approach with each phase having its own target weeks and the respective activities that will be looked into:

Phase	Duration	Activities
Requirements Gathering	2 Weeks	Understand fiscalization regulations, finalize system requirements. Get insights from taxpayers since they are the system users and tax regulators as they will determine how the system is going to be developed.
System Design	2 Weeks	User interface and User experience design, Database modelling , system logic design
Development	6 Weeks	System frontend and backend development, fiscal API gateway integration.
Testing and Deployment	2 Weeks	Beta testing , bug fixing and final application deployment

Table 2: Project plan

Total Duration: **12 Weeks**

1.8.2 Gantt Chart

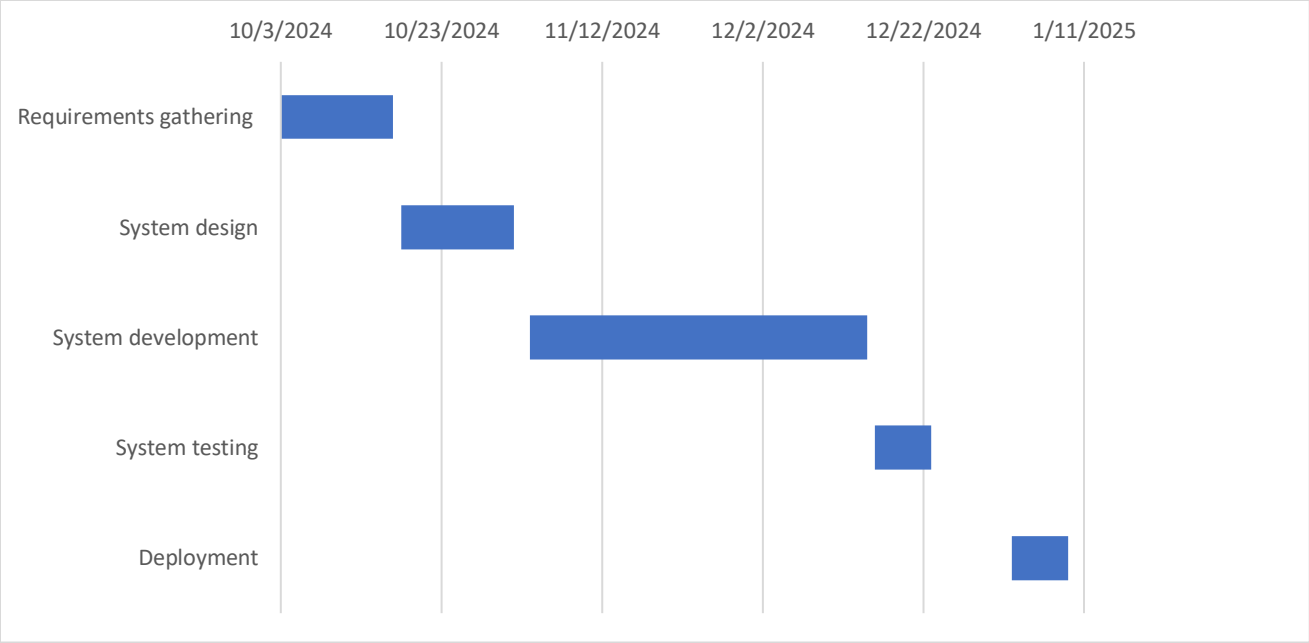


Figure 1: Gantt chart

Chapter Two: Literature Review

2.1 Introduction

Fiscalization in business operations is a critical step toward enhancing transparency, improving tax collection, and minimizing fraud. Over the years, various technologies have been adopted globally to support fiscalization, ranging from traditional Electronic Cash Registers (ECRs) to modern cloud-based Point of Sale (POS) systems.

However, with the challenges of connectivity, especially in developing regions, there is growing interest in mobile-based fiscal solutions. This chapter explores existing fiscalization technologies, highlights their strengths and weaknesses, and reviews relevant research and projects related to mobile fiscalization, anomaly detection, and sales data analytics.

2.2 Related Work

The term fiscalization refers to the implementation of systems and processes that ensure accurate reporting and monitoring of financial transactions with the main goal set to ensure proper tax compliance. This tax compliance had been enforced through hardware based fiscal devices and has since evolved alongside technological advancements to include software based and hybrid based systems [1].

Combating tax evasion and enhancing revenue collection are the main objectives of fiscalization, especially in cash-heavy industries like retail and hospitality [2]. Fiscalization has been implemented in a variety of ways by different countries. For example, Sweden and Kenya have shifted to software-driven solutions that are integrated with their national tax systems, whereas others like Italy and Serbia have depended on fiscal cash registers (FCRs) [10].

Fiscalization Technologies and Standards

According to [5], fiscalization has historically necessitated the deployment of physical fiscal memory devices that safely held transaction data and made manipulation difficult. These device

were ranging from electronic cash registers, electronic signature devices to fiscalized printers that were being distributed by some companies like Eltrade. With recent development in fiscal and tax regulatory systems, these device have since become obsolete and seen to be causing a lot of challenges when it comes to being compatible with these modern day tax systems. More scalability and flexibility are provided by more recent systems, which make use of cloud computing, encryption, and real-time data transfer [3]. To advance the digitization of fiscal data, the OECD created the framework Audit File for Tax (SAF-T), which establishes a framework for the electronic interchange of trustworthy accounting data [1]. In countries like Zimbabwe there has been a rise in the use of virtual fiscalization systems that are working alongside accounting packages to ensure that tax collectors are able to issue out fiscal invoices.

Key technological components of modern fiscal systems include:

- Encryption and digital signatures: Ensuring data integrity and authenticity [4].
- QR codes: Common in countries like Zimbabwe and Hungary, these provide verifiable links to tax-registered invoices [9] [7].
- APIs for real-time reporting: As seen in Croatia's fiscalization model, real-time APIs connect point-of-sale systems directly to tax authorities [8].

Fiscalization in mobile applications

Fiscalization has spread into the mobile application area as a result of the growing usage of smartphones and mobile point-of-sale (mPOS) systems. Businesses in the informal sector and small and medium-sized enterprises (SMEs) can join formal economies with lower overheads because to mobile fiscalization [7]. The field has rapidly advanced in recent years as we seen improvements and innovations ensuring that compliance can be ensured at ease with the use of mobile devices.

Mobile-compatible fiscalization, which has been adopted by nations like Zimbabwe, enables retailers to issue and fiscalize invoices through mobile applications that connect to national tax authority systems through secure API calls [9]. In some countries like Kenya and Nigeria, real-time mobile tax compliance has been in use for some years now. Kenya's iTax and Nigeria's eTax leverage mobile platforms to streamline VAT reporting and enhance compliance in the countries

[10]. Many mobile fiscal apps in different countries now include an automated QR code generation feature, which contains fiscal data like tax identification, invoice totals and digital signatures, ensuring easy verification by customers and authorities.

Despite the promise in mobile fiscalization there are some challenges that remain and can be foreseen with the implementation of such systems. A main issue lies in device and network limitations. In regions with limited internet infrastructure, real time fiscalization be difficult to implement according to [6]. Modern mobile applications also required mobile devices with better architecture meaning that acquiring a compatible mobile device could be unnecessarily expensive to some individuals or startups. In some regions it has been seen that ensuring mobile solutions comply with diverse fiscal regulations across jurisdictions requires an adaptive design [5].

Conversely, mobile fiscalization offers transformative opportunities for **financial inclusion** and **enhanced tax collection**, particularly in developing economies.

The development of fiscalization is indicative of more general patterns in tax compliance and digital governance. Modern fiscalization makes greater use of cloud services, APIs, and mobile technologies than did traditional systems, which were dependent on physical hardware. Particularly in developing nations, mobile fiscalization has a great deal of promise for expanding formal economic involvement to SMEs and unofficial enterprises. Future studies can concentrate on investigating machine learning techniques for automated tax fraud detection and improving security frameworks for mobile fiscal systems.

Receipt Anomaly detection

Receipt anomaly detection involves identifying transactions that may indicate fraud, errors, or compliance issues. Research has shown that machine learning models such as Decision Trees, Random Forests, and Neural Networks are effective at detecting anomalies in financial data.

- **Example:** In 2022, a study by Li et al[12], demonstrated the use of supervised learning models to detect anomalies in POS data with an accuracy exceeding 90%.

In the proposed system, lightweight models suitable for mobile execution will be integrated to provide immediate anomaly feedback to users.

Electronic Cash Registers (ECRs)

Electronic Cash Registers were among the first tools adopted to enforce fiscalization laws. These devices are hardwired to comply with tax regulations and include features like fiscal memory to store sales data securely.

- **Strengths:** High security, government-approved compliance.
- **Weaknesses:** High cost, lack of mobility, dependency on stable network environments for real-time reporting.

In Zimbabwe, the fiscalization system using ECRs has faced challenges in remote areas due to unreliable network connections, resulting in delays in reporting and difficulties in issuing fiscalized receipts.

2.3 Conclusion

The literature reveals a shift from traditional fiscal devices toward mobile-based fiscalization solutions, driven by the need for flexibility, affordability, and reliability in low-network areas. While ECRs and fiscal printers have served their purpose, they no longer fully address the connectivity and cost challenges faced by small businesses, especially in rural areas.

Research into anomaly detection and sales analytics demonstrates that integrating intelligence into fiscalization tools provides significant added value for businesses beyond mere compliance.

Thus, the proposed mobile fiscalization app, which combines offline receipt generation, receipt anomaly detection, and data-driven sales analytics, addresses a clear and growing need. It leverages lessons learned from global best practices while tailoring the solution to local conditions, ensuring greater accessibility, usability, and business empowerment.

Chapter Three: Analysis

3.1 Information Gathering Tools

To ensure the development of a system that meets user needs and regulatory requirements, several information gathering techniques were employed:

1. **Interviews**

   Participants:

   - 3 small business owners
   - 2 tax consultants
   - 1 ZIMRA compliance officer

   Key Questions Asked:

   - What challenges do you face with current fiscal systems?
   - What features would you like to see in a mobile-based fiscal solution?
   - How do you currently track and report taxes?

   Key Findings:

   - Manual and desktop fiscal systems are slow and error-prone.
   - Users want mobile accessibility and real-time fiscalization.
   - There's concern over system compliance with ZIMRA standards.
   - Interest in features like automatic VAT calculation, receipt anomaly alerts, and PDF invoice generation.

   Result Summary:
   Stakeholders expressed high interest in a lightweight, mobile-first platform that reduces tax reporting burdens while ensuring legal compliance.

2. Questionnaires

Respondents: 25 business users across Masvingo and Zvishavane

Format: Paper Forms

Response Rate: 88% (22/25 respondents)

Key Questions & Aggregated Responses:

Question	Most Common Response
Are you satisfied with your current fiscal ECR device/system?	73% said "No"
Do you use a mobile application for fiscal purposes?	91% said "No"
Would anomaly detection help your business?	95% said "Yes"
Preferred features	PDF invoices, tax auto-calculation, ZIMRA compliance

Table 3 : Key questions

Result Summary:

There is strong demand for a mobile fiscal app that simplifies tax compliance. Respondents prioritized ease of use, anomaly detection, and offline functionality.

3. Document reviews

Reviewed Documents:

- ZIMRA Fiscalization Guidelines (2021)
- VAT Act of Zimbabwe (Section 64)
- User manuals of existing fiscal devices (e.g., FP300, FP600)

Findings:

- ZIMRA requires fiscalization to produce signed receipts with digital signatures and QR codes.
- Compliance must include secure certificate handling, signature generation, and data transmission to ZIMRA servers.
- Existing systems are either too technical or not user-friendly for small traders.

Result Summary:

Any new system must strictly align with legal fiscalization protocols, including secure certificate use and real-time or batch communication with tax authorities.

4. Observations

Locations Visited:

- 2 retail stores
- 1 pharmacy

What Was Observed:

- Time-consuming processes to generate receipts and monthly reports
- Frequent human errors during VAT entry
- Limited training provided by current vendors
- Low understanding of tax evasion safeguards

Result Summary:

Businesses lack intuitive systems that reduce manual steps. Observation confirmed the need for automation, training aids, and mobile-friendly interfaces.

3.2 Description of the Current System

The current system works with an electronic cash register that looks like more of a calculator that a fiscalization device. The device has an inbuilt fiscalization module and a printer that process the user's punched in codes to determine the sold products and the respective customers involved in the transaction.

3.2.1 System Components:

The cash register's system consists of the following components:

- Fiscal memory – it holds all of the fiscalization data used by the device. Such information includes the tax payer details, the services provided and the currency and taxation data.
- SIM Card module – the device works with a local sim card for data transmission purposes.
- Printer – Contains an inbuilt printer that prints out receipts per each transaction

3.2.2 Current Operation flow:

The use of the electronic cash registers follows the following pattern:

- Fiscal day opening:
  The device will not process any receipts without opening the fiscal day so that's is the first prompt users see when they first open the device. With this operation the device sends Open day requests to the ZIMRA servers
- Receipt generation:
  The receipts are generated on a terminal that required users to first select the desired service

by punching in the code and then putting the price and the quantity purchased

- Receipt printout:

  The device prints out the required QR Code fiscal receipt if the device is upgraded or receipt with a signature on the footer if the device is not upgraded to meet current fiscal requirements

3.2.3 Current System Limitations:

The Current system making use of the electronic cash registers has the following limitation:

- The devices face a lot of network issues resulting in unprocessed receipts and complete denial of the opening of a fiscal day which mean users will not be able to generate some receipts
- There is a lack of inventory management as the device does not keep track of stocks and other product related information.
- There is no data backup , if the fiscal memory card gets damaged there would be nowhere to get the data
- There is a lack of adequate reporting that shows users their periodic tax returns.

3.3 Data Analysis

3.3.1 Context Level DFD of Existing System

Existing System: Traditional ECR Fiscalization system

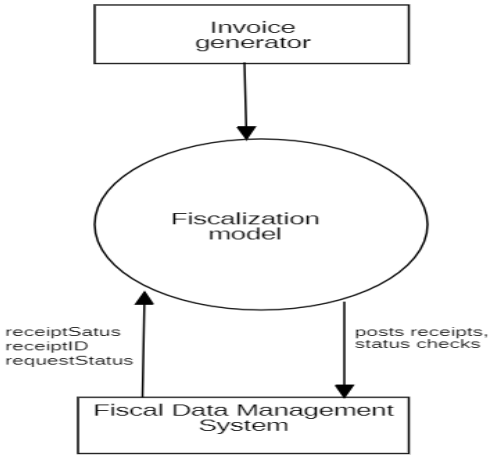


Figure 2: Level 0 DFD of existing system

Description:

- User inputs sale into the ECR.
- ECR generates receipt and attempts to transmit fiscal data immediately.
- If the network is down, data transmission fails, resulting in compliance issues.

3.3.2 Level 1 Data Flow Diagram (DFD) of Existing System

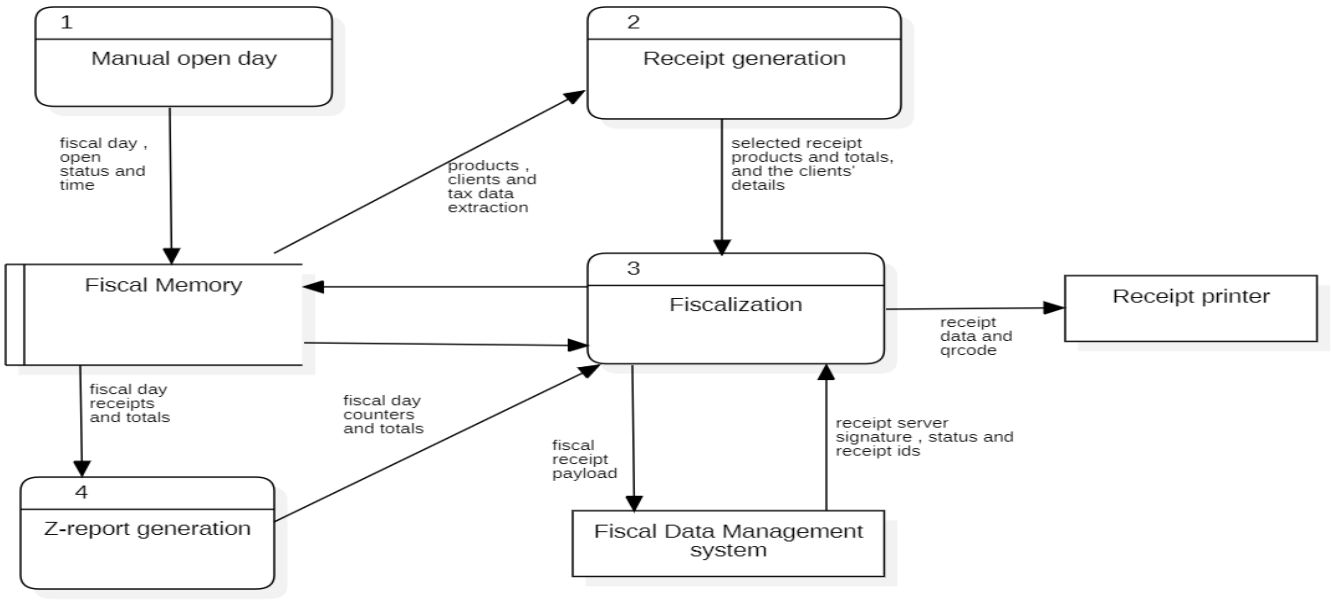


Figure 3: level 1 DFD of existing system

Issues Identified:

- Heavy reliance on immediate internet access which means the device will not work well in areas with poor network infrastructures leading to partial compliance of tax payers
- Risk of unsynchronized sales data.
- Users are not presented with a proper and easily understandable report of the amount of tax collected quarterly.
- High maintenance cost of ECR devices.
- Transaction reversal and cancellation is a complex and time consuming process.
- The device receipts cannot be issues to companies that required A4 fiscalized invoices

3.4 Evaluation of Alternative Systems

Before embarking on the development of the fiscalized mobile application with receipt anomaly detection, it was essential to explore and evaluate alternative systems currently available or in use for similar purposes. This evaluation provides a comparative understanding of existing solutions in terms of functionality, cost, scalability, compliance, and adaptability to local tax regulations.

1. **Continue with ECR system:**

   This proposes the intention of keeping up the use of the electronic cash registers

   Pro:

   - Approved by ZIMRA
   - Legacy system that has been in use for many years so taxpayer are affiliated to them more
   - Basic setup and hardware makes it simple to use for people with low technical abilities

   Cons:

   - There are high costs when it comes to system upgrade
   - There are no possible data backups do recover data in cases of device damage leading to damage of the fiscal memory that is found within the devices

- With current development in the nation's network infrastructure and requirements, the old devices network abilities are now poor as they use 3G network which is posing a lot of challenges when connecting and communicating with the fiscal data management system

2. Develop a cloud only solution

With advancements is technology and network infrastructure most of the modern systems are being launched and run on cloud platform to leverage on the pool of cloud computing benefits. Cloud computing is proving an easy way to develop, deploy and maintaining application so having a cloud based fiscalization solution was possible solution to the fiscalization problem.

Pros:

- It improves the developed system's accessibility. This will enable multi location access and allow the application to be launched on a number of devices.
- Proper backup of fiscal data is guaranteed as this is offered by the cloud service providers
- Vast amounts of storage and processing capabilities so the application with work swiftly.

Cons:

- Cloud solutions require constant internet connection to the able to make complete use of the systems hosted, therefore there impractical in low network zones and regions that have a poor network infrastructure
- There are lot of running costs associated with cloud based solutions are they follow a pay as you go model. So the user will have to constantly pay for the services and utilities provided

3. Mobile based fiscalization system(Proposed):

This proposes the development and implementation of mobile based fiscalization solution that integrates the fiscalization module with a point of sale system to leverage on the benefit of having such a system.

Pros:

- Cost effective
- Flexible
- Offline capabilities
- Smart features

Cons:

- Requires secure local storage
- devices

Conclusion:
The mobile-based fiscalization system offers the best balance between compliance, cost, flexibility, and innovation.

3.5 Functional Analysis of Proposed System

3.5.1 Functional Requirements

Functional Requirements	Description
Receipt and invoice generation	• The system shall allow users to generate a receipt for their issued service of purchased product • The system shall allow user to generate an A4 fiscal invoice with its fiscal signature
Offline functionality	• The system show allow fiscalization and invoices generation when there is no network
Tax reporting	• The system shall generated monthly tax return report for the client.
Anomaly detection	• The system shall periodically go through processed receipts and check for anomalies that could lead to fraud or invalid verification of

	invoice signature
User authentication	• The system shall authenticate and allow each verified user to access and use the system
QR code generation	• The system shall use the invoice meta-data to come up with a ZIMRA URL that will be used to generate the invoice QR code
Fiscalization operations	• The system shall allow the open of a fiscal day • The system shall allow checking of device configuration on the ZIMRA server • The system shall allow users to check for device status • The system shall allow uses to do fiscal close day operations • The system shall allow the users to submit the missing receipts.

Table 4: Functional requirements

3.5.2 Non-Functional Requirements

Non Functional Requirements	Description
Security	• The system shall encrypt fiscal receipts and stored user data using SHA-256 encryption. • The system shall restrict user access to the system • The system shall encrypt all receipt data in transit using SSL.
Usability	• The system shall be accessible on mobile devices and in a responsive manner • The system shall provide real-time feedback (e.g., loading spinners, error messages) for all user actions. • The system shall provide contextual help and tooltips for all major functions.

	• The system shall provide an intuitive mobile interface optimized for touchscreen interactions
Performance	• The system shall handle transaction data processing within 5 seconds • The system shall perform daily sales report generation in less than 5 seconds. • The system shall generate a fiscal invoice and QR code in under 4 seconds after transaction confirmation.
Reliability	• The system shall ensure that no duplicate receipts are generated for a single transaction. • The system shall queue and retry failed fiscal data submissions to tax servers. • The system shall perform data validation to prevent processing of corrupted input. • The system shall recover automatically from application crashes without data loss.
Maintainability	• The system shall be developed with modular architecture to allow easy updates and bug fixes. • The system shall isolate fiscal engine logic from UI components for easier upgrades. • The system shall include clear documentation for APIs, database schema, and core modules.
Scalability	• The system shall support on-boarding of up to 10,000 business users without architecture changes. • The system shall handle a 10x increase in transaction volume with no more than a 15% increase in response time. • The system shall allow dynamic addition of fiscal modules without affecting existing functionality.

Table 5: Non Functional Requirements

3.6 Use Case Diagrams

3.6.1 Main Use Case Diagram

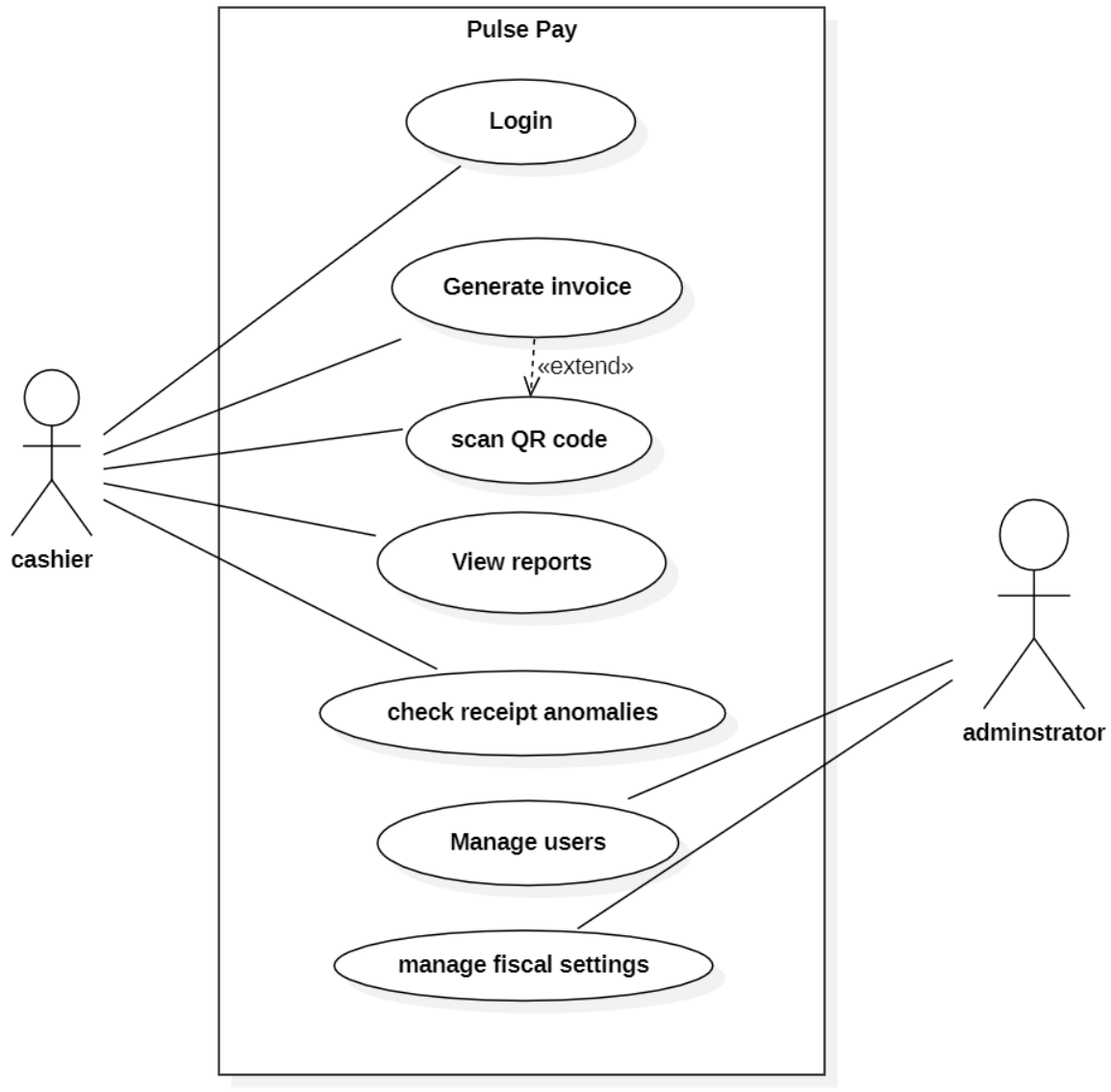


Figure 4: System use case diagram

Chapter 4: System Design and Implementation

4.1 Introduction

This chapter outlines the design and implementation of a fiscalized Point of Sale (POS) mobile application tailored to operate on POS machines. The system is engineered to generate and print fiscalized receipts directly upon sale completion, ensuring compliance with tax authority requirements. It integrates seamlessly with fiscal devices and communicates with tax authority servers for real-time fiscalization.

4.2 Proposed Solution

The proposed solution is a mobile POS application capable of processing transactions, generating fiscal receipts, and transmitting data to tax authority servers for validation. It is designed to work on POS machines, supporting essential functionalities like sales tracking, receipt printing, inventory management, and fiscal data reporting.

4.3 Solution Architecture

The solution architecture follows a client-server model:

- **Client Side:** Mobile POS application running on Android-based POS machines.
- **Server Side:** Cloud-based fiscalization server that interacts with tax authority systems (e.g., ZIMRA) to verify and approve transactions.
- **Communication:** Secure API endpoints using HTTPS with encryption protocols for secure data transmission

4.4 Systems Diagrams

4.4.1 UML-Activity Diagram

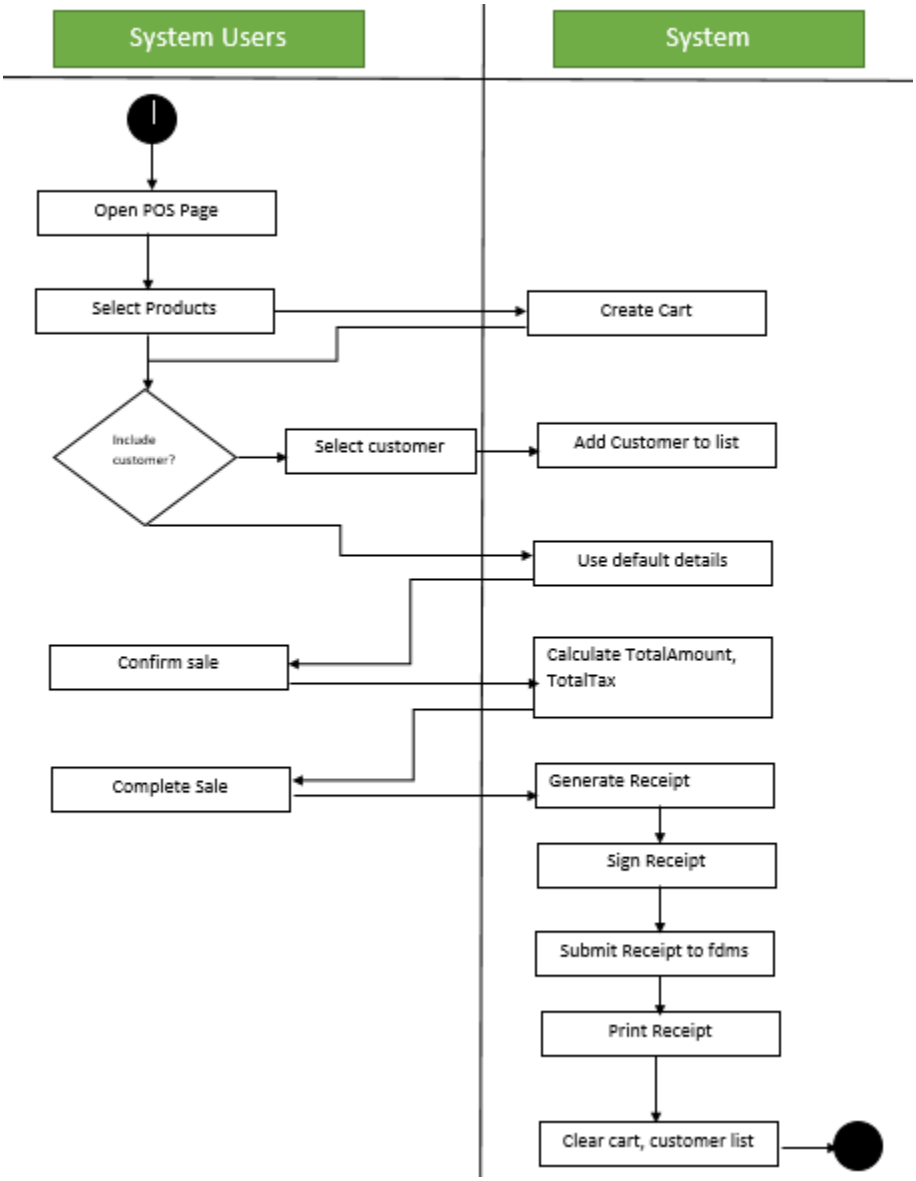


Figure 5:UML Activity Diagram

4.4.2 Level 0 DFD

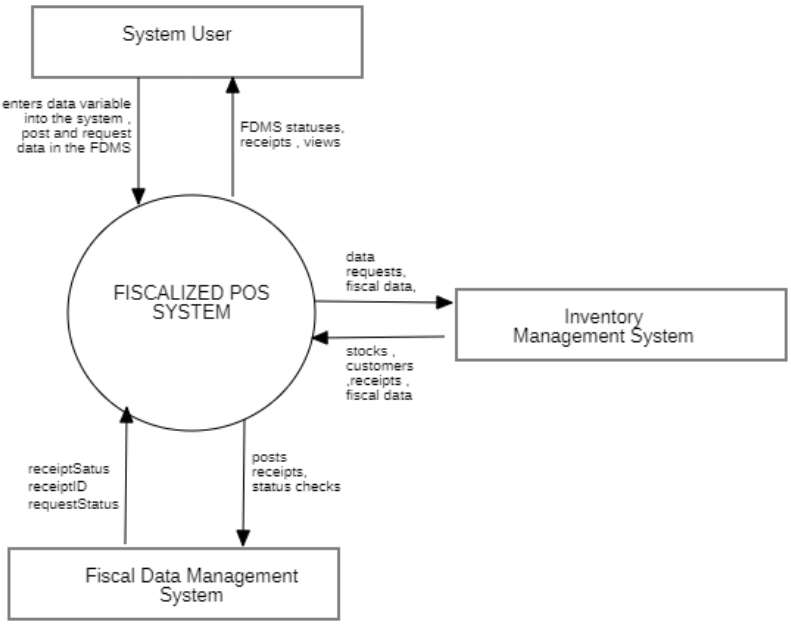


Figure 6: Level O DFD of proposed system

4.4.3 Level 1 DFD

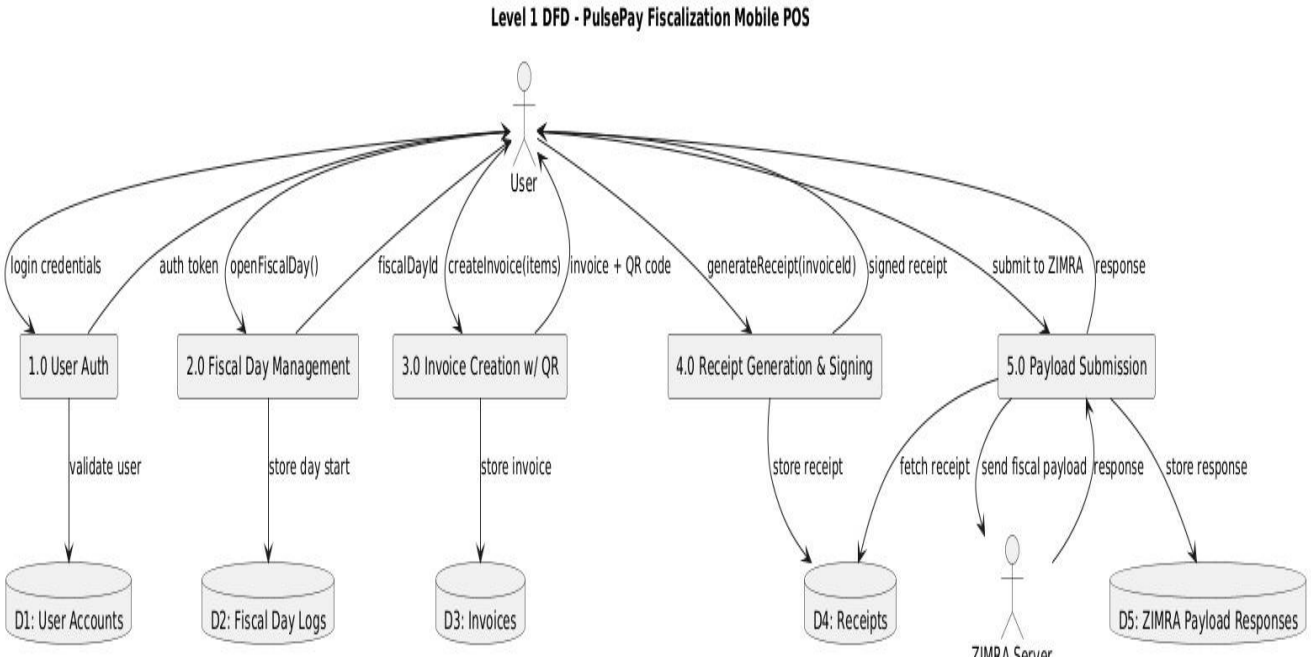


Figure 7 : Proposed system level1 DFD

4.4.4 Architectural Design-hardware

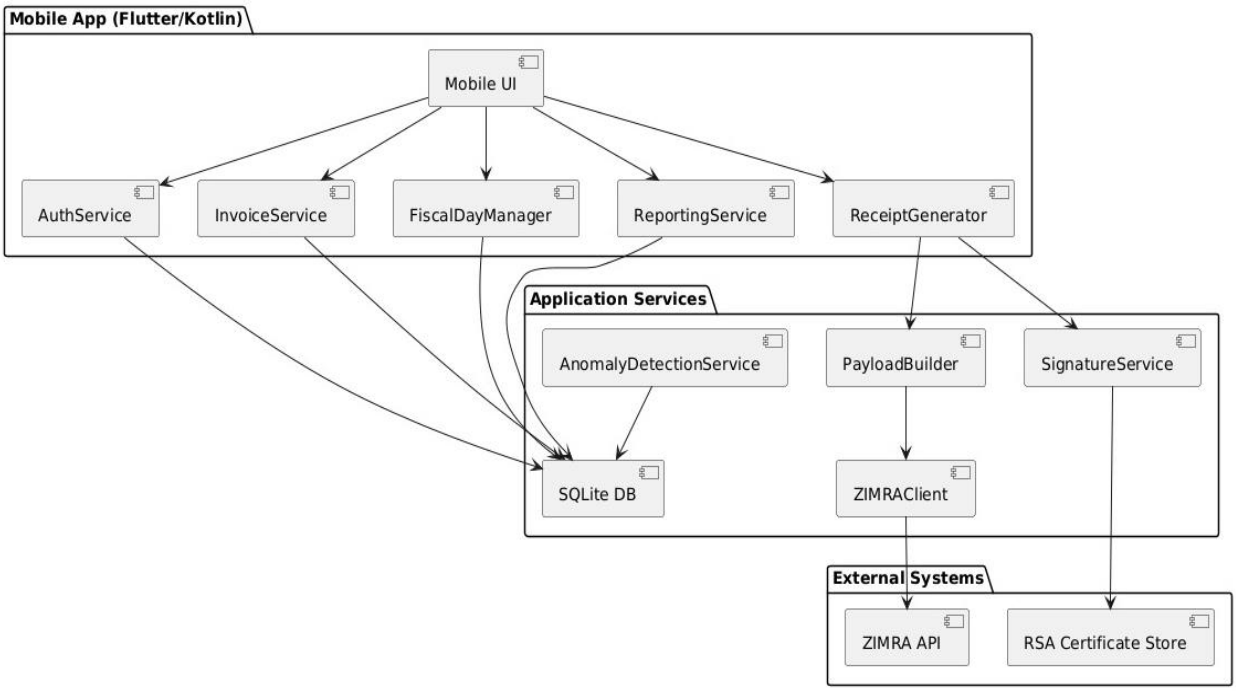


Figure 8: Architectural design, Hardware

4.5 Database Design

4.5.1 E-R Diagram

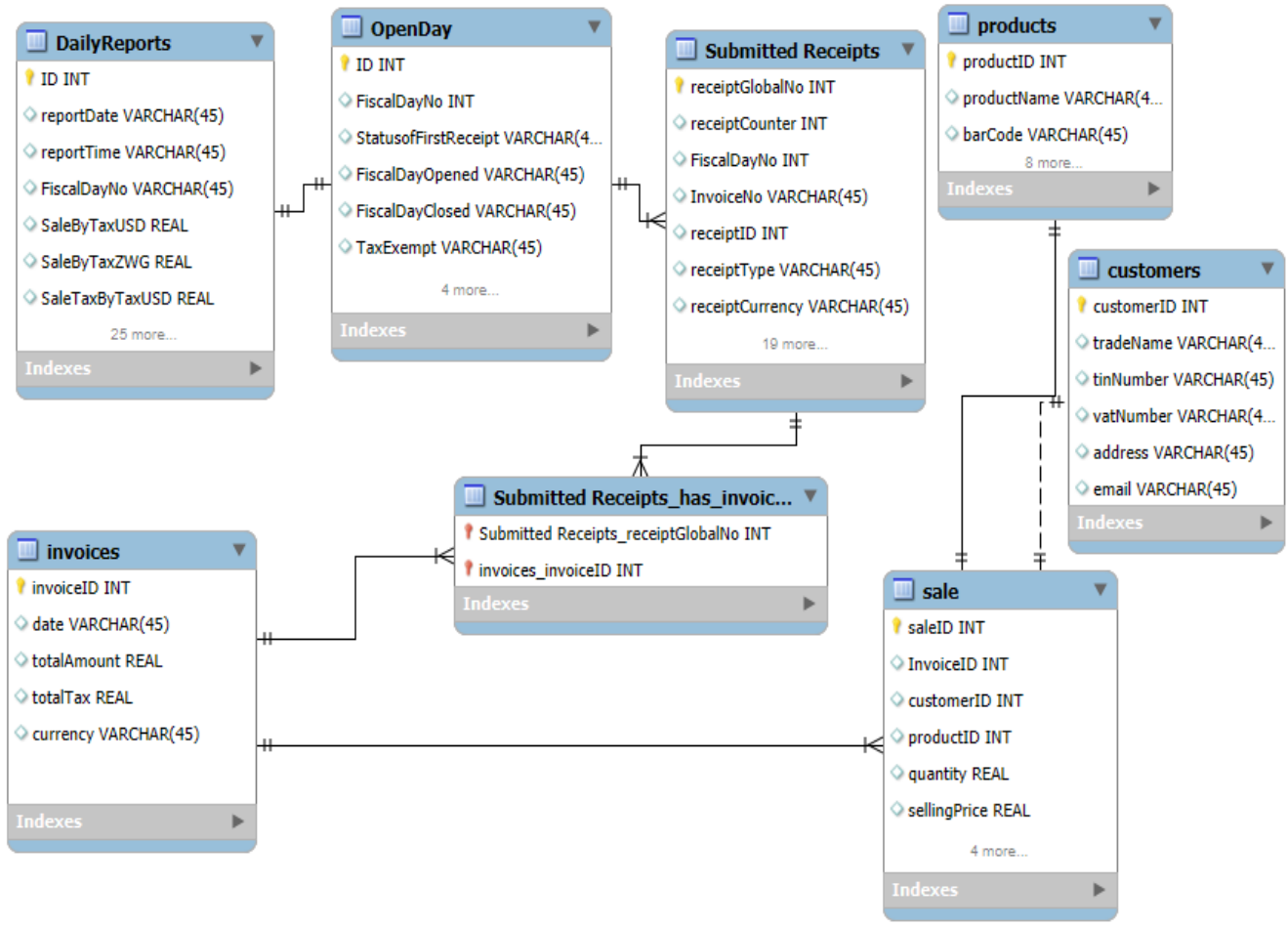


Figure 9 : ER diagram

4.5.2 Relational Schema

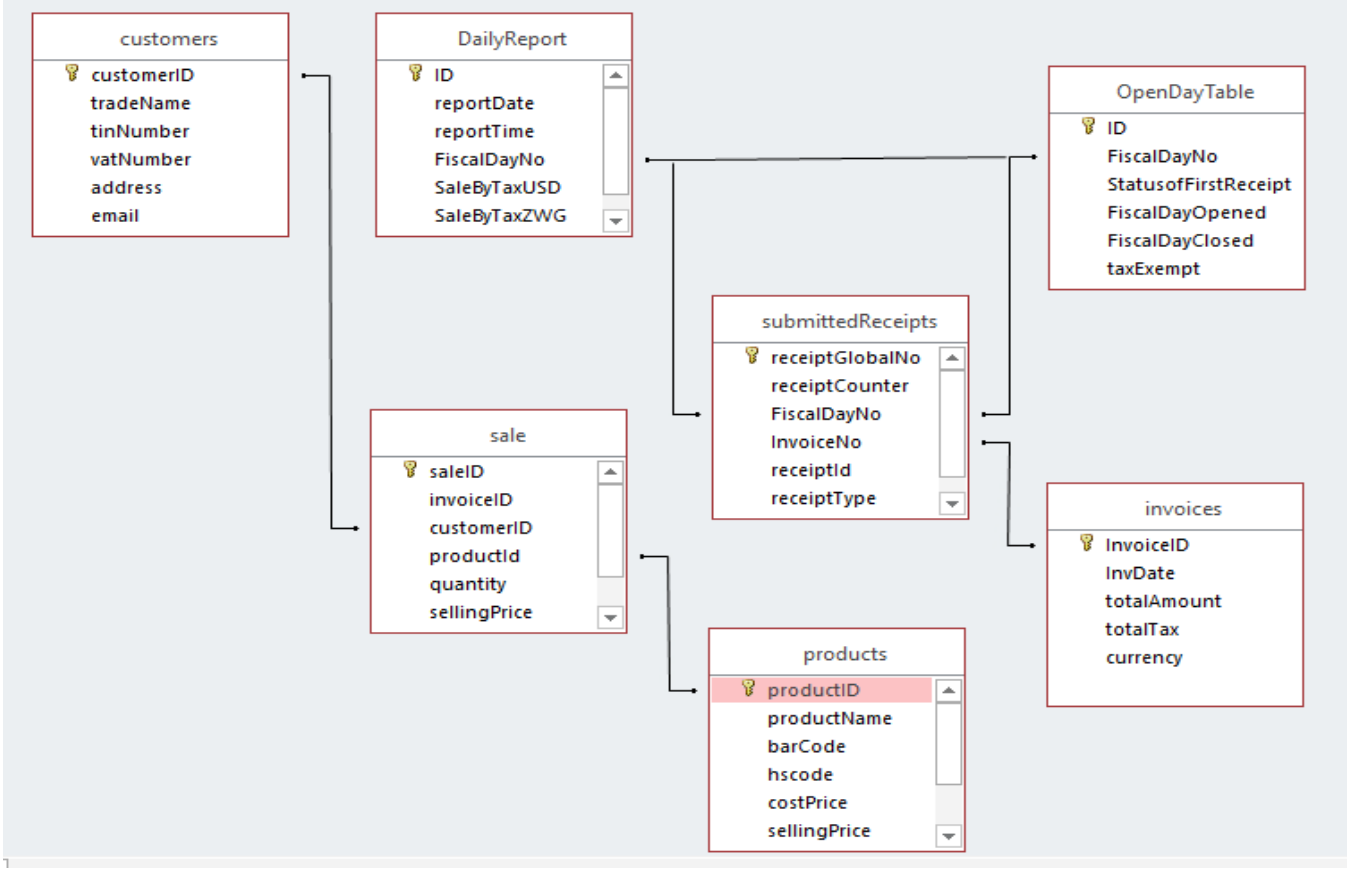


Figure 10 : Relational Schema

4.5.3 Normalization

Ensures atomicity of data by eliminating repeating groups.

Normalized Databases:

Table: dailyReports

Field Name	Data Type	Constraints	Description
ID	INTEGER	PRIMARY KEY AUTOINCREMENT	Unique report ID
reportDate	TEXT	NOT NULL	Date of the report
reportTime	TEXT	NOT NULL	Time of the report
FiscalDayNo	INTEGER	NOT NULL	Reference to fiscal day
reportHash	TEXT	NOT NULL	Hash of the report
reportSignature	TEXT	NOT NULL	Signature of the report
reportJsonBody	TEXT	NOT NULL	JSON body

fiscalDayStatus	TEXT	NOT NULL	Status of fiscal day

Table 6 : Daily report

Table: openDay

Field Name	Data Type	Constraints	Description
ID	INTEGER	PRIMARY KEY AUTOINCREMENT	Unique open day ID
FiscalDayNo	INTEGER	NOT NULL	Fiscal day number
StatusOfFirstReceipt	TEXT	NOT NULL	First receipt status
FiscalDayOpened	TEXT	NOT NULL	Timestamp opened
FiscalDayClosed	TEXT	NOT NULL	Timestamp closed
TaxExempt	INTEGER	NOT NULL	Exempt tax count
TaxZero	INTEGER	NOT NULL	Zero-rated tax count
Tax15	INTEGER	NOT NULL	Standard tax count
TaxWT	INTEGER	NOT NULL	Withholding tax count

Table 7 : Open day table

Table: submittedReceipts

Field Name	Data Type	Constraints	Description
receiptGlobalNo	INTEGER	PRIMARY KEY AUTOINCREMENT	Global receipt number
receiptCounter	INTEGER	NOT NULL	Local counter
FiscalDayNo	INTEGER	NOT NULL	Fiscal day reference
InvoiceNo	INTEGER	NOT NULL	Related invoice number
receiptID	INTEGER		Local receipt ID
receiptType	TEXT	NOT NULL	Type of receipt
receiptCurrency	TEXT	NOT NULL	Currency used
moneyType	TEXT	NOT NULL	Payment method
receiptDate	TEXT	NOT NULL	Date of receipt
receiptTime	TEXT	NOT NULL	Time of receipt
receiptTotal	REAL	NOT NULL	Total with tax
taxCode	TEXT	NOT NULL	Tax code
taxPercent	TEXT	NOT NULL	Tax percentage
taxAmount	REAL	NOT NULL	Tax value
SalesAmountwithTax	REAL	NOT NULL	Total sales including tax
receiptHash	TEXT	NOT NULL	Receipt hash
receiptJsonbody	TEXT	NOT NULL	Receipt body
StatustoFDMS	TEXT	NOT NULL	Status to fiscal system
qrurl	TEXT	NOT NULL	QR code URL
receiptServerSignature	TEXT		Server signature

submitReceiptServerresponseJSON	TEXT		Response from fiscal system
Total15VAT	TEXT	NOT NULL	Total standard VAT
TotalNonVAT	REAL	NOT NULL	Non-VAT sales
TotalExempt	REAL	NOT NULL	Tax-exempt sales
TotalWT	REAL	NOT NULL	Withholding tax

Table 8 : Submitted receipts

Table: sales

Field Name	Data Type	Constraints	Description
saleId	INTEGER	PRIMARY KEY AUTOINCREMENT	Sale ID
invoiceId	INTEGER	FOREIGN KEY REFERENCES invoices(invoiceId)	Related invoice
customerID	INTEGER	FOREIGN KEY REFERENCES customer(customerID)	Buyer
productId	INTEGER	FOREIGN KEY REFERENCES products(productid)	Product sold
quantity	INTEGER	NOT NULL	Quantity sold
sellingPrice	REAL	NOT NULL	Price at time of sale
tax	REAL	NOT NULL	Applied tax
currency	TEXT	NOT NULL	Sale currency

Table 9 : Sales

Table: products

Field Name	Data Type	Constraints	Description
productid	INTEGER	PRIMARY KEY AUTOINCREMENT	Product ID
productName	TEXT	UNIQUE	Name
barcode	TEXT	NOT NULL	Barcode
hsCode	INTEGER	NOT NULL	Harmonized system code
costPrice	REAL	NOT NULL	Cost price
sellingPrice	REAL	NOT NULL	Selling price
sellqty	REAL	NOT NULL	Sold quantity (for stats)
tax	TEXT	NOT NULL	Tax category
stockQty	INTEGER	NOT NULL	Available stock quantity

Table 10 : Products

Table: Invoices

Field Name	Data Type	Constraints	Description

invoiceId	INTEGER	PRIMARY KEY AUTOINCREMENT	Invoice ID
date	TEXT	NOT NULL	Invoice date
totalAmount	REAL	NOT NULL	Total invoice amount
totalTax	REAL	NOT NULL	Total tax
currency	TEXT	NOT NULL	Currency used

Table 11 : Invoices

Table: stockPurchases

Field Name	Data Type	Constraints	Description
purchaseId	INTEGER	PRIMARY KEY AUTOINCREMENT	Purchase ID
date	TEXT		Purchase date
productid	INTEGER	FOREIGN KEY REFERENCES products(productid)	Purchased product
quantity	INTEGER	NOT NULL	Quantity purchased
payMethod	TEXT	NOT NULL	Payment method used
supplier	TEXT	NOT NULL	Supplier name

Table 12 : stock purchases

Table: customer

Field Name	Data Type	Constraints	Description
customerID	INTEGER	PRIMARY KEY AUTOINCREMENT	Customer ID
tradeName	TEXT	NOT NULL	Business or trade name
tinNumber	INT	NOT NULL	TIN
vatNumber	INT	NOT NULL	VAT number
address	TEXT	NOT NULL	Address
email	TEXT	NOT NULL	Email address

Table 13 : Customer

4.6 Program Design

4.6.1 UML-Class Diagram

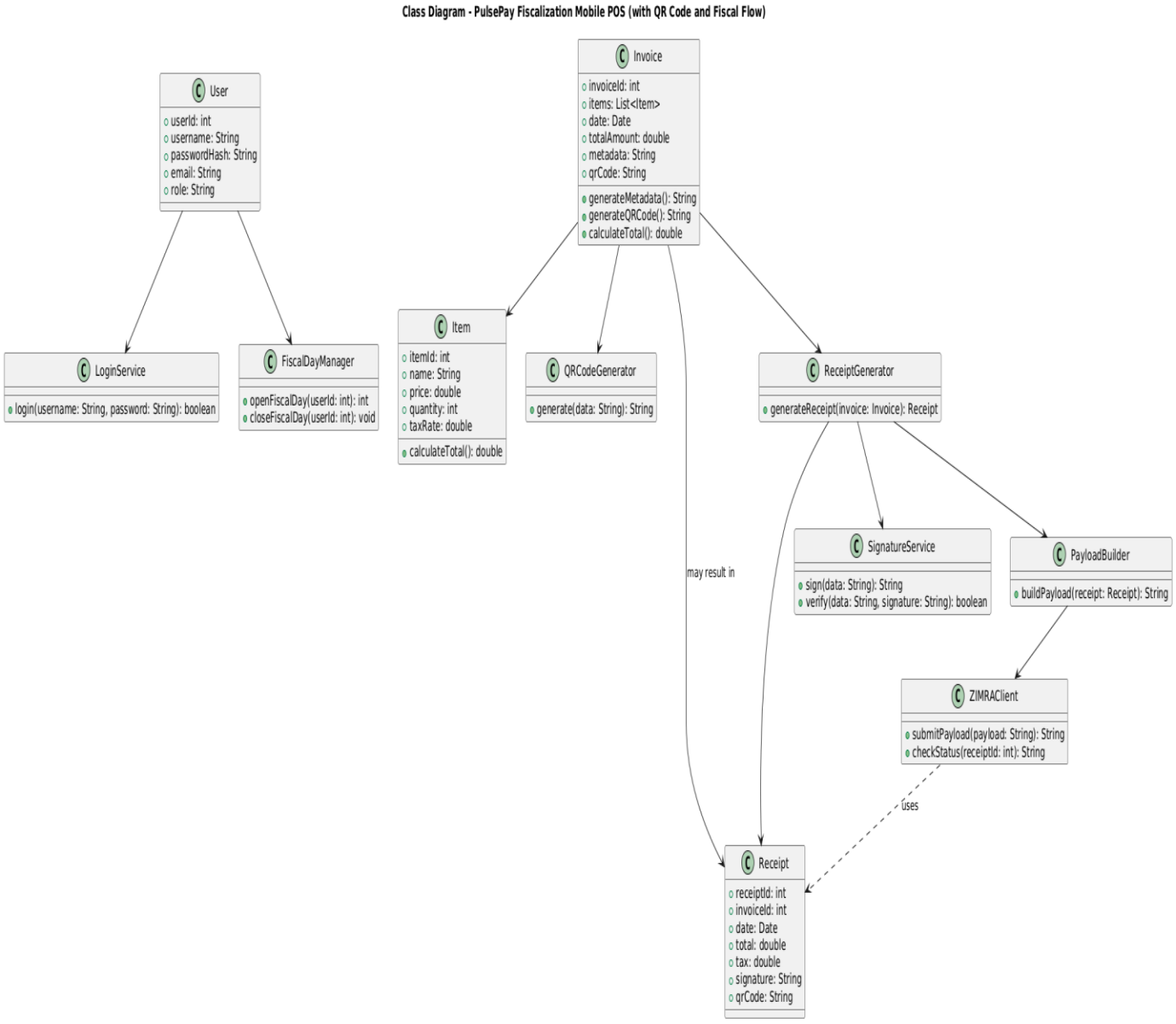


Figure 11 : UML class diagram

4.6.2 UML-Sequence Diagram

The process starts when a taxpayer initially registers or updates their information using the ZIMRA registration portal. After completing this process, the taxpayer is provided with device ID and Activation Key. Device registration must be done once before starting to use a new device. After device registration, it needs to get its configurations (config) from FDMS.

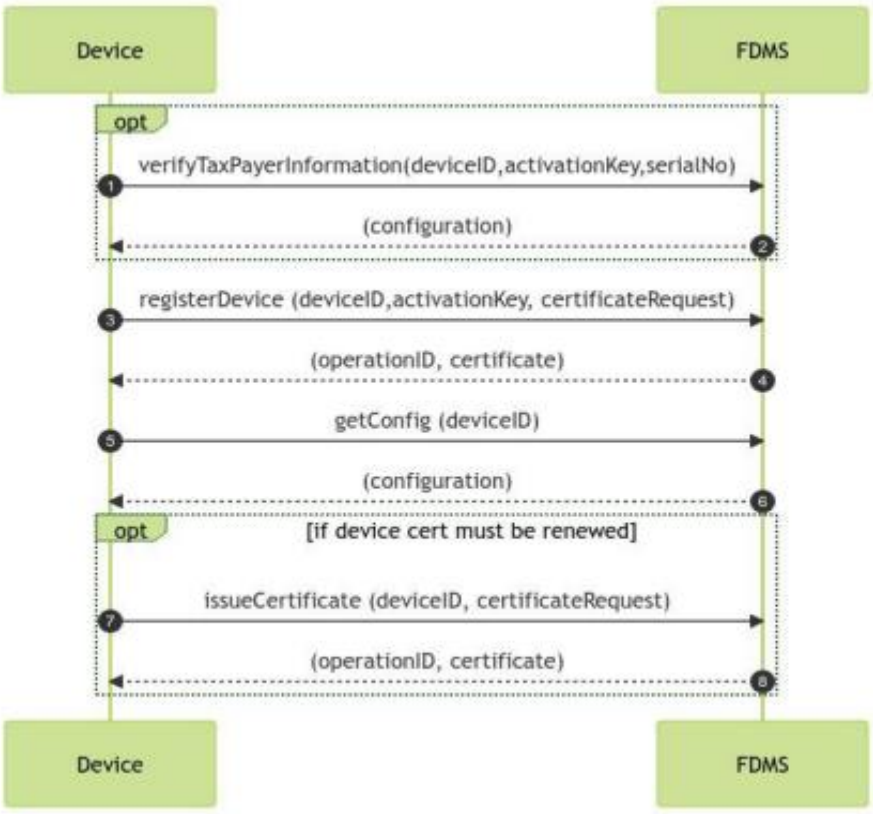


Figure 12 : Device configuration

Fiscal Day Operations:

After successful device registration, it can be used for submitting sales to FDMS. Sales submission is possible only when fiscal day is opened. When work is finished with device, it must close fiscal day.

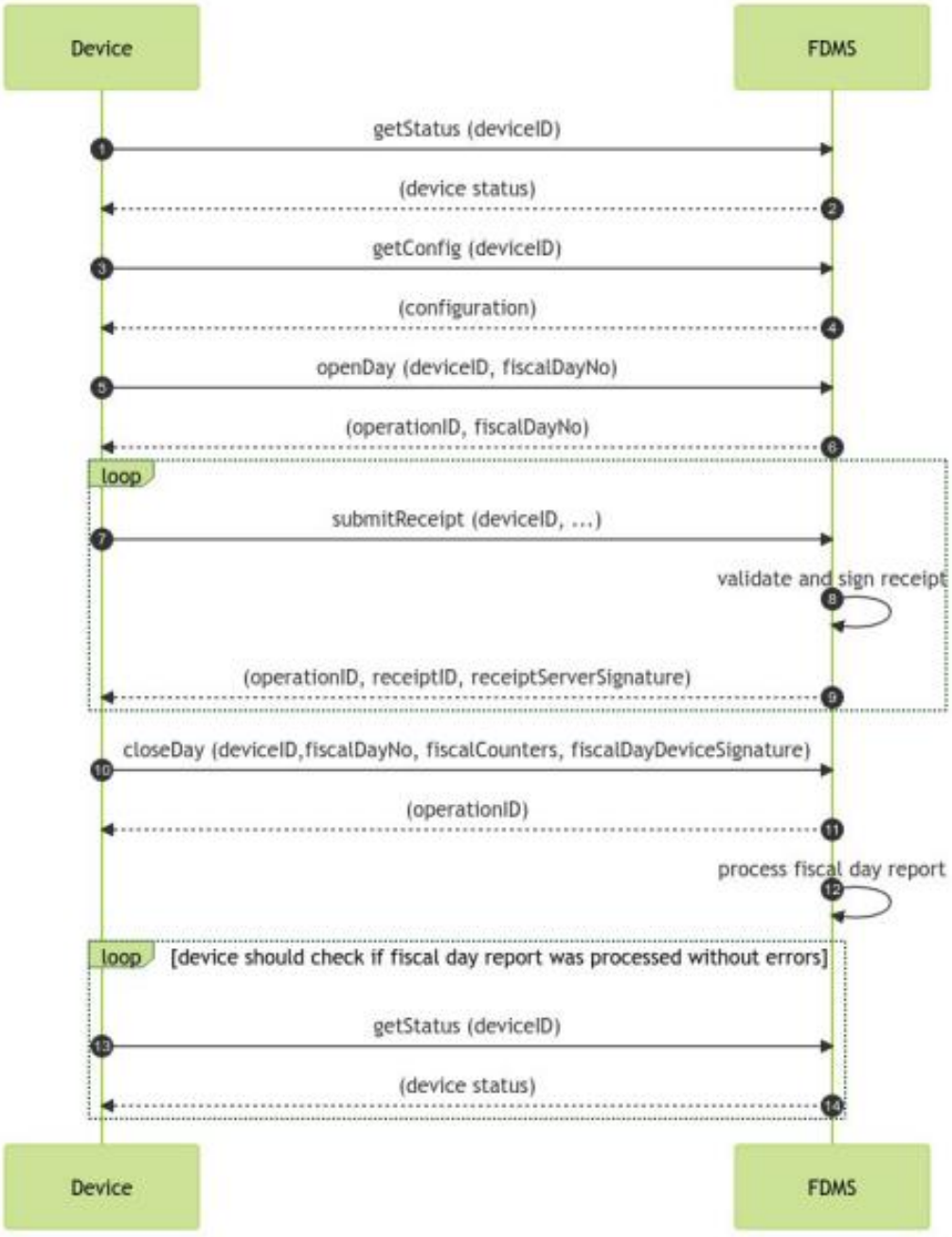


Figure 13 : Fiscal Day operations

System Overall Sequence Diagram:

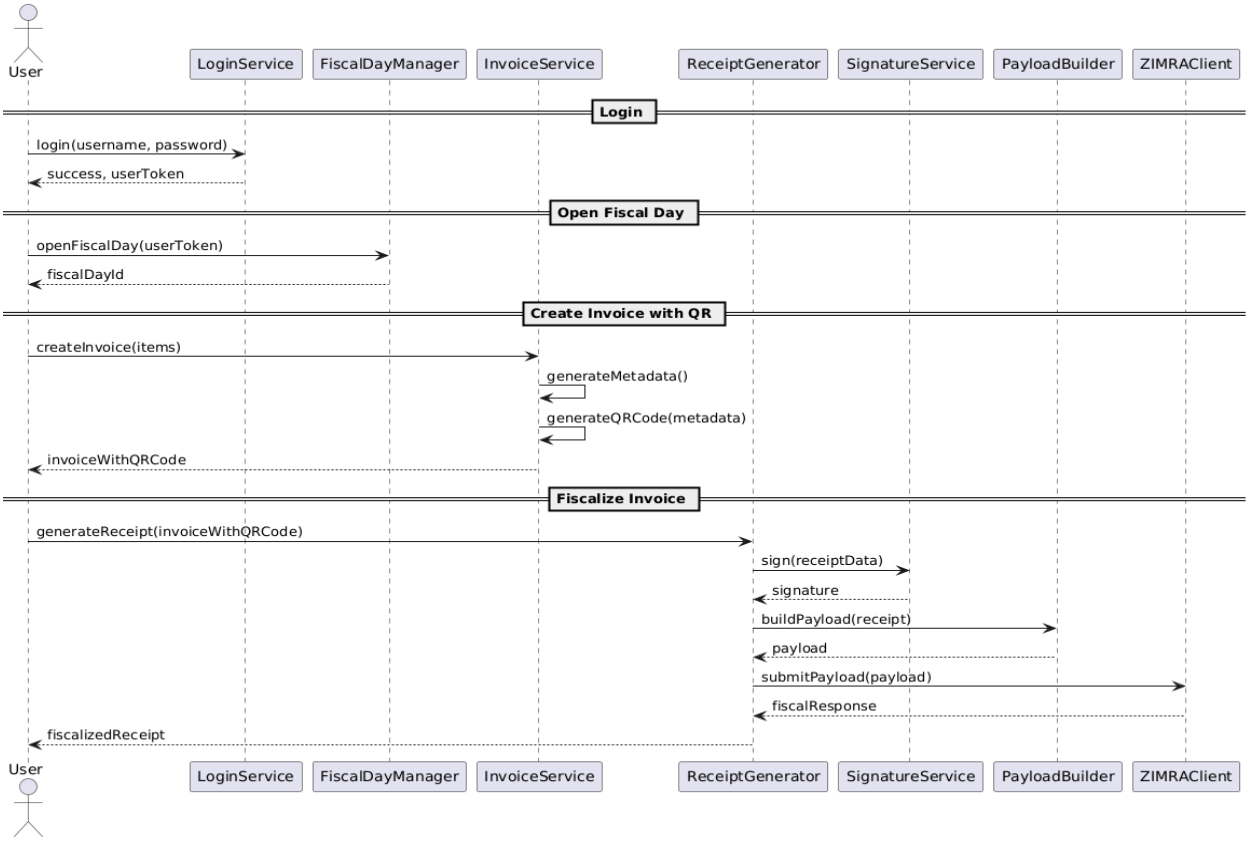


Figure 14 : System sequence diagram

4.6.3 Package Diagram

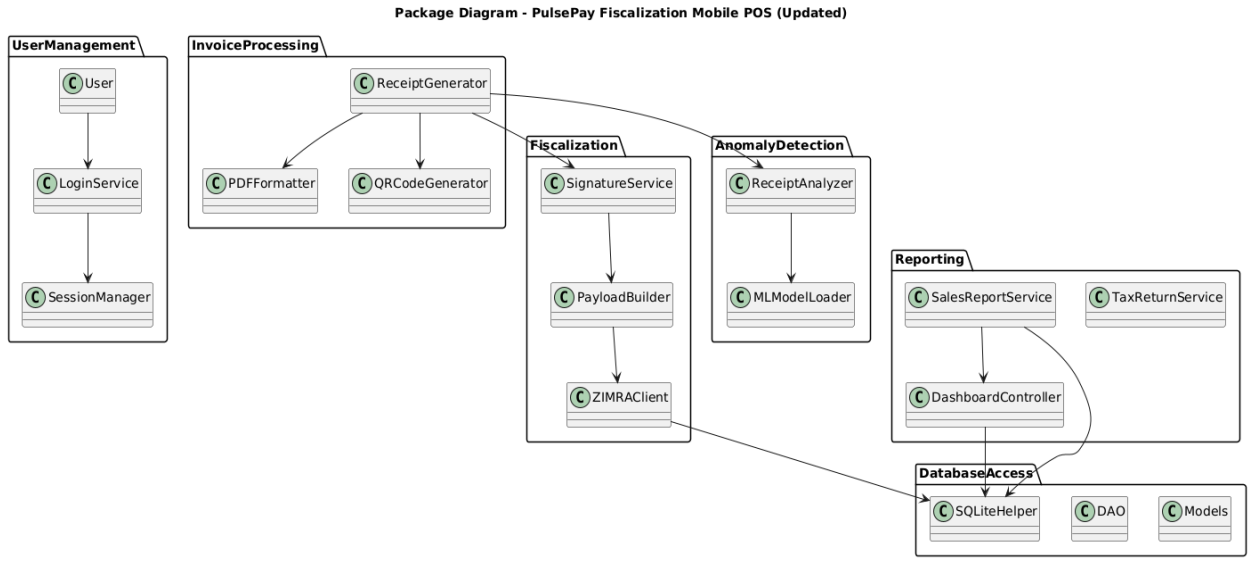


Figure 15 : package diagram

4.6.4 Pseudo Codes for Modules

Pseudocode: Signature Handling via Flutter-Kotlin Bridge

Main Activity Startup

```
Procedure MainActivity_OnCreate()
   Call EnsureBouncyCastleProvider()
End Procedure
```

Ensure Bouncy Castle Provider

```
Procedure EnsureBouncyCastleProvider()
   If BouncyCastle provider is not available
      Add BouncyCastle to system providers
   Else
      Remove existing BouncyCastle provider
      Re-add fresh BouncyCastle provider
   End If
End Procedure
```

Flutter Method Channel Setup

```
Procedure ConfigureFlutterEngine()
   Set up MethodChannel "flutter/kotlin"

   On MethodCall Received:
      If method == "signData"
         Extract filePath, password, data from arguments
         If any argument is null
            Return error "INVALID_ARGS"
```

```
        Else
            signedData ← Call SignData(filePath, password, data)
            Return signedData
        End If

    Else If method == "verifySignature"
        Extract filePath, password, data, signature from arguments
        If any argument is null
            Return error "INVALID_ARGS"
        Else
            isValid ← Call VerifySignature(filePath, password, data, signature)
            Return isValid
        End If

    Else
        Return "Method not implemented"
    End If
End Procedure

Procedure SignData(filePath, password, data) → Map
  Try
    Load keystore from filePath using password
    Get alias from keystore
    Extract privateKey using alias and password

    hashedData ← SHA256(data)

    Initialize signature algorithm SHA256withRSA
    signature.initSign(privateKey)
    signature.update(data)
    signedBytes ← signature.sign()
```

```
    hexHash ← MD5(signedBytes) → hex string
    base64Signature ← Base64Encode(signedBytes)
    first16 ← Call GetFirst16CharsOfSignature(base64Signature)

    Return Map {
      "receiptDeviceSignature_signature_hex" → hexHash,
      "receiptDeviceSignature_signature" → base64Signature,
      "receiptDeviceSignature_signature_md5_first16" → first16
    }
  Catch error
    Return Map { "error" → error.message }
  End Try
End Procedure


Procedure VerifySignature(filePath, password, data, base64Signature) → Boolean
  Try
    Load keystore from filePath using password
    Get alias and publicKey from certificate

    signedBytes ← Base64Decode(base64Signature)

    Initialize verifier with SHA256withRSA
    signature.initVerify(publicKey)
    signature.update(data)

    Return signature.verify(signedBytes)
  Catch error
    Return False
  End Try
End Procedure
```

```
Procedure GetFirst16CharsOfSignature(base64Signature) → String
    If base64Signature is blank
        Raise error "Invalid input"
    End If

    byteArray ← Base64Decode(base64Signature)
    hexStr ← Convert byteArray to hex string
    hexBytes ← Convert hex string back to byte array
    md5Hash ← MD5(hexBytes)
    Return first 16 hex characters of md5Hash
End Procedure
```

Pseudocode: generateReceiptTaxes(receiptItems)

```
Procedure generateReceiptTaxes(receiptItems: List of Items) → List of Tax Maps
    Initialize empty Map taxGroups (Key: taxID, Value: Map with tax summary)

    For each item in receiptItems Do
        Set taxID ← item["taxID"]
        Set taxPercent ← item["taxPercent"]
        Set total ← item["total"]

        If taxID not in taxGroups Then
            Initialize taxGroups[taxID] with:
                "taxID" ← taxID
                "taxPercent" ← taxPercent (blank if empty)
                "taxCode" ← item["taxCode"]
                "taxAmount" ← 0.0
                "salesAmountWithTax" ← 0.0
```

```
        End If

        // Compute taxAmount based on taxPercent
        If taxPercent is empty Then
            taxAmount ← 0.00
        Else If taxPercent equals "15.00" Then
            taxAmount ← total - (total / 1.15)
        Else
            taxAmount ← 0.0
        End If

        // Accumulate totals in taxGroups
        taxGroups[taxID]["taxAmount"] ← taxGroups[taxID]["taxAmount"] + taxAmount
        taxGroups[taxID]["salesAmountWithTax"] ← taxGroups[taxID]["salesAmountWithTax"] +
total
    End For

    // Convert taxGroups map to list with formatting
    Initialize resultList as empty List

    For each tax in taxGroups.values Do
        Set taxID ← tax["taxID"]
        Set taxCode ← tax["taxCode"]
        Set isGroupA ← (taxCode == "A" OR taxID == 1)

        Create newTaxMap with:
            "taxID" ← taxID as String
            If isGroupA is False Then
                Add "taxPercent" ← tax["taxPercent"]
            End If
            "taxCode" ← taxCode
```

```
        "taxAmount" ← "0" if isGroupA is True, else round(tax["taxAmount"], 2)
        "salesAmountWithTax" ← tax["salesAmountWithTax"]

      Append newTaxMap to resultList
   End For

   Return resultList
End Procedure
```

Pseudo Code: submitReceipt

```
Procedure submitReceipt()
   // Generate JSON data for receipt
   jsonString ← generateFiscalJSON()
   receiptJson ← Encode(jsonString)
   Display snackbar notification:
      Title: "Fiscalizing"
      Message: "Processing"
   // Ping ZIMRA server
   pingResponse ← ping()
   // Generate JSON body again
   receiptJsonbody ← generateFiscalJSON()
   jsonData ← Decode(receiptJsonbody)
   Initialize db as instance of DatabaseHelper
   If receiptPayments exist in jsonData Then
      moneyType ← jsonData['receipt']['receiptPayments'][0]['moneyTypeCode']
   Else
      moneyType ← ""
```

```
    Display receiptDate and invoiceNo from jsonData
    fiscalDayNo ← db.getlatestFiscalDay()
    receiptTotal ← Parse as Double(jsonData['receipt']['receiptTotal'])
    formattedDeviceID ← deviceID left-padded to 10 digits
    formattedDate ← Format receiptDate to "ddMMyyyy"
    latestReceiptGlobalNo ← db.getLatestReceiptGlobalNo()
    currentGlobalNo ← latestReceiptGlobalNo + 1
    formattedReceiptGlobalNo ← currentGlobalNo left-padded to 10 digits
    receiptQrData ← Use first16Chars (or use hash function)
    qrurl ← genericzimraqrurl + formattedDeviceID + formattedDate +
formattedReceiptGlobalNo + receiptQrData

    If pingResponse == "200" Then
        apiEndpoint ← ZIMRA endpoint URL
        deviceModelName ← "Server"
        deviceModelVersion ← "v1"
        securityContext ← Create SSL context
        response ← SubmitReceipts to ZIMRA with API endpoint and receiptJsonbody
        Show snackbar with response
        responseBody ← Decode(response["responseBody"])
        statusCode ← response["statusCode"]
        submitReceiptServerresponseJson ← Stringify responseBody

        If statusCode == 200 Then
            Try
                dbinit ← dbHelper.initDB()
                Insert into 'submittedReceipts':
```

Use values from jsonData and responseBody:

- receiptCounter, invoiceNo, receiptID, receiptType, receiptCurrency
- moneyType, receiptDate, receiptTime, receiptTotal, receiptHash
- taxCode = "C", taxPercent = "15.00", taxAmount, SalesAmountwithTax
- StatustoFDMS = "Submitted", qrurl, receiptServerSignature
- submitReceiptServerresponseJSON
- Total15VAT, TotalNonVAT, TotalExempt, TotalWT = 0

Call generateInvoiceFromJson(jsonData, qrurl)

Catch error

Show error snackbar

Else

Try

dbinit ← dbHelper.initDB()

Insert into 'submittedReceipts' with:

Same values as above, but:

- receiptID = 0
- StatustoFDMS = "NOTSubmitted"
- receiptServerSignature = ""
- submitReceiptServerresponseJSON = "noresponse"

Call generateInvoiceFromJson(jsonData, qrurl)

Catch error

Show error snackbar

Else

Try

dbinit ← dbHelper.initDB()

Insert into 'submittedReceipts' with:

Same structure as above for NOTSubmitted

Call generateInvoiceFromJson(jsonData, qrurl)

Catch error

Show error snackbar

End Procedure

4.7 Interface Design

Login Page:

- This is the log in page where the user will enter their system credentials to access the system.

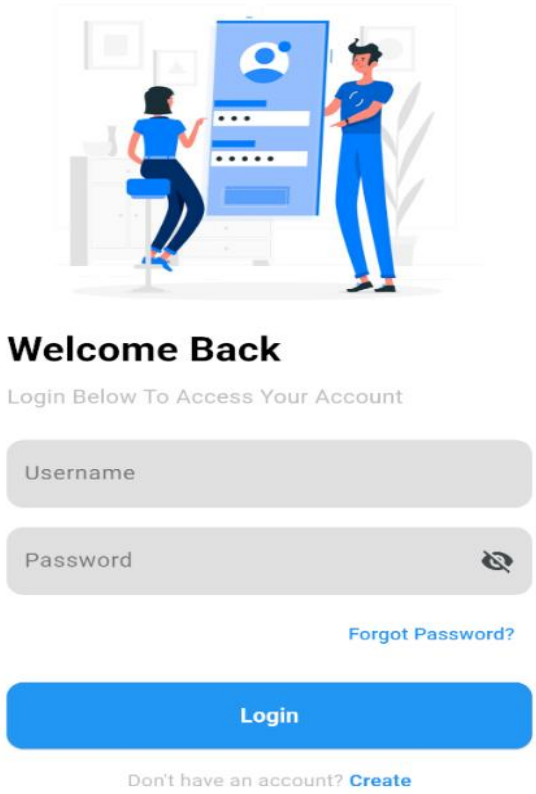


Figure 16 : Login page

Main Menu:

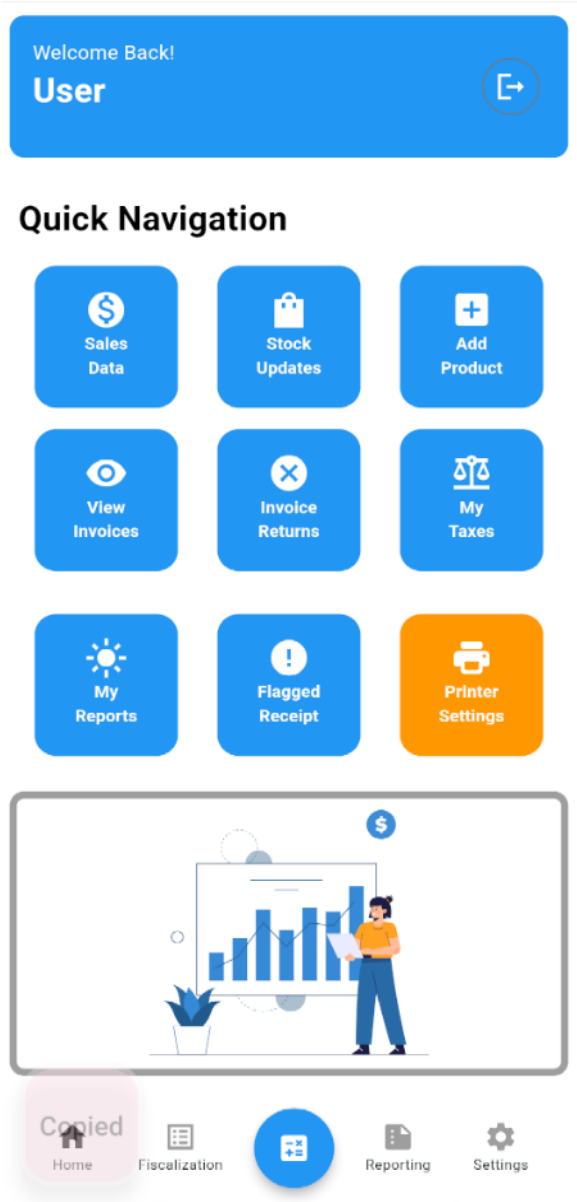


Figure 17 : Main menu

- The system's entry point where system navigation will begin.
- The user is provided with a quick navigation pan that has the essential system options that will allow the user to do the priority operations

Selling terminal:

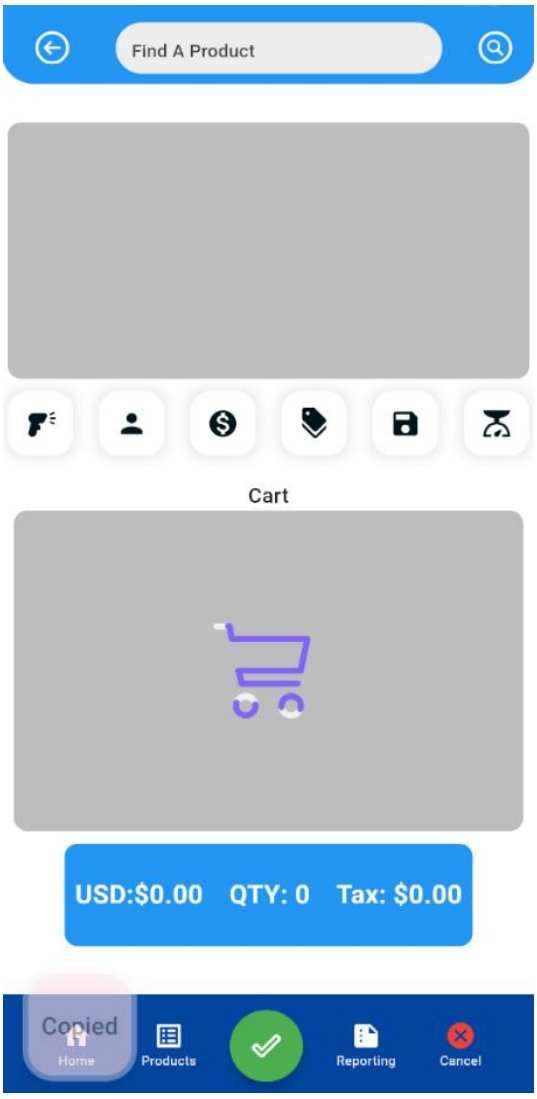
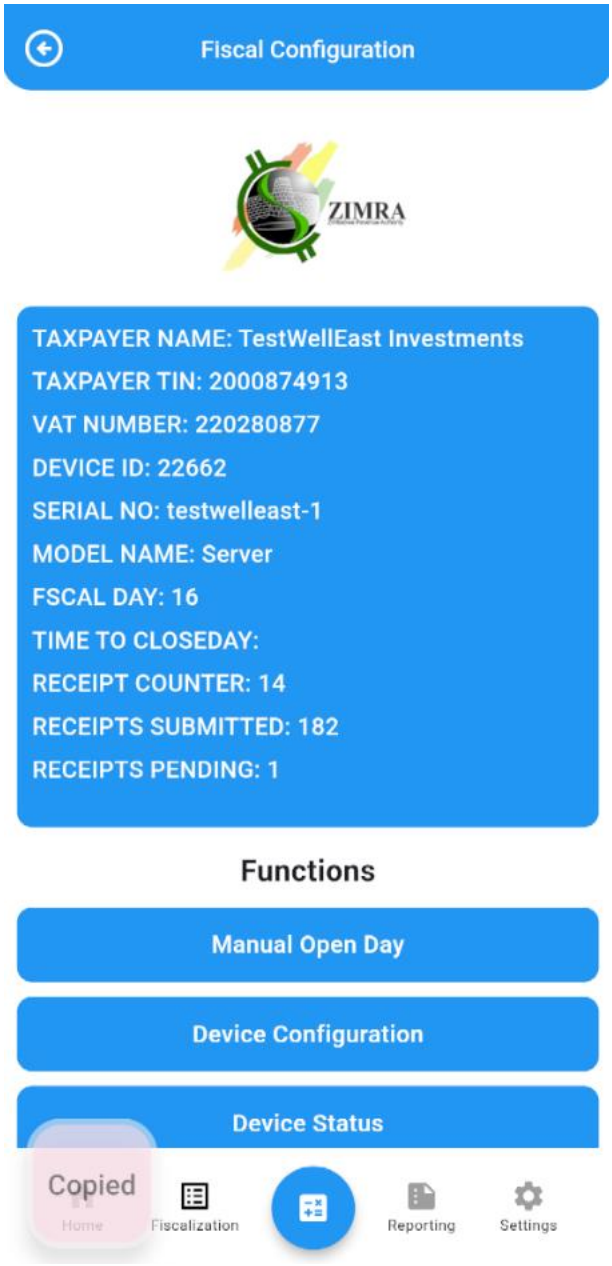


Figure 18 : Point of sale terminal

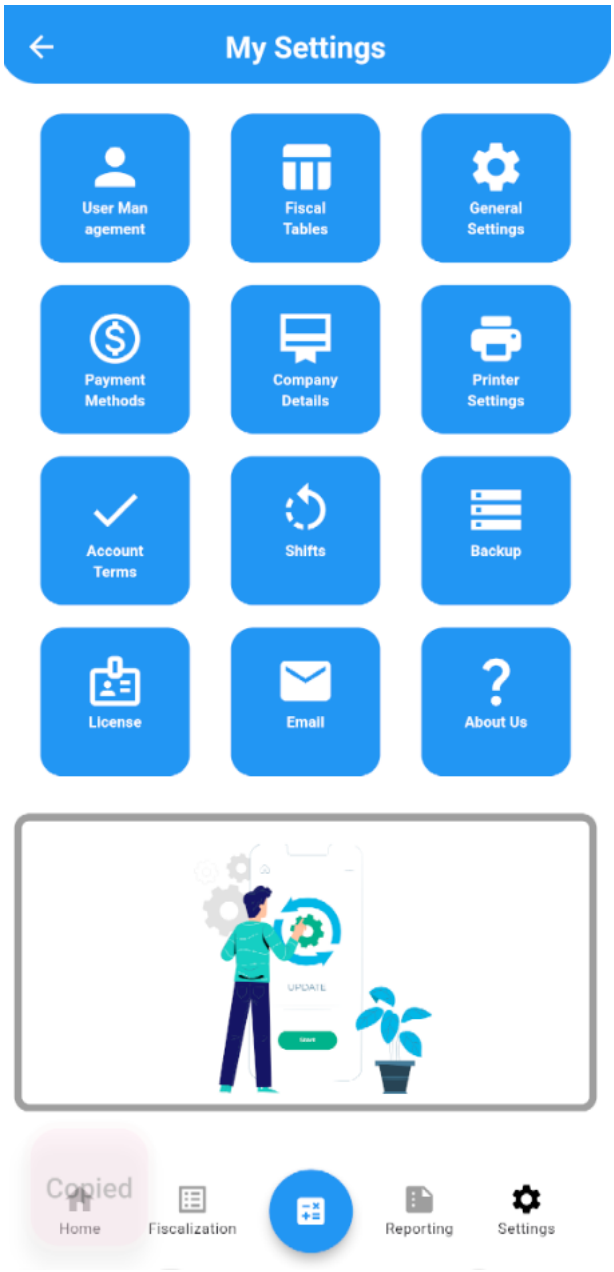
- This is the selling terminal that will be used to select and charge for the services that will be provided by user
- Will allow product selection , customer selection and invoice currency selection

Fiscal Page:



- The fiscal page will pave way for the handling of fiscalization processes. the users will use this page to open fiscal days, check device configuration, check device status, close fiscal day and submit missing receipts
- On this page the user will also see their device details and fiscal days related data.

Settings Page:



- The settings page brings the user to a number of settings options to set up their device.
- Mandatory settings that should not left out will be payment methods as they are crucial in invoice generation.

Taxes Page:

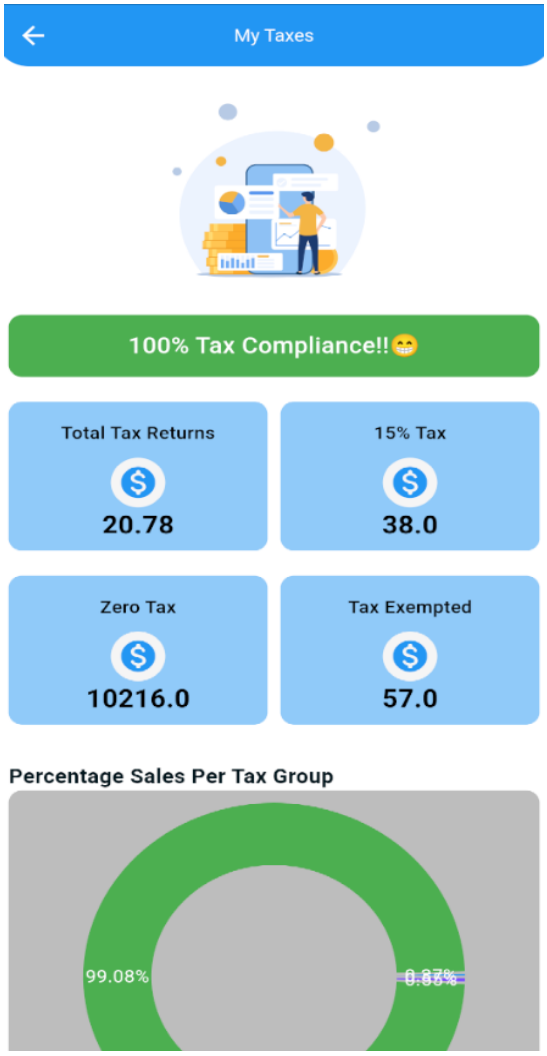


Figure 19 : taxes page

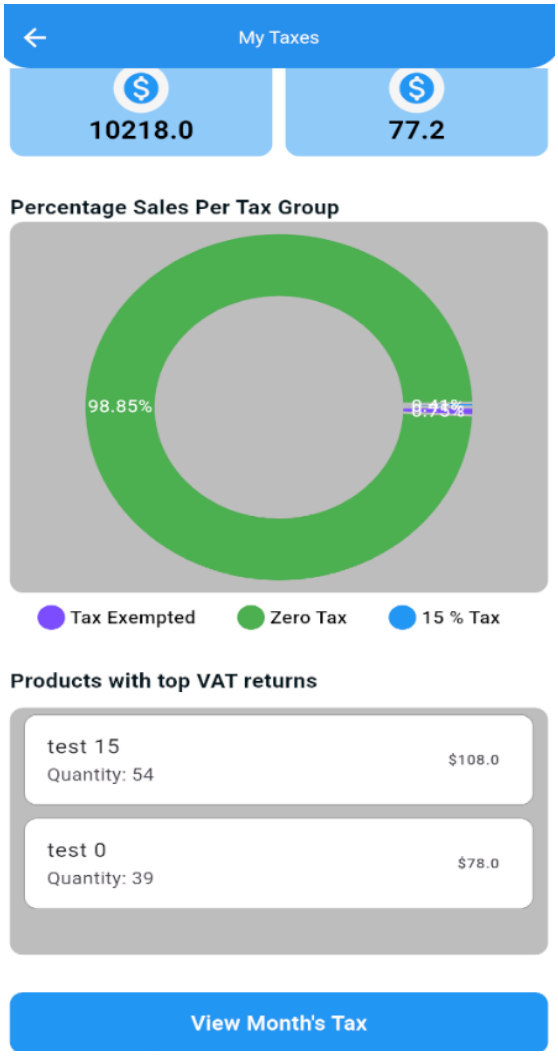
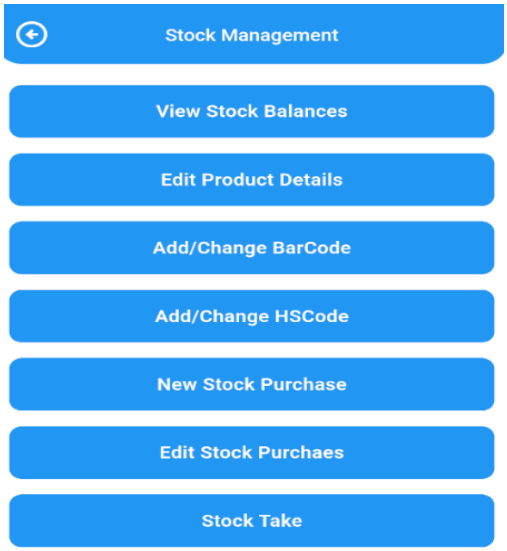


Figure 20 : taxes page 2

- The taxes page shows users a break-down of their taxes and whether they are tax compliant or not
- It shows the total sales per each tax group and also the amount of tax for the respective tax groups, further down the screen the users will have an option to see their monthly tax returns

Stock Management:



- This page allows the user to handle stock management operations.

Figure 21 : Stock management

4.8 Conclusion

This chapter presented the comprehensive design and system models of the fiscalized POS mobile application. The proposed solution ensures compliance with fiscal regulations while offering a robust and user-friendly interface for retail transactions. The next chapter will cover the implementation and testing phases of the system.

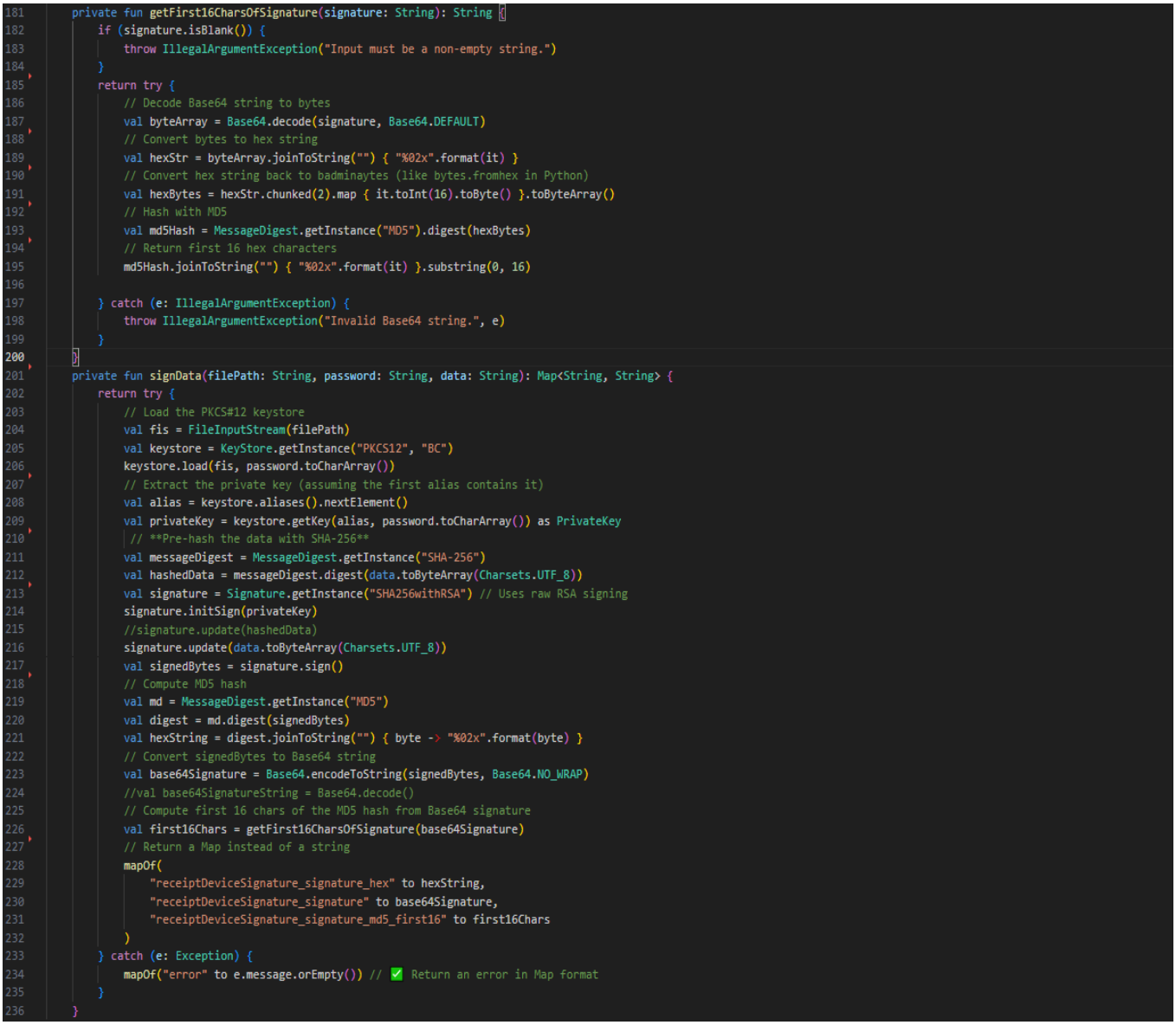
Chapter Five: Implementation and Testing

5.1 Sample Code of Major Modules

Below is the code for the major modules implemented in the mobile fiscalization application:

1. Signature Generation Module

```kotlin
class MainActivity : FlutterActivity() {
    private val CHANNEL = "flutter/kotlin"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        ensureBouncyCastleProvider()
    }

    private fun ensureBouncyCastleProvider() {
        val provider = Security.getProvider("BC")
        if (provider == null) {
            Security.addProvider(BouncyCastleProvider())  // Add BouncyCastle if not present
        } else {
            Security.removeProvider("BC")
            Security.addProvider(BouncyCastleProvider())  // Refresh provider
        }
    }

    override fun configureFlutterEngine(flutterEngine: FlutterEngine) {
        super.configureFlutterEngine(flutterEngine)
        MethodChannel(flutterEngine.dartExecutor.binaryMessenger, CHANNEL).setMethodCallHandler { call, result ->
            when (call.method) {
                "signData" -> {
                    val filePath: String? = call.argument("filePath")
                    val password: String? = call.argument("password")
                    val data: String? = call.argument("data")
                    if (filePath != null && password != null && data != null) {
                        val signedData: Map<String, String> = signData(filePath, password, data)
                        result.success(signedData) // ✅ Correct return type
                    } else {
                        result.error("INVALID_ARGS", "File path, password, or data is null", null)
                    }
                }

                "verifySignature" -> {
                    val filePath: String? = call.argument("filePath")
                    val password: String? = call.argument("password")
                    val data: String? = call.argument("data")
                    val signature: String? = call.argument("signature")
                    if (filePath != null && password != null && data != null && signature != null) {
                        val isValid: Boolean = verifySignature(filePath, password, data, signature)
                        result.success(isValid)
                    } else {
                        result.error("INVALID_ARGS", "File path, password, data, or signature is null", null)
                    }
                }
                else -> result.notImplemented()
            }
        }
    }

```



```
    private fun getFirst16CharsOfSignature(signature: String): String {
        if (signature.isBlank()) {
            throw IllegalArgumentException("Input must be a non-empty string.")
        }
        return try {
            // Decode Base64 string to bytes
            val byteArray = Base64.decode(signature, Base64.DEFAULT)
            // Convert bytes to hex string
            val hexStr = byteArray.joinToString("") { "%02x".format(it) }
            // Convert hex string back to badminaytes (like bytes.fromhex in Python)
            val hexBytes = hexStr.chunked(2).map { it.toInt(16).toByte() }.toByteArray()
            // Hash with MD5
            val md5Hash = MessageDigest.getInstance("MD5").digest(hexBytes)
            // Return first 16 hex characters
            md5Hash.joinToString("") { "%02x".format(it) }.substring(0, 16)

        } catch (e: IllegalArgumentException) {
            throw IllegalArgumentException("Invalid Base64 string.", e)
        }
    }
    private fun signData(filePath: String, password: String, data: String): Map<String, String> {
        return try {
            // Load the PKCS#12 keystore
            val fis = FileInputStream(filePath)
            val keystore = KeyStore.getInstance("PKCS12", "BC")
            keystore.load(fis, password.toCharArray())
            // Extract the private key (assuming the first alias contains it)
            val alias = keystore.aliases().nextElement()
            val privateKey = keystore.getKey(alias, password.toCharArray()) as PrivateKey
            // **Pre-hash the data with SHA-256**
            val messageDigest = MessageDigest.getInstance("SHA-256")
            val hashedData = messageDigest.digest(data.toByteArray(Charsets.UTF_8))
            val signature = Signature.getInstance("SHA256withRSA") // Uses raw RSA signing
            signature.initSign(privateKey)
            //signature.update(hashedData)
            signature.update(data.toByteArray(Charsets.UTF_8))
            val signedBytes = signature.sign()
            // Compute MD5 hash
            val md = MessageDigest.getInstance("MD5")
            val digest = md.digest(signedBytes)
            val hexString = digest.joinToString("") { byte -> "%02x".format(byte) }
            // Convert signedBytes to Base64 string
            val base64Signature = Base64.encodeToString(signedBytes, Base64.NO_WRAP)
            //val base64SignatureString = Base64.decode()
            // Compute first 16 chars of the MD5 hash from Base64 signature
            val first16Chars = getFirst16CharsOfSignature(base64Signature)
            // Return a Map instead of a string
            mapOf(
                "receiptDeviceSignature_signature_hex" to hexString,
                "receiptDeviceSignature_signature" to base64Signature,
                "receiptDeviceSignature_signature_md5_first16" to first16Chars
            )
        } catch (e: Exception) {
            mapOf("error" to e.message.orEmpty()) // ✅ Return an error in Map format
        }
    }
```

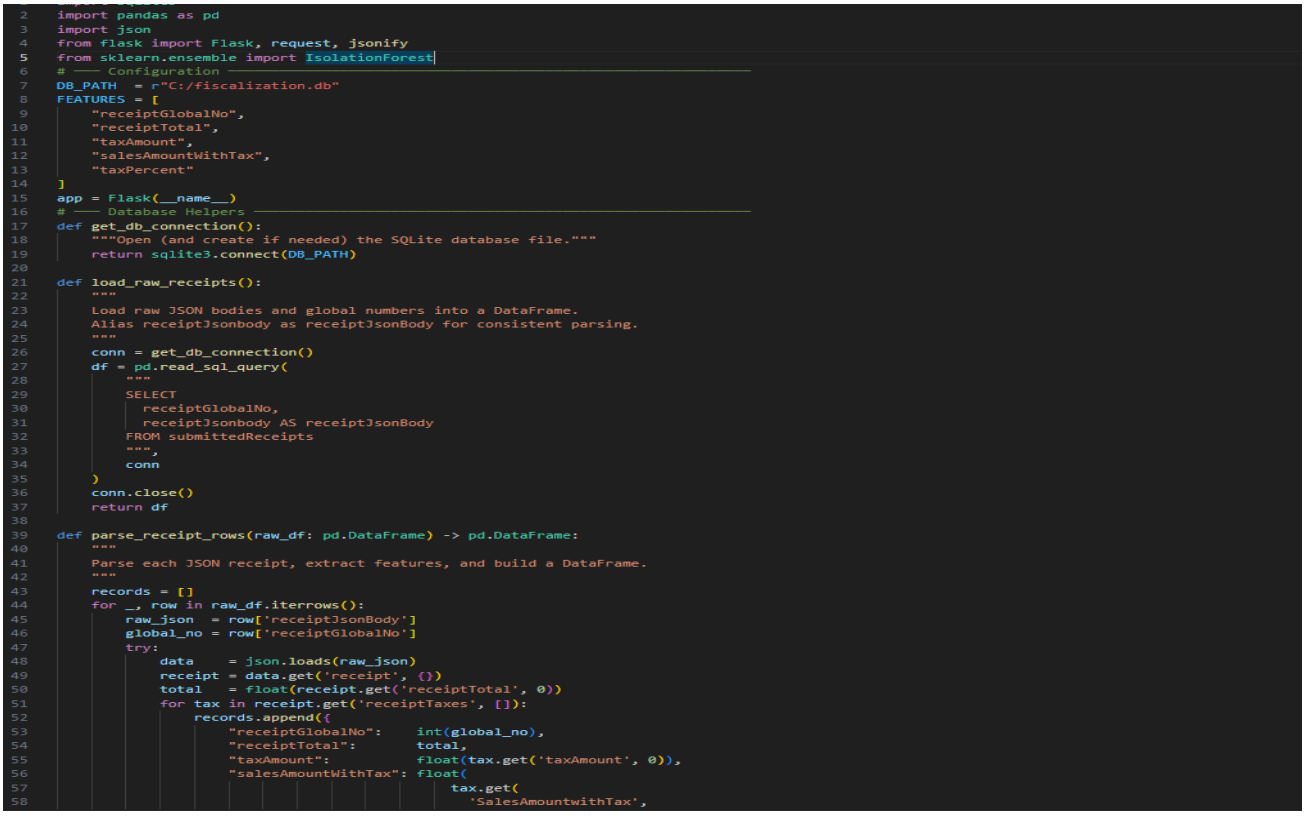
Figure 22 : Signature generation module

2. Fiscal End Of Day Module

```
Future<(
  List<FiscalDayCounter> invoices,
  List<FiscalDayCounter> creditNotes,
  List<FiscalDayCounter> balances,
  String concatenatedString
)> buildFiscalDayCountersAndConcat(
    int fiscalDayNo,
) async {
  final invMap  = <String, FiscalDayCounter>{};
  final crdMap  = <String, FiscalDayCounter>{};
  final balMap  = <String, FiscalDayCounter>{};

  DatabaseHelper dbHelper = DatabaseHelper();
  final db = await dbHelper.initDB();

  final rows = await db.query(
    'submittedReceipts',
    columns: ['receiptType','receiptJsonbody'],
    where: 'FiscalDayNo = ?',
    whereArgs: [fiscalDayNo],
  );

  for (final row in rows) {
    final receiptType = row['receiptType'] as String;
    final body        = jsonDecode(row['receiptJsonbody'] as String);
    final r           = body['receipt'] as Map<String, dynamic>;
    final curr        = r['receiptCurrency'] as String;
    final isCredit    = receiptType != 'FISCALINVOICE';

    for (final t in r['receiptTaxes'] as List<dynamic>) {
      final rawTaxAmt = t['taxAmount'];
      final rawSales  = t['salesAmountWithTax'];
      final taxAmt    = rawTaxAmt is num ? rawTaxAmt.toDouble() : double.tryParse(rawTaxAmt.toString()) ?? 0;
      final salesAmt  = rawSales is num ? rawSales.toDouble() : double.tryParse(rawSales.toString()) ?? 0;

      final perc      = double.parse(t['taxPercent'] as String);
      final taxId     = int.parse(t['taxID'].toString());

      if (!isCredit) {
        final sbtKey = 'SaleByTax|$curr|${perc.toStringAsFixed(2)}|$taxId';
        invMap.putIfAbsent(sbtKey, () => FiscalDayCounter(
          type: 'SaleByTax', currency: curr, percent: perc, taxID: taxId))
          .accumulate(salesAmt);

        final sttKey = 'SaleTaxByTax|$curr|${perc.toStringAsFixed(2)}|$taxId';
        invMap.putIfAbsent(sttKey, () => FiscalDayCounter(
          type: 'SaleTaxByTax', currency: curr, percent: perc, taxID: taxId))
          .accumulate(taxAmt);
      } else {
        final cbtKey = 'CreditNoteByTax|$curr|${perc.toStringAsFixed(2)}|$taxId';
        crdMap.putIfAbsent(cbtKey, () => FiscalDayCounter(
          type: 'CreditNoteByTax', currency: curr, percent: perc, taxID: taxId))
          .accumulate(salesAmt); // accumulate positive value

        final cttKey = 'CreditNoteTaxByTax|$curr|${perc.toStringAsFixed(2)}|$taxId';
        crdMap.putIfAbsent(cttKey, () => FiscalDayCounter(
          type: 'CreditNoteTaxByTax', currency: curr, percent: perc, taxID: taxId))
          .accumulate(taxAmt); // accumulate positive value
      }
    }

    for (final p in r['receiptPayments'] as List<dynamic>) {
      final mType   = p['moneyTypeCode'] as String;
      final rawAmt  = p['paymentAmount'];
      final amt     = rawAmt is num ? rawAmt.toDouble() : double.tryParse(rawAmt.toString()) ?? 0;

      final bKey = 'BalanceByMoneyType|$curr|$mType';
      balMap.putIfAbsent(bKey, () => FiscalDayCounter(
        type: 'BalanceByMoneyType', currency: curr, moneyType: mType))
        .accumulate(amt);
    }
  }

  final allCounters = [
    ...invMap.values,
    ...crdMap.values,
    ...balMap.values,
  ];

  const counterOrder = [
    'SaleByTax',
    'SaleTaxByTax',
    'CreditNoteByTax',
    'CreditNoteTaxByTax',
    'BalanceByMoneyType',
  ];

  allCounters.sort((a, b) =>
    counterOrder.indexOf(a.type).compareTo(counterOrder.indexOf(b.type)));

  final concat = StringBuffer();
  for (final c in allCounters) {
    concat.write(c.toConcatString());
  }

  final invoices = allCounters
      .where((c) => c.type.startsWith('Sale') && double.parse(c.value.toStringAsFixed(2)) != 0.0)
      .toList();
  final creditNotes = allCounters
      .where((c) => c.type.startsWith('CreditNote') && double.parse(c.value.toStringAsFixed(2)) != 0.0)
      .toList();
  final balances = allCounters
      .where((c) => c.type == 'BalanceByMoneyType' && double.parse(c.value.toStringAsFixed(2)) != 0.0)
      .toList();

  return (invoices, creditNotes, balances, concat.toString());
}
```

Figure 23 : End of day module

3. Receipt Anomaly detection module



```
import pandas as pd
import json
from flask import Flask, request, jsonify
from sklearn.ensemble import IsolationForest
# ── Configuration ──────────────────────────────────────────────
DB_PATH  = r"C:/fiscalization.db"
FEATURES = [
    "receiptGlobalNo",
    "receiptTotal",
    "taxAmount",
    "salesAmountWithTax",
    "taxPercent"
]
app = Flask(__name__)
# ── Database Helpers ───────────────────────────────────────────
def get_db_connection():
    """Open (and create if needed) the SQLite database file."""
    return sqlite3.connect(DB_PATH)

def load_raw_receipts():
    """
    Load raw JSON bodies and global numbers into a DataFrame.
    Alias receiptJsonbody as receiptJsonBody for consistent parsing.
    """
    conn = get_db_connection()
    df = pd.read_sql_query(
        """
        SELECT
          receiptGlobalNo,
          receiptJsonbody AS receiptJsonBody
        FROM submittedReceipts
        """,
        conn
    )
    conn.close()
    return df

def parse_receipt_rows(raw_df: pd.DataFrame) -> pd.DataFrame:
    """
    Parse each JSON receipt, extract features, and build a DataFrame.
    """
    records = []
    for _, row in raw_df.iterrows():
        raw_json  = row['receiptJsonBody']
        global_no = row['receiptGlobalNo']
        try:
            data    = json.loads(raw_json)
            receipt = data.get('receipt', {})
            total   = float(receipt.get('receiptTotal', 0))
            for tax in receipt.get('receiptTaxes', []):
                records.append({
                    "receiptGlobalNo":    int(global_no),
                    "receiptTotal":       total,
                    "taxAmount":          float(tax.get('taxAmount', 0)),
                    "salesAmountWithTax": float(
                                              tax.get(
                                                'SalesAmountwithTax',
                                                tax.get('salesAmountWithTax', 0)
                                              )
                                            ),
                    "taxPercent":         float(tax.get('taxPercent', 0))
                })
        except Exception as e:
            print(f"Error parsing JSON for ID {global_no}: {e}")
    return pd.DataFrame(records)

# ── Model Training ─────────────────────────────────────────────
def train_model() -> IsolationForest:
    raw_df = load_raw_receipts()
    df     = parse_receipt_rows(raw_df)
    X      = df[FEATURES]
    model  = IsolationForest(
        n_estimators=150,
        max_samples='auto',
        contamination=0.02,
        random_state=42
    )
    model.fit(X)
    return model

# Train on startup
model = train_model()

# ── API Endpoint ───────────────────────────────────────────────
@app.route('/analyze', methods=['POST'])
def analyze():
    try:
        data = request.get_json(force=True)
        # Validate required features
        missing = [f for f in FEATURES if f not in data]
        if missing:
            return jsonify({"error": f"Missing features: {missing}"}), 400

        # Prepare DataFrame for inference
        input_df = pd.DataFrame([data], columns=FEATURES)

        # Perform anomaly detection
        pred  = model.predict(input_df)[0]
        score = model.decision_function(input_df)[0]

        return jsonify({
            "is_anomaly":    bool(pred == -1),
            "anomaly_score": round(float(score), 4),
            "details":       data
        })
    except Exception as e:
        return jsonify({"error": str(e)}), 400

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Figure 24 : Receipt anomaly detection

4. Generate Receipt Taxes:

```dart
List<Map<String, dynamic>> generateReceiptTaxes(List<dynamic> receiptItems) {
  Map<int, Map<String, dynamic>> taxGroups = {}; // Store tax summaries

  for (var item in receiptItems) {
    int taxID = item["taxID"];
    String taxPercent = item["taxPercent"];
    double total = item["total"];

    if (!taxGroups.containsKey(taxID)) {
      taxGroups[taxID] = {
        "taxID": taxID,
        "taxPercent": taxPercent.isEmpty ? "" : taxPercent, // Leave blank if empty
        "taxCode": item["taxCode"],
        "taxAmount": 0.0,
        "salesAmountWithTax": 0.0
      };
    }

    // Calculate tax amount
    //double taxAmount = taxPercent.isEmpty
      //  ? 0.00  // If taxPercent is empty, set taxAmount to 0.00
      // : total - double.parse((total / 1.15).toString());
    double taxAmount;
    if(taxPercent.isEmpty){
      taxAmount = 0.00;
    }
    else if(taxPercent=="15.00"){
      taxAmount = total - double.parse((total / 1.15).toString());
    }
    else{
      taxAmount = total * 0;
    }
    taxGroups[taxID]!["taxAmount"] += taxAmount;
    taxGroups[taxID]!["salesAmountWithTax"] += total;
  }

  // Convert map to list and round values
  // return taxGroups.values.map((tax) {
  //   return {
  //     "taxID": tax["taxID"],
  //     "taxPercent": tax["taxPercent"],  // Blank if empty
  //     "taxCode": tax["taxCode"],
  //     "taxAmount": tax["taxAmount"].toStringAsFixed(2), // Rounded to 2 decimal places
  //     "salesAmountWithTax": tax["salesAmountWithTax"],
  //   };
  // }).toList();
  return taxGroups.values.map((tax) {
    final taxID = tax["taxID"];
    final taxCode = tax["taxCode"];
    final isGroupA = (taxCode == "A" || taxID == 1);

    return {
      "taxID": taxID.toString(),
      if (!isGroupA) "taxPercent": tax["taxPercent"], // Omit if group A
      "taxCode": taxCode,
      "taxAmount": isGroupA ? "0" : tax["taxAmount"].toStringAsFixed(2),
      "salesAmountWithTax": tax["salesAmountWithTax"],
    };
  }).toList();
}
```

Figure 25 : Generate receipt taxes

5. Generate tax summary

```dart
String generateTaxSummary(List<dynamic> receiptItems) {
  Map<int, Map<String, dynamic>> taxGroups = {};

  for (var item in receiptItems) {
    int taxID = item["taxID"];
    double total = item["total"];
    String taxCode = item["taxCode"];

    // Preserve empty taxPercent when missing
    String? taxPercentValue = item["taxPercent"];
    double taxPercent = (taxPercentValue == null || taxPercentValue == "")
        ? 0.0
        : double.parse(taxPercentValue);

    if (!taxGroups.containsKey(taxID)) {
      taxGroups[taxID] = {
        "taxCode": taxCode,
        "taxPercent": taxPercentValue == null || taxPercentValue == ""
            ? 0
            : (taxPercent % 1 == 0
                ? "${taxPercent.toInt()}.00"
                : taxPercent.toStringAsFixed(2)),
        "taxAmount": 0.0,
        "salesAmountWithTax": 0.0
      };
    }
    double taxAmount ;
    if(taxPercentValue=="15.00"){
      taxAmount = total - double.parse((total / 1.15).toString());
    }else{
      taxAmount = total * 0;
    }
    taxGroups[taxID]!["taxAmount"] += taxAmount;
    taxGroups[taxID]!["salesAmountWithTax"] += total;
  }

  List<Map<String, dynamic>> sortedTaxes = taxGroups.values.toList()
    ..sort((a, b) => a["taxCode"].compareTo(b["taxCode"]));

  // return sortedTaxes.map((tax) {
  //   return "${tax["taxCode"]}${tax["taxPercent"]}${(tax["taxAmount"] * 100).round().toString()}${(tax["salesAmountWithTax"] * 100).round().toString()}";
  // }).join("");
  return sortedTaxes.map((tax) {
    final taxCode = tax["taxCode"];
    final taxPercent = tax["taxPercent"];
    final taxAmount = (tax["taxAmount"] * 100).round().toString();
    final salesAmount = (tax["salesAmountWithTax"] * 100).round().toString();

    // Omit taxPercent for taxCode A
    if (taxCode == "A") {
      return "$taxCode$taxAmount$salesAmount";
    }

    return "$taxCode$taxPercent$taxAmount$salesAmount";
  }).join("");
}
```

Figure 26 : Generate tax summary

6. Generate Signed Invoice

```dart
Future<void> generateInvoiceFromJson(Map<String , dynamic> receiptJson , String qrUrl) async{
  final receipt = receiptJson['receipt'];
  final supplier = {
    'name': 'Pulse Pvt Ltd',
    'tin': '1234567890',
    'address': '16 Ganges Road, Harare',
    'phone': '+263 77 14172798',
  };
  final customer = {
    'name': receipt['buyerData']?['buyerTradeName'] ?? 'Customer',
    'tin': receipt['buyerData']?['buyerTIN'] ?? '0000000000',
    'vat': receipt['buyerData']?['VATNumber'] ?? '00000000',
  };
  String receiptGlobalNo = receipt['receiptGlobalNo'].toString().padLeft(10, '0');
  String deviceID = "22162";
  final receiptLines = List<Map<String, dynamic>>.from(receipt['receiptLines']);
  final receiptTaxes = List<Map<String, dynamic>>.from(receipt['receiptTaxes']);
  final receiptTotal = double.tryParse(receipt['receiptTotal'].toString()) ?? 0.0;
  final signature = receipt['receiptDeviceSignature']?['signature'] ?? 'No Signature';
  double totalTax = 0.0;
  for (var tax in receiptTaxes) {
    totalTax += double.tryParse(tax['taxAmount'].toString()) ?? 0.0;
  }
  pdf.addPage(
    pw.Page(
      pageFormat: PdfPageFormat.a4,
      margin: const pw.EdgeInsets.all(24),
      build: (pw.Context context) {
        return pw.Column(
          crossAxisAlignment: pw.CrossAxisAlignment.start,
          children: [
            pw.Row(
              mainAxisAlignment: pw.MainAxisAlignment.spaceBetween,
              children: [
                pw.Column(
                  crossAxisAlignment: pw.CrossAxisAlignment.start,
                  children: [
                    pw.Text('FISCAL TAX INVOICE', style: pw.TextStyle(fontSize: 24, fontWeight: pw.FontWeight.bold)),
                    pw.SizedBox(height: 8),
                    pw.Text('Supplier: ${supplier['name']}'),
                    pw.Text('TIN: ${supplier['tin']}'),
                    pw.Text('Address: ${supplier['address']}'),
                    pw.Text('Phone: ${supplier['phone']}'),
                  ],
                ), // pw.Column
                pw.Container(
                  width: 100,
                  height: 100,
                  child: pw.BarcodeWidget(
                    barcode: pw.Barcode.qrCode(),
                    data: qrUrl,
                    drawText: false,
                  ), // pw.BarcodeWidget
                ), // pw.Container
              ],
            ), // pw.Row
```

```dart
            pw.Divider(
              thickness: 5,
              color: PdfColors.blue,
            ), // pw.Divider
            pw.SizedBox(height: 8),
            pw.Text('Customer: ${customer['name']}'),
            pw.Text('TIN: ${customer['tin']}'),
            pw.Text('VAT: ${customer['vat']}'),
            pw.SizedBox(height: 12),
            pw.Text('Invoice No: ${receipt['invoiceNo']}', style: pw.TextStyle(fontSize: 14)),
            pw.Text('Date: ${receipt['receiptDate']}'),
            pw.SizedBox(height: 12),
            pw.Table.fromTextArray(
              headers: ['No.', 'Item', 'Qty', 'Unit Price', 'Tax %', 'Total'],
              data: receiptLines.map((item) {
                return [
                  item['receiptLineNo'],
                  item['receiptLineName'],
                  item['receiptLineQuantity'].toString(),
                  '\$${item['receiptLinePrice'].toString()}',
                  '${item['taxPercent'] ?? '0'}%',
                  '\$${item['receiptLineTotal'].toString()}',
                ];
              }).toList(),
            ), // pw.Table.fromTextArray
            pw.Divider(),
            pw.SizedBox(height: 10),
            pw.Align(
              alignment: pw.Alignment.centerRight,
              child: pw.Column(
                crossAxisAlignment: pw.CrossAxisAlignment.end,
                children: [
                  pw.Text('Total Tax: \$${totalTax.toStringAsFixed(2)}'),
                  pw.Text('Grand Total: \$${receiptTotal.toStringAsFixed(2)}', style: pw.TextStyle(fontWeight: pw.FontWeight.bold)),
                ],
              ), // pw.Column
            ), // pw.Align
            pw.SizedBox(height: 20),
            pw.Text('Signature Hash:', style: pw.TextStyle(fontSize: 10)),
            pw.Text(receipt['receiptDeviceSignature']?['hash'] ?? '', style: pw.TextStyle(fontSize: 8)),
            pw.Text("You can verify this manually at:", style: pw.TextStyle(fontSize: 10)),
            pw.Text("https://fdmstest.zimra.co.zw", style: pw.TextStyle(fontSize: 8, color: PdfColors.blue)),
            pw.Text("Device ID: $deviceID", style: pw.TextStyle(fontSize: 10)),
            pw.Text("Receipt Global No: $receiptGlobalNo", style: pw.TextStyle(fontSize: 10)),
          ],
        ); // pw.Column
      },
    ), // pw.Page
  );
  try {
    final directory = Directory('/storage/emulated/0/Pulse/Invoices');
```

```
    // Create the directory if it doesn't exist
    if (!await directory.exists()) {
      await directory.create(recursive: true);
    }
    final filePath = p.join(directory.path, 'invoice_${receipt['invoiceNo']}.pdf');
    final file = File(filePath);
    await file.writeAsBytes(await pdf.save());

    print('Invoice saved at ${file.path}');
  } catch (e) {
    print('Error saving invoice: $e');
  }
}
```

Figure 27 : Generate QR coded invoice

5.2 Software Testing

To ensure the reliability and performance of the mobile fiscalization application, comprehensive testing was conducted at various levels:

5.3.1 Unit Testing

The objective here is to test individual functions and methods.

Test case ID	Function	Test Objective	Input	Expected output	Status
UT001	validateLogin()	Ensure valid credentials are accepted	Valid username and password	Login success	Pass
UT002	calculateVat()	Correct VAT calculation	Amount: $100, VAT:15%	$15	Pass
UT003	detectAnomaly()	Identify duplicate receipt	Receipt ID already exists	Anomaly flag raised	Pass

Table 14 : Unit testing

5.3.2 Module Testing

Here were testing each functional modules separately.

Test case ID	Function	Test Objective	Input	Expected output	Status
MT001	Login Module	Authenticate user	Valid credentials	Redirect to dashboard	Pass
MT002	Receipt Module	Generate fiscal receipt	Item details + tax info	Fiscal receipt with QR code	Pas
MT003	Reporting Module	Generate sales report	Date range input	Summary report	Pass

Table 15 : Modular testing

5.3.3 Integration Testing

Test complete flows between interconnected modules.

Test case ID	Function	Test Objective	Input	Expected output	Status
IT001	Login + menu	Post-login redirection	Successful login	Menu loads	Pass
IT002	Receipt + PDF	Receipt to invoice PDF	Generate PDF from receipt	Valid PDF generated	Pass
IT003	Receipt + Anomaly Detection	Analyze new receipt	Add suspicious receipt	Flag anomaly	Pass/Fail

Table 16 : Integration testing

5.3.4 System Testing

Test the whole system as a single entity.

Test case ID	Function	Test Objective	Input	Expected output	Status
ST001	Full transaction cycle	Login > Create Receipt > Generate PDF > View Reports	All steps complete successfully	Pass	Full transaction cycle
ST002	Admin tasks	Add user > Update settings	All changes saved and active	Pass	Admin tasks

Table 17 : System testing

5.3.5 Database Testing

This test ensures that data is correctly stored and retrieved.

Test case ID	Test Objective	Test Description	Expected outcome	Status
DB001	Data persistence	Save new receipt	Receipt stored correctly	Pass
DB002	Integrity check	Delete user with linked receipts	Prevent deletion, show warning	Pass

Table 18 : Database testing

5.3.6 Acceptance Testing

Validate the system with real users based on requirements.

Test case ID	Requirement	Test Objective	expected outcome	Status
AT001	Mobile access	User accesses system from phone	Interface adapts correctly	Pass
AT002	Anomaly alerts	System flags tampered receipts	Alert is triggered, report shown	Pass
AT003	Tax compliance	Generate ZIMRA-compliant receipt	Receipt includes fiscal QR & signature	Pass

Table 19 : Acceptance testing

Result:

Acceptance testing confirmed that the system meets user expectations for offline fiscalization, reporting, and data reliability.

Chapter Six: Conclusions and Recommendations

6.1 Results and Summary

The primary objective of this project was to develop a **mobile fiscalization application** aimed at addressing the limitations of traditional Electronic Cash Register (ECR) systems, particularly in areas with poor network connectivity. The application successfully enables users to generate **fiscalized receipts and invoices**, even while offline, ensuring continuous compliance with tax regulations.

Key achievements include:

- **Mobile Fiscal Receipt and Invoice Generation**: Users can create fiscal-compliant documents, stored securely on the device and synchronized once a network connection is available.
- **Offline Capability**: The system functions efficiently in offline mode, storing transaction data locally until internet access is restored.
- **Receipt Anomaly Detection**: Built-in lightweight machine learning models allow the detection of suspicious or erroneous receipt data.
- **Sales and Tax Analytics**: The app provides users with insights into their sales performance and future tax obligations using predictive analytics.
- **User-Friendly Interface**: The mobile application offers an intuitive, easy-to-use design to accommodate users with varying technical skills.
- **Security and Compliance**: Data encryption and digital signatures are incorporated to ensure authenticity, integrity, and confidentiality of fiscal data.

The results demonstrate that mobile fiscalization can significantly enhance tax compliance, business reporting, and user accessibility, especially for businesses operating in rural or network-challenged environments.

6.1.2 Achievements of objectives

This mobile based fiscalization system successfully achieved all its stated objectives:

- Achieve 100% successful connection to ZIMRA's API gateway and process at least 98% of fiscalization requests
- Ensure 100% of generated invoices are encrypted and digitally signed using an approved cryptographic method, with validation logs showing zero unsigned invoices during monthly audits.
- Implement an anomaly detection module that identifies fraudulent or erroneous invoices with a false positive rate
- Develop a reporting dashboard that provides users with real-time insights covering 100% of submitted invoices, including return summaries, and compliance status, with reports available monthly and on demand.

6.2 Recommendations

While the application meets the fundamental objectives, several areas have been identified for enhancement:

- **Expand Machine Learning Models**: Future iterations could incorporate more sophisticated AI techniques (e.g., Deep Learning models) to improve the accuracy and scope of anomaly detection.
- **Multi-Device Synchronization**: Implement cloud backup and device synchronization features for businesses operating across multiple mobile devices.
- **Localization and Multi-language Support**: Translate the application into multiple local languages to widen adoption across different regions.
- **Enhanced Reporting Modules**: Develop customizable reporting tools allowing businesses to generate specific reports such as VAT returns, expense tracking, and profit/loss analysis.
- **Payment Gateway Integration**: Integrating the POS with payment gateways to easily cater for online and card payments. This introduces a wide range of seamless payment methods
- **AI based sales trend analysis**: Incorporate some artificial intelligence algorithm to forecast some sales trends and potentials. This would also help users in future planning.

- Introduce biometrics authentication when accessing the system to reduce the chances of unauthorized access to the system.
- **Cloud Sync and Backup**: Cloud storage of some system data would help in introducing functionalities like multi-location access to the same data, ensuring consistency when the same system is used in different locations.

6.3 Future Works

To further advance the mobile fiscalization solution and extend its impact, the following future works are proposed:

1. **Block chain based Fiscalization**: Explore the use of block chain technology to enhance the transparency and immutability of fiscal transaction records.
2. **Integration with Mobile Payment Platforms**: Allow seamless payment and receipt generation through popular mobile money services such as Eco Cash, One Money, and Tele Cash.
3. **Web Dashboard for Business Owners**: Create a web-based platform where businesses can manage their receipts, view analytics, and download reports from any device.
4. **Intelligent Tax Advisory Bot**: Incorporate a chat bot powered by AI that provides users with real-time advice on tax filing, compliance, and sales strategies.
5. **Partnerships with Tax Authorities**: Collaborate with ZIMRA and other revenue authorities to standardize and promote mobile fiscalization across industries.

Bibliography

1. OECD. (2015). *Tax Administration 2015: Comparative Information on OECD and Other Advanced and Emerging Economies.*

2. Richard Bird and Eric M. Zolt 2008, *Technology and Taxation in Developing Countries: From Hand to Mouse*

3. Bajak, A., & Majkic, Z. (2020). *The Evolution of Fiscalization: A Comparative Analysis*

4. Benussi, S., Scarpi, D., & Fabbri, M. (2016). *Secure Electronic Fiscal Systems: From Hardware to Software Solutions*. Tax Technology Review, 8(1), 44-59.

5. Di Matteo, M., & Piatti, G. (2013). Fiscal Devices and Tax Compliance: Lessons from Europe. OECD Working Papers

6. World Bank. (2020). *Digital Financial Services and the Informal Economy*.

7. Hungarian Tax Authority. (2018). *QR Code Implementation Guide for Fiscal Receipts*

8. Mikuš, M., & Škvorc, J. (2014). *Real-Time Fiscalization of Cash Transactions in Croatia*. Croatian Tax Review, 18(4), 37-50.

9. ZIMRA. (2023). *Fiscalization Guidelines for Mobile Applications*

10. KRA. (2022). Kenya Revenue Authority Annual Report.

11. Li et al-2022-Nature

Appendices

Appendix A: User manual

PulsePay Fiscalization Mobile Application – User Guide

Login:

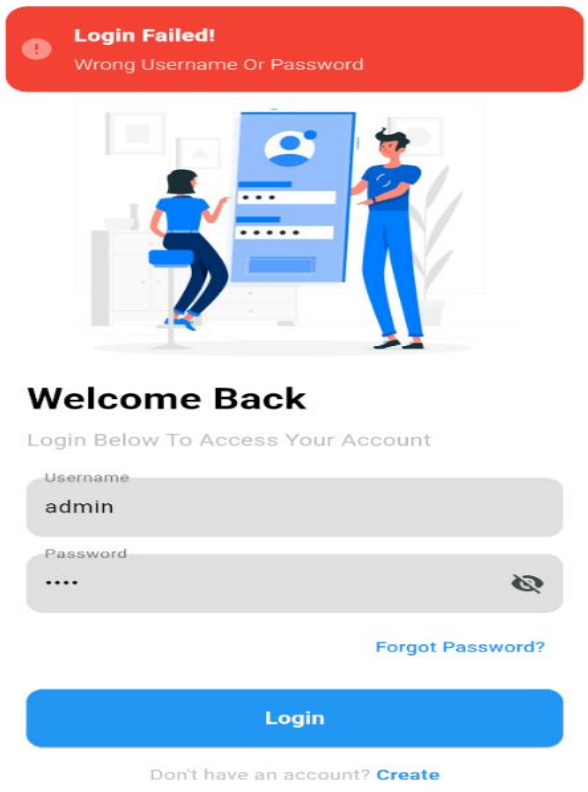


Figure 28 : Login

- Enter valid credentials to access the system.

Figure 29 : Main menu

- Upon accessing the system navigate to the Fiscalization page to open a fiscal day

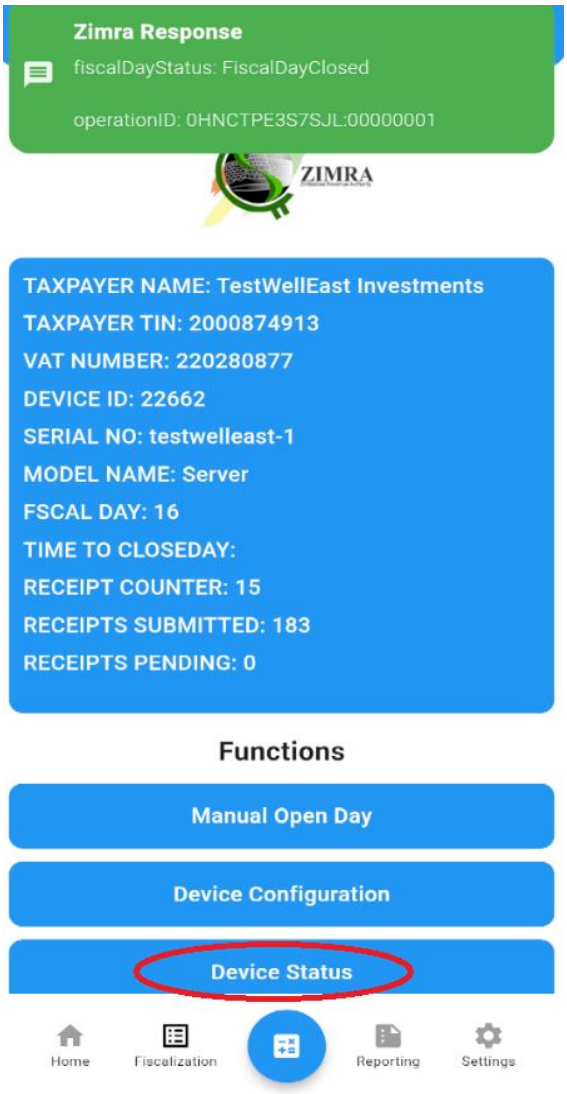


Figure 30 : fiscal page

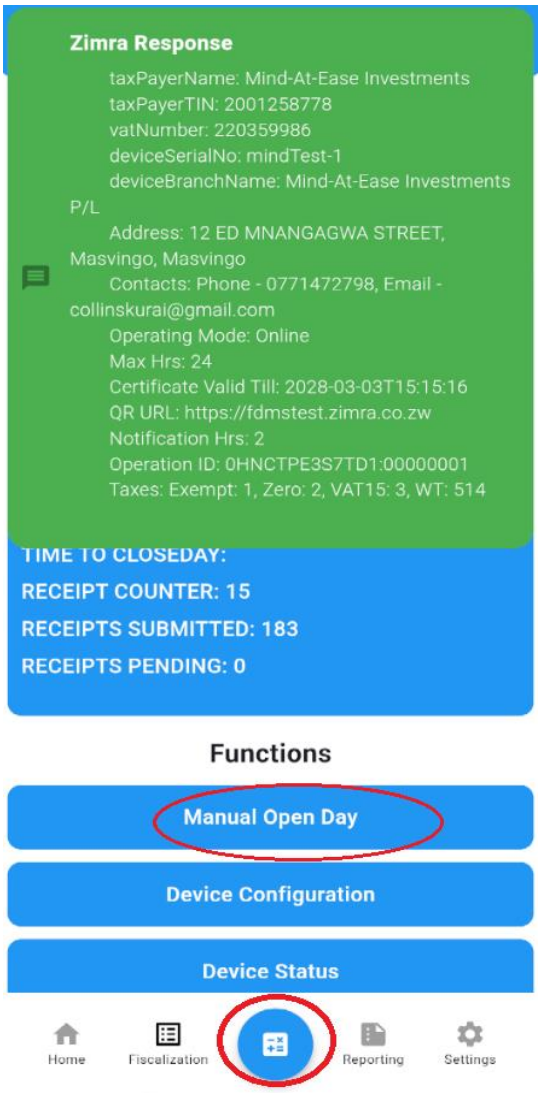


Figure 31 : fiscal page

- Upon getting to the fiscal page ,click device status to check if fiscal day is closed

- To open a fiscal day click on manual open day and successful opening of a day will show the configuration details.
- After that click the centre floating button to open the POS terminal

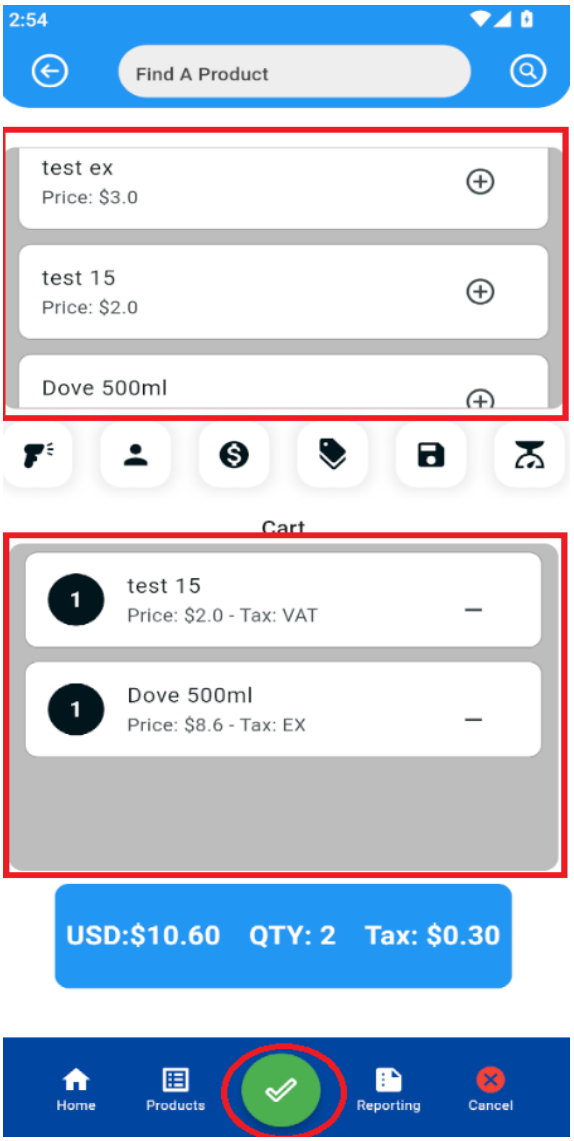


Figure 32 : POS terminal

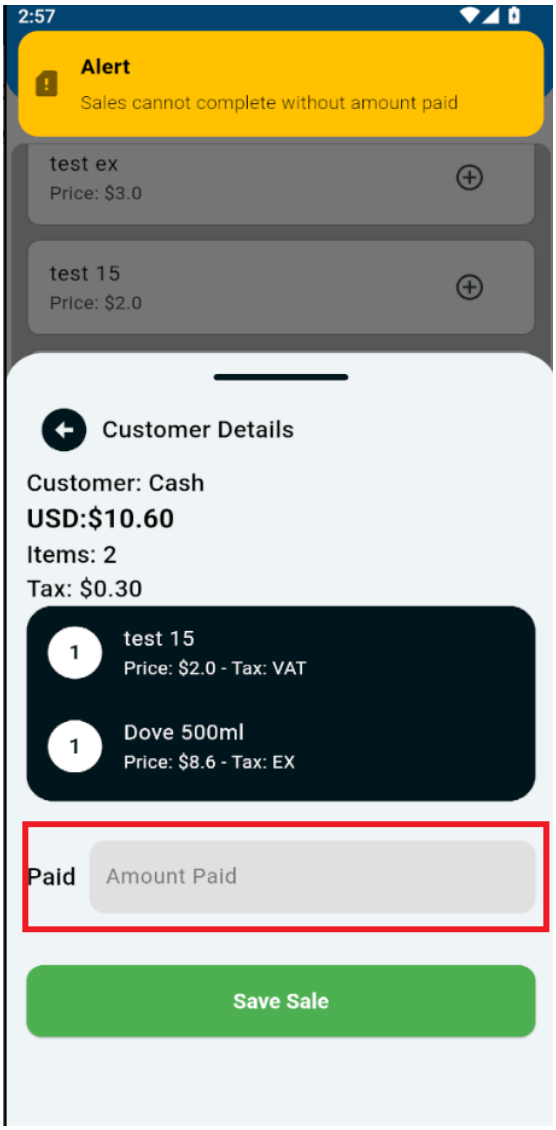


Figure 33 : Payment

- To find a product or service ,go to search product and click the search button. The products will show in the container below the search bar
- Upon selecting a product it will be moved to the cart below
- When done selecting the products, click the green centre floating button to display the payment bar

- Enter the amount to be paid in the amount paid textbox

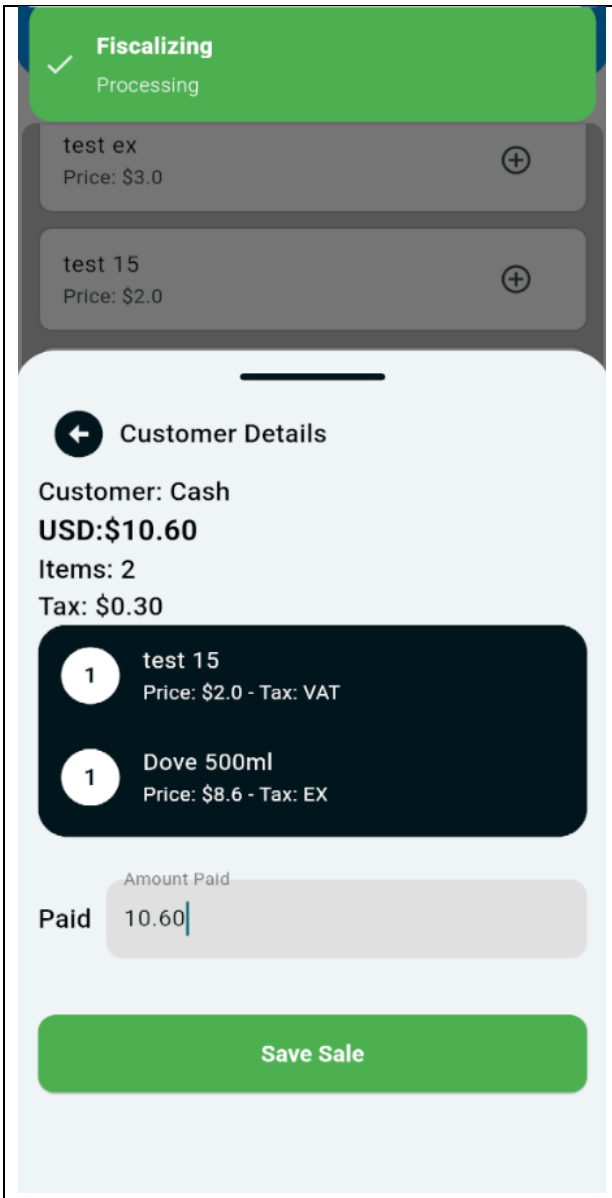


Figure 34 : Payment entry

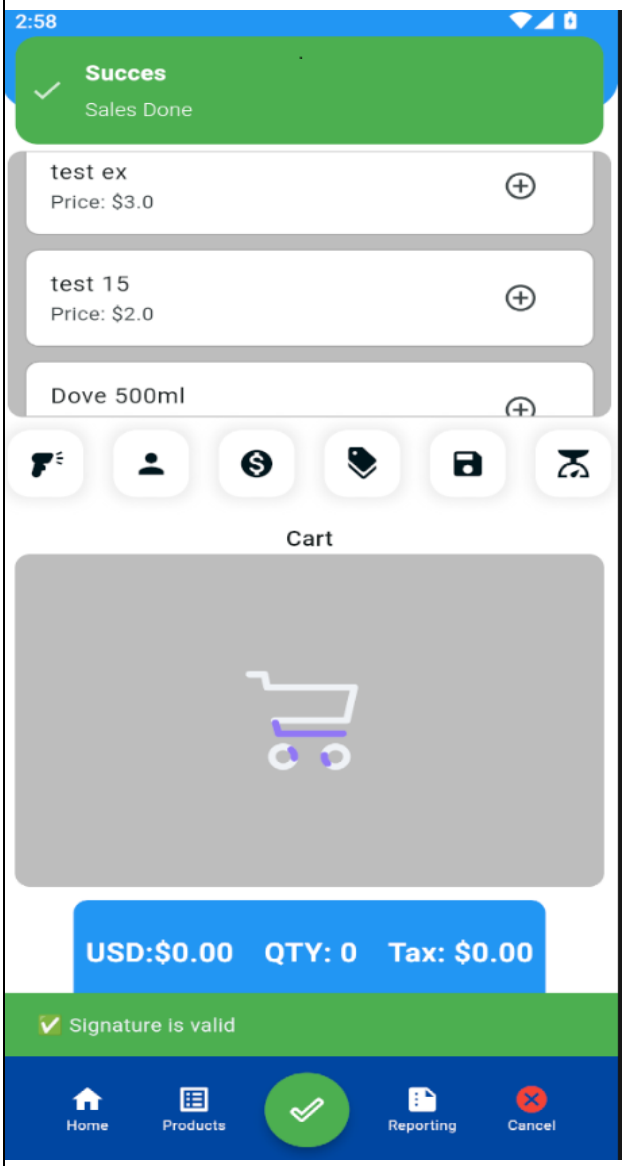


Figure 35 : Sale done

- Click save sale and the fiscalization process will start

- The application takes you back to the selling terminal

FISCAL TAX INVOICE

Supplier: Pulse Pvt Ltd
TIN: 1234567890
Address: 16 Ganges Road, Harare
Phone: +263 77 14172798

Customer: Cash Sale
TIN: 0000000000
VAT: 123456789

Invoice No: 174
Date: 2025-06-02T19:51:00

No.	Item	Qty	Unit Price	Tax %	Total
1	test ex	1	$3.00	0%	$3.00
2	Dove 500ml	1	$8.60	0%	$8.60
3	test 15	1	$2.00	15.00%	$2.00

Total Tax: $0.26
Grand Total: $13.60

Signature Hash:
You can verify this manually at:
Device ID: 22162
Receipt Global No: 0000000182

Figure 36 : Fiscal tax invoice

- To get a copy of your fiscal invoice, navigate to files in local storage , go the to Pulse folder and choose the invoices folder to get all the fiscalized invoices

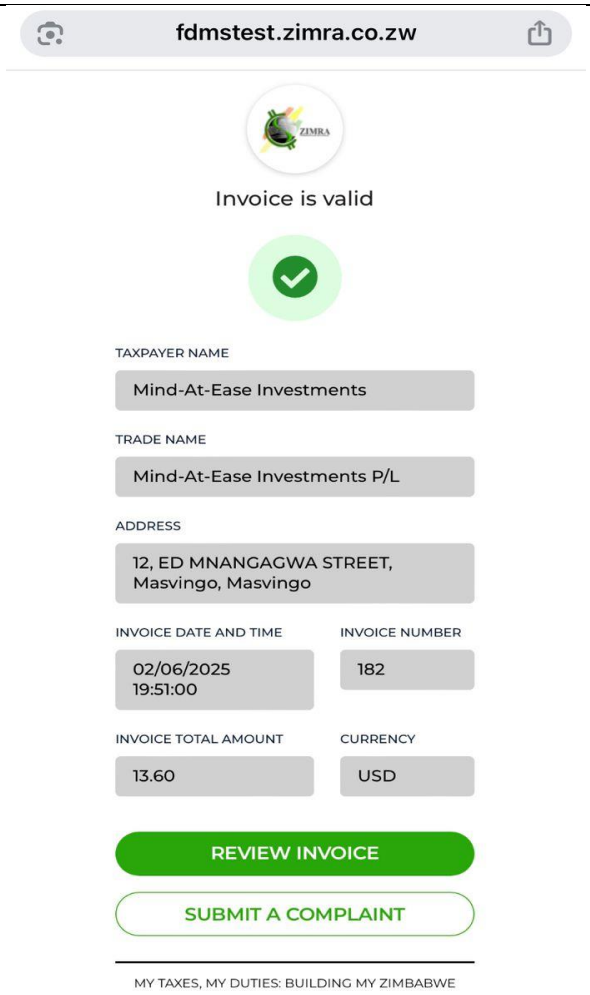


Figure 37 : Invoice validation portal

- To validate the invoice, scan the qrcode on the invoice that will lead you to this ZIMRA webpage that shows whether your invoice is valid or not

APPENDIX B:

Information Gathering Tools - Questionnaire

Pulse Pay Fiscalization System - Information Gathering Questionnaire

This questionnaire is designed to collect insights regarding the need for a mobile application in the Pulse Pay Fiscalization System. Your feedback will help us assess why current users may be moving away from traditional electronic cash registers and fiscal printers.

Please feel free to fill in your details below:

Full name: _____

Company: _____

Role : _____

Date : _____

Questionnaire

1. Do you currently use an electronic cash register or fiscal printer? If yes, which one?
   Yes ☐ No ☐

   _____
   _____
   _____

2. What challenges have you experienced with your current fiscalization method?
   _____
   _____
   _____
   _____

3. Would a mobile-based solution simplify your daily business operations? Why or why not?
   Yes ☐ No ☐

   -
   _____
   _____
   _____

4. How important is portability and mobility in your sales and invoicing processes?
   _____
   _____
   _____

5. What features would you expect in a mobile fiscalization application?

6. Have you faced any issues with hardware maintenance or connectivity with your current device?

7. Would you prefer a system that allows digital receipts and QR code generation on the go?

8. How confident are you in using smartphone applications for business operations?

9. What security concerns do you have regarding fiscal mobile apps?

10. Why do you think businesses may want to shift from traditional fiscal printers to mobile systems?

PulsePay – Fiscalization Mobile App

Collins K Chimbganda; Mr. M Mukosera

Department of Software Engineering, School of Information Sciences and Technology Harare Institute of Technology, Harare, Zimbabwe

collinskurai@gmail.com; mmukosera@hit.ac.zw

Abstract

This study describes a mobile based fiscalization system used by tax collectors in Zimbabwe to generate fiscal invoices to ensure tax compliance to ZIMRA. The technology attempts to improve the tax collection process by making it accurate and simpler for tax collectors to send their taxation records to the authority , making it clear for them to visualize the amounts collected and get an overview of their tax performance. The suggested system has these stages:

- **Invoice generation**
- **Data Encryption and digital signature generation**
- **Payload submission and response pulling**
- **Fiscal QR Code generation**
- **Receipt Anomaly detection**

The system transforms the users' invoice data into the required JSON payload format. Creates receipt string with a summary of the receipts taxes and invoice data from which a digital signature and hash are generated using RSA keys. The signature and hash are sent along with the receipt JSON payload to Zimra. An md5 hash version of the generated signature is then used to come up with a URL used so invoice validation and invoice QR Code generated. Anomaly detection measures are put on the system to detected erroneous receipts and potential cases of fraud. This system has the potential to be widely adopted by many tax payers in various working environment because of its simple usability, diversity and its ability to do some useful tax reporting.

Keywords: FDMS (Fiscal Data Management System), Fiscalization, Digital Signature, Encryption, Anomaly.

I. INTRODUCTION

In the digital age, fiscalization has become a fundamental part in ensuring compliance with tax regulations. Traditionally, businesses in Zimbabwe have relied heavily on Electronic Cash Registers (ECRs) and fiscalized printers for issuing fiscalized receipts and invoices. However, these are now face significant challenges with the introduction of ZIMRA's FDMS (Fiscal Data Management System), particularly in remote and low-network areas, where connectivity problems are disrupting the fiscalization processes.

With the growth of mobile technology, there is a unique opportunity to overcome these challenges. This project proposes a mobile-based fiscalization application that allows users to generate fiscalized receipts and invoices even in areas with poor network connectivity. Beyond receipt generation, the mobile app also offers detailed tax reporting, receipt anomaly detection, and advanced data analytics for sales and tax predictions — equipping businesses with intelligent tools for compliance and operational insights.

The objectives of this paper are threefold:

- To successfully connect to Zimra's API gateway and handle all fiscalization processes and requests
- To encrypt invoice data ,putting digital signature on every generated invoice
- To detect anomalies in receipt to reduce erroneous invoices and invoice fraud
- To analyse invoices' data to give users full insights on tax performance and returns

The suggested system has the potential to boost tax compliance among VAT registered companies in

Zimbabwe, overall improving the countries year tax collections.

II. PROBLEM STATEMENT

As from January 2024, TARMS and FDMS from ZIMRA replaced the already existing ZIMRA tax collection systems with a new one that required the upgrading of already existing tax collection devices and systems to match up with the current implementation. Interaction with FDMS is heavily dependent on stable internet connection to perform the fiscalization tasks and these upgrades on devices that are almost obsolete are posing agonizing network challenges and unexpected device failures. These challenge are not only affecting the tax collection authorities but also inconveniences businesses and customers alike. Therefore, there is an unforeseen need for a mobile fiscalization solution that is not only cheap but can operate reliably given the nation's current conditions while also offering intelligent reporting and anomaly detection features

III. RELATED WORK

The term fiscalization refers to the implementation of systems and processes that ensure accurate reporting and monitoring of financial transactions with the main goal set to ensure proper tax compliance. This tax compliance had been enforced through hardware based fiscal devices and has since evolved alongside technological advancements to include software based and hybrid based systems [1].
Combating tax evasion and enhancing revenue collection are the main objectives of fiscalization, especially in cash-heavy industries like retail and hospitality [2]. Fiscalization has been implemented in a variety of ways by different countries. For example, Sweden and Kenya have shifted to software-driven solutions that are integrated with their national tax systems, whereas others like Italy and Serbia have depended on fiscal cash registers (FCRs) [10].

Fiscalization Technologies and Standards

According to [5], fiscalization has historically necessitated the deployment of physical fiscal memory devices that safely held transaction data and made manipulation difficult. These device were ranging from electronic cash registers, electronic signature devices to fiscalized printers that were being distributed by some companies like Eltrade. With recent development in fiscal and tax regulatory systems, these device have since become obsolete and seen to be causing a lot of challenges when it comes to being compatible with these modern day tax systems. More scalability and flexibility are provided by more recent systems, which make use of cloud computing, encryption, and real-time data transfer [3]. To advance the digitization of fiscal data, the OECD created the framework Audit File for Tax (SAF-T), which establishes a framework for the electronic interchange of trustworthy accounting data [1]. In countries like Zimbabwe there has been a rise in the use of virtual fiscalization systems that are working alongside accounting packages to ensure that tax collectors are able to issue out fiscal invoices.

Key technological components of modern fiscal systems include:

- Encryption and digital signatures: Ensuring data integrity and authenticity [4].
- QR codes: Common in countries like Zimbabwe and Hungary, these provide verifiable links to tax-registered invoices [9] [7].
- APIs for real-time reporting: As seen in Croatia's fiscalization model, real-time APIs connect point-of-sale systems directly to tax authorities [8].

Fiscalization in mobile applications

Fiscalization has spread into the mobile application area as a result of the growing usage of smartphones and mobile point-of-sale (mPOS) systems. Businesses in the informal sector and small and medium-sized enterprises (SMEs) can join formal economies with lower overheads because to mobile fiscalization [7]. The field has rapidly advanced in recent years as we seen improvements and innovations ensuring that compliance can be ensured at ease with the use of mobile devices.
Mobile-compatible fiscalization, which has been adopted by nations like Zimbabwe, enables retailers to issue and fiscalize invoices through mobile applications that connect to national tax authority systems through secure API calls [9]. In some countries like Kenya and Nigeria, real-time mobile tax compliance has been in use for some years now. Kenya's iTax and Nigeria's eTax leverage mobile platforms to streamline VAT reporting and enhance compliance in the countries [10]. Many mobile fiscal apps in different countries now include an automated QR code generation feature, which contains fiscal data like tax identification , invoice totals and digital signatures , ensuring easy verification by customers and authorities.

Despite the promise in mobile fiscalization there are some challenges that remain and can be foreseen with the implementation of such systems. A main issue lies in device and network limitations. In regions with limited

internet infrastructure, real time fiscalization be difficult to implement according to 6[]. Modern mobile applications also required mobile devices with better architecture meaning that acquiring a compatible mobile device could be unnecessarily expensive to some individuals or startups. In some regions it has been seen that ensuring mobile solutions comply with diverse fiscal regulations across jurisdictions requires an adaptive design [5].
Conversely, mobile fiscalization offers transformative opportunities for **financial inclusion** and **enhanced tax collection**, particularly in developing economies.

The development of fiscalization is indicative of more general patterns in tax compliance and digital governance. Modern fiscalization makes greater use of cloud services, APIs, and mobile technologies than did traditional systems, which were dependent on physical hardware. Particularly in developing nations, mobile fiscalization has a great deal of promise for expanding formal economic involvement to SMEs and unofficial enterprises. Future studies can concentrate on investigating machine learning techniques for automated tax fraud detection and improving security frameworks for mobile fiscal systems.

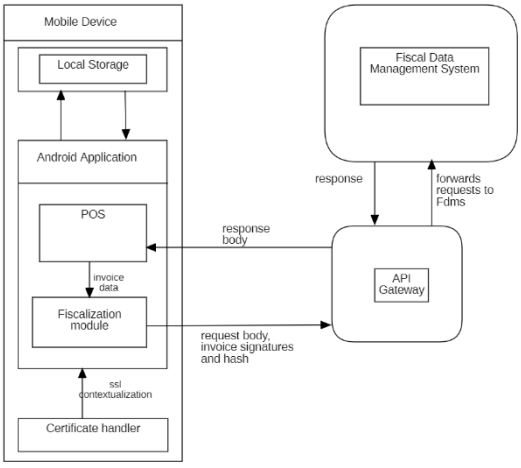
IV. SOLUTION

The mobile fiscalized system is there to revolutionize and simplify the fiscalization process. It a more cheaper and convenient way to issues out fiscal invoices. The users will also benefit from the other inbuilt features of the system such as thorough reporting, fraud detection and inventory management.
However, careful assessment of the problems and prudent implementation are required to maximize its benefits while minimizing any negatives.

A. Features of the system

- **SSL Contextualization**: This is core function as it is initialized first before interacting with the API gateway for any of the included functions. It ensures that all payment data is encrypted in transit, verifies that API calls between the application and the fiscal server are secured with SSL and makes sure that certificates are valid, current and trusted.
- Fiscalization: this where transaction data is analyzed and tax calculations are done and Ids are identified. The invoice metadata is prepared and a signature is generated. In the process the metadata is also used in the generation of URL and a QR code that can be used for invoice verification.
- Anomaly detection: the anomaly detection module goes through all submitted receipt and tries to pick up potential anomalies that can lead to invoice verification failure or tax fraud.
- Reporting and analysis: this feature of the Pulse Pay provides users with comprehensive insights into their sales and tax data. It generates detailed reports on transactions, including total sales, tax breakdowns, and receipt summaries.
- Inventory Management: enables businesses to track and manage their stock levels efficiently. It keeps real-time records of product quantities, sales, and restocking needs, ensuring that inventory is always up to date. The system automatically updates stock when sales are made and alerts users when items reach low stock thresholds.
- Invoicing: this feature simplifies the creation and management of customer invoices. Users can generate fiscalized invoices directly from the mobile POS, complete with tax calculations and QR codes for verification. Each invoice is automatically recorded in the system, linking to sales transactions and customer data

B. Solution Architecture



1. **Sales and Invoicing Module**: Responsible for ensuring that users are presented with the best layout to carry out the sale.

Captures various sale and invoice data to be used in various sections of the system.

2. **Fiscalization Module**: This module is responsible for transforming the invoice/sale data into required formats of the Json Payloads. It ensures that receipts are submitted to the Zimra servers, invoice data recorded and ensures the generation of the verification QR codes.
3. **Security Module**: This component manages user authentication, access control, SSL contextualization and Key encryption to generate the required digital signatures and hashes.
4. **Reporting Module**: focuses on generating numerous system reports to be used by the users for analysis and summarizing purposes. It ensures that users get insights on their sales and tax totals
5. **Database Interface**: the system has a single database storing all the system data in a number of structured tables.

C. Technology Stack

- Mobile : Android, Flutter , Kotlin
- Backend : Python
- Database : SQFlite

D. Methodology

The system was developed using the agile development method to ensure rapid development and integration of modules. This also allowed for incremental delivery of features and iterative refinement of the system based on feedback. Development was mainly centered on these areas:

- User Interface and User Experience
- Fiscal data handling and encryption
- Receipt anomaly detection
- Secure storage and retrieval of invoice information

E. Workflow

- Transaction Initiation
- **Data Capture and fiscalization process:** upon completion , transaction data is encrypted using the merchant's PKCS 12 private key , it is packaged into a fiscal receipt format compliant with national tax standards and transmitted to the authority's API endpoint
- **QR Code generation:** A QR code containing fiscal metadata (signature, receipt number, timestamp) is generated and attached to the receipt.
- **Anomaly Detection:** an anomaly detection algorithm runs in the background going through the submitted receipt checking for receipts with potential errors and abnormalities.
- **Error handling and feedback:** In case of errors (e.g., network failure or fiscal API rejection), the system queues the transaction for retry while notifying the user with actionable error messages.

V. RESULTS AND FUTURE WORKS

A. Results

The system worked so well in allowing the users to complete a sale and issue out a fiscalized invoice that was successfully validated by the revenue authorities on the test server.

1. **Accuracy**: 99% of the invoices generated were calculated correctly.
2. **Precision**: 99% of the invoice were validated with the correct tax amounts and tax data
3. **Verification Rate**: 95% of invoices generated could be verified by the Zimra servers.
4. **Encryption Accuracy**: 97% of the invoices generated had a valid signature and hash
5. **System Response Time**: 3 seconds (average time for the system to process and send invoice tax data)

Objectives	Fully Achieved	Partially Achieved
To successfully connect to Zimra's API gateway and handle all fiscalization processes and requests	✓	
To encrypt invoice data ,putting digital signature on every generated invoice	✓	
To detect anomalies in receipt to reduce erroneous invoices and invoice fraud	✓	
To analyze invoices' data to give users full insights on tax performance and returns	✓	

Future Works

1. Enhanced Functionality:
   - Payment Gateway Integration: Integrating the POS with payment gateways to easily cater for online and card payments. This introduces a wide range of seamless payment methods
   - Accounting package integration: Linking up the system with other accounting packages like Pastel and QuickBooks for further accounting functionalities and financial reporting.
2. Advanced Analysis:
   - AI based sales trend analysis: Incorporate some artificial intelligence algorithm to forecast some sales trends and potentials. This would also help users in future planning.
3. Advanced Security Features:
   - Introduce biometrics authentication when accessing the system to reduce the chances of unauthorized access to the system.
4. Cloud Sync and Backup – Cloud storage of some system data would help in introducing functionalities like multi-location access to the same data, ensuring consistency when the same system is used in different locations.

If the future works are looked into and implemented the system becomes a more powerful and complete fiscalization and accounting tool. It'll include features that most of the tax collectors would love to be incorporated with their operations improving overall tax compliance in the country.

Conclusion

In conclusion, the proposed Pulse Pay, fiscalization system presents a promising solution for improving fiscal devices used by tax collectors. By overcoming challenges and focusing on future advancements, this has the potential to revolutionize the way fiscal invoices are issues out, ultimately contributing to better tax collection which means achieving of national tax targets

ACKNOWLEDGEMENT

To everyone who helped me finish this project successfully, I would like to extend my sincere gratitude and appreciation. I want to start by giving thanks to the Lord Almighty, whose grace and presence have been with me from the beginning of this program till now, when I am reaching a significant milestone.

I want to express my gratitude to Mr. Mukosera, my supervisor, for his helpful advice, knowledge, and unwavering support over the entire study process. This project has been much influenced and improved by his perceptive comments, helpful critiques, and support. Additionally, I am incredibly appreciative of the Software Engineering department's faculty members, whose guidance and instruction have given me a solid background in my field of study. Their enthusiasm for learning and commitment to teaching have been incredibly motivating.

I want to express my sincere gratitude to my mother for supporting me during the trip. For their constant encouragement and faith in me, I am also appreciative of my family and friends.

I would like to express my gratitude to the study participants, whose readiness to share their knowledge and perspectives has greatly influenced the research results. Their input has been crucial in determining how this dissertation has turned out.

Finally, I would want to sincerely thank everyone who has indirectly supported this research by their publications, academic work, and earlier research. The theoretical foundation and technique of this dissertation have been greatly influenced by their contributions. Finally, I would want to express my sincere gratitude to everyone who helped to complete this project, no matter how modest their contribution was. Your encouragement, advice, and support have been priceless, and I sincerely appreciate the chance to have embarked on this research adventure with your steadfast assistance.

REFERENCES

[1] OECD. (2015). *Tax Administration 2015: Comparative Information on OECD and Other Advanced and Emerging Economies.*

[2] Richard Bird and Eric M. Zolt 2008, *Technology and Taxation in Developing Countries: From Hand to Mouse*

[3] Bajak, A., & Majkic, Z. (2020). *The Evolution of Fiscalization: A Comparative Analysis.* Journal of Digital Taxation, 5(2), 112-130.

[4] Benussi, S., Scarpi, D., & Fabbri, M. (2016). *Secure Electronic Fiscal Systems: From Hardware to Software Solutions.* Tax Technology Review, 8(1), 44-59.

[5] Di Matteo, M., & Piatti, G. (2013). Fiscal Devices and Tax Compliance: Lessons from Europe. OECD Working Papers.

[6] World Bank. (2020). *Digital Financial Services and the Informal Economy*.

[7] Hungarian Tax Authority. (2018). QR Code Implementation Guide for Fiscal Receipts.

[8] Mikuš, M., & Škvorc, J. (2014). *Real-Time Fiscalization of Cash Transactions in Croatia.* Croatian Tax Review, 18(4), 37-50.

[9] ZIMRA. (2023). *Fiscalization Guidelines for Mobile Applications.*

[10] KRA. (2022). Kenya Revenue Authority Annual Report.