



NUKA PROJECT PROPOSAL

Adam Musakabantu Muyobo 202104060

Introduction

In today's business space, many Zambian enterprises are struggling with unreliable Point of Sale (POS) systems that are either overpriced, unreliable, lack necessary features and or fail to follow tax regulations. These inefficiencies lead to financial issues, double entry, and challenges in managing something important to many businesses SALES. Nuka, meaning "river" in Lozi, is designed to control the flow of transactions in a business, ensuring good sales processing, automated tax returns, and overall financial transparency.



By integrating with the Zambia Revenue Authority (ZRA) Smart Invoicing API, Nuka will allow businesses to generate tax-compliant invoices in real time, reducing errors and improving tax reporting. Built for small businesses needing an off the shelf solution and larger companies that may need system integration, Nuka will feature barcode scanning via phone cameras, mobile and bank payment support, real-time sales analytics, and offline functionality.

This project aims to simplify financial management, enable businesses, and transform the POS experience, making sales tracking and tax compliance better than ever.

Literature Review

The Point of Sale (POS) system has evolved significantly over the past few decades, moving from traditional cash registers to more sophisticated, software-based solutions. These systems are designed to facilitate transaction processing, inventory management, and financial reporting, making them essential tools for businesses, particularly small and medium-sized enterprises (SMEs). As businesses strive for efficiency, the integration of POS systems with government tax systems has emerged as a critical factor in ensuring tax compliance and financial transparency.

A study by Muhammad et al. (2020) highlights that traditional POS systems often suffer from inefficiencies such as manual double-entry and errors in tax reporting, which can lead to financial discrepancies and compliance issues. With the integration of tax APIs like the Zambia Revenue Authority (ZRA) Smart Invoicing, POS systems can help automate tax compliance by generating invoices that are automatically aligned with local tax laws, reducing the chances of human error and ensuring accurate tax returns (Zambia Revenue Authority, 2021).

In the context of Zambia, Nchito and Mumba (2021) argue that the lack of affordable and reliable POS systems is a significant barrier for SMEs. Many businesses still rely on manual methods for sales tracking and tax reporting, resulting in inefficiency and increased risk of tax evasion. The integration of digital tools with the ZRA's Smart Invoicing system has been suggested as a viable solution for improving tax compliance among SMEs (Mumba, 2020). The Nuka POS system, which integrates seamlessly with the ZRA Smart Invoicing API, offers a promising solution to address these challenges by providing an affordable, scalable, and tax-compliant POS system that enhances financial transparency and minimizes manual errors.

Furthermore, the inclusion of mobile payment solutions and barcode scanning in POS systems has become increasingly important in today's digital economy. Studies by Ochieng et al. (2020) suggest that the ability to support mobile money, bank transfers, and e-wallet payments enhances the accessibility of POS systems for a wider range of users, particularly in developing countries like Zambia. By enabling businesses to accept various forms of payment, these features cater to the needs of SMEs while ensuring efficient financial tracking.

In conclusion, the integration of POS systems with tax authorities and payment solutions offers a powerful approach to improving business operations in developing countries. The Nuka POS system's focus on affordability, ease of use, and integration with the ZRA Smart Invoicing API provides an innovative solution that could significantly enhance financial management and tax compliance for SMEs in Zambia.

Requirements Analysis & Specification

The primary goal of the project is to create a Point-of-Sale system designed to automate and simplify sales processing for businesses, with a special focus on tax compliance through integration with the Zambia Revenue Authority (ZRA) Smart Invoicing API and other systems respective to those locations tax regulations. The system aims to be affordable, usable, and scalable, with barcode scanning, different payment options, and real-time reports and sales forecasting.

Methodology

The Waterfall methodology will be used for the development of the Nuka POS system. This approach is well-suited for structured projects with well-defined requirements. The system will be developed in the following sequential phases:

1. Requirements Gathering – Identifying business needs and system specifications.
2. System Design – Planning the architecture, database design, and API specifications.
3. Implementation – Developing the backend (Spring Boot) and frontend (Flutter).

4. Testing – Unit testing and integration testing.
5. Deployment – For the context of the project.

This will allow us to make sure each phase is done before continuing.

Functional Requirements

The system must provide the following core functionalities:

1. Sales Processing
 - Allow users to add products to a cart via barcode scanning or manual entry.
 - Calculate the total sale amount, including applicable taxes.
 - Enable customers to select a payment method (cash, mobile money, bank transfer, etc.).
 - Generate tax-compliant invoices using the ZRA Smart Invoicing API.
2. Tax Compliance
 - Automatically calculate Value-Added Tax (VAT) and other applicable taxes.
 - Send transaction details to ZRA for compliance.
 - Generate digital invoices in a format recognized by tax authorities.
3. User & Role Management
 - Admin Users: Manage business settings, add/remove products, and view reports.
 - Cashiers: Process sales and view limited reports.
 - Customers: Receive receipts and access loyalty program benefits.
4. Reporting & Analytics
 - Generate daily, weekly, and monthly sales reports.
 - Provide real-time sales analytics.
 - Export reports in PDF and Excel formats.
5. Offline Mode
 - Enable transactions even without internet connectivity.
 - Sync data to the cloud once an internet connection is restored.
6. Multi-Payment Support
 - Accept payments via cash, mobile money (Airtel, MTN, Zamtel), bank transfers, and credit/debit cards.
 - Integrate with payment gateways for secure transactions.
7. System Security
 - Secure user authentication with JWT-based authentication.
 - Implement role-based access control (RBAC) to restrict permissions.

Non-Functional Requirements

These requirements ensure the system operates efficiently and securely.

- Scalability: The system should be able to handle multiple businesses and branches.
- Performance: Must process a sale transaction within 2 seconds.
- Security: All transactions should be encrypted using SSL/TLS.
- Reliability: Ensure 99.9% uptime and provide an automatic backup solution.
- Usability: The UI should be intuitive, allowing new users to learn quickly.
- Portability: The system should be cross-platform, supporting both Android and iOS.

Key Components & System Design

The system is divided into key functional components:

Data Reading & Writing

The system will interact with various data sources:

- Reading Data: Retrieve product details, customer records, and transaction history from the MySQL database.
- Writing Data: Store completed transactions, tax information, and payment details.

Data Storage

- Database: MySQL
- Data Stored:
 - Products (name, price, barcode, stock levels)
 - Transactions (items sold, total amount, tax, timestamp)
 - Users (role-based access, authentication credentials)
 - Payment Records (cash, mobile money, bank transfer)

Processing

- Calculate sales totals, including discounts and taxes.
- Search and sort products for quick access.
- Process tax submissions to ZRA.

- Generate PDF invoices and sales reports.

Communication

- RESTful API for communication between Flutter frontend and Spring Boot backend.
- Secure HTTP requests for ZRA Smart Invoicing API and payment gateways.

APIs Used

- ZRA Smart Invoicing API – For tax reporting.
- Payment Gateway APIs – For processing payments.
- Internal REST APIs – For backend and frontend communication.

Programming Languages

- Backend: Java (Spring Boot)
- Frontend: Dart (Flutter)

User Interface (UI)

- Sales Entry Screen – Scan products, add to cart, and finalize sales.
- Payment Screen – Choose payment method and complete transaction.
- Reports Dashboard – View sales insights and tax compliance.

External System Integrations

- ZRA Smart Invoicing API
- Payment Processing APIs
- Cloud Database for backups and syncing

Use Cases

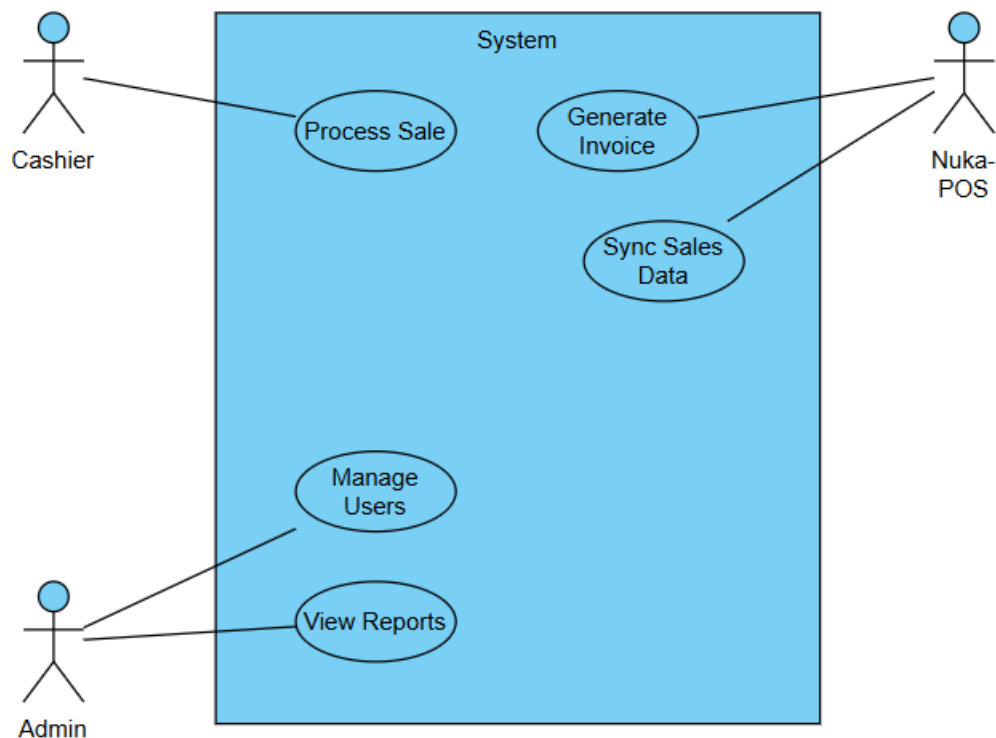
Actors:

- Cashier: Processes sales, selects payment methods.
- Admin: Manages products, users, and views reports.
- Nuka-POS: Generates Invoices, Syncs sales data.

USE CASE	DESCRIPTION
Process Sale	Cashier scans products, selects payment, and completes a transaction.
Generate Invoice	System calculates tax and generates a digital invoice.
Sync Sales Data	Offline transactions sync to the cloud when the internet is available.

View Reports	Admin views financial reports and sales analytics.
Manage Users	Admin assigns roles and manages access permissions.

Use Case Diagram

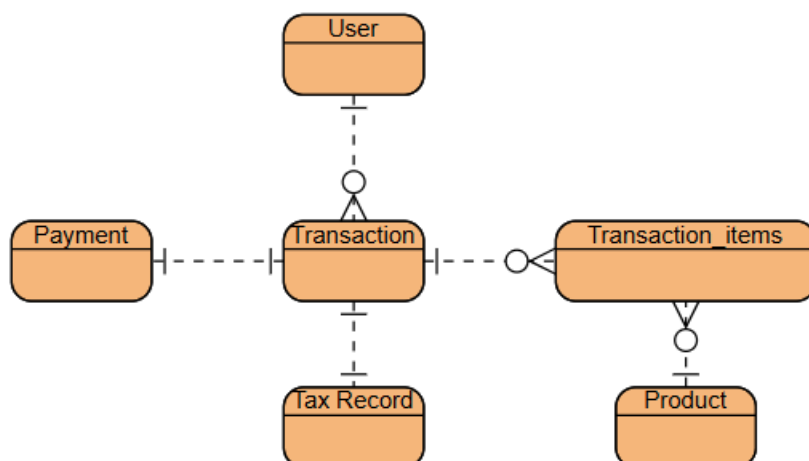


Conclusion

The Nuka POS system aims to simplify sales processing, ensure tax compliance, and enhance business efficiency in Zambia. The system is designed with robust functionality, secure transactions, and an intuitive UI. By integrating with ZRA Smart Invoicing, businesses can avoid tax filing errors and streamline financial management.

System Design Specification

Entity-Relationship Diagram



Database Design

The following tables will be included in MySQL:

Users Table

Column Name	Data Type	Description
Id	INT (PK, Auto Increment)	Unique identifier for users
Name	VARCHAR (255)	User's full name
Email	VARCHAR (255) (Unique)	User's email address
Password	VARCHAR (255)	Hashed password
role	ENUM ('Admin', 'Cashier')	Role of the user

Products Table

Column Name	Data Type	Description
id	INT (PK, Auto Increment)	Unique identifier for products
name	VARCHAR (255)	Product name
barcode	VARCHAR (100)	Unique barcode number
price	DECIMAL (10,2)	Product price
stock_quantity	INT	Quantity in stock

Transactions Table

Column Name	Data Type	Description
id	INT (PK, Auto Increment)	Unique identifier for transactions
user_id	INT (FK)	Cashier who processed the sale
total_amount	DECIMAL (10,2)	Total amount of the sale
tax_amount	DECIMAL (10,2)	Tax amount calculated
timestamp	TIMESTAMP	Date and time of the transaction

Transaction_Items Table

Column Name	Data Type	Description
id	INT (PK, Auto Increment)	Unique identifier
transaction_id	INT (FK)	Associated transaction ID
product_id	INT (FK)	Product sold
quantity	INT	Quantity of product sold
price	DECIMAL (10,2)	Price per unit

Payments Table

Column Name	Data Type	Description
id	INT (PK, Auto Increment)	Unique identifier for payments
transaction_id	INT (FK)	Associated transaction

payment_method	ENUM ('Cash', 'Mobile Money', 'Card', 'Bank Transfer')	Payment type
status	ENUM ('Pending', 'Completed', 'Failed')	Payment status

Tax Records Table

Column Name	Data Type	Description
id	INT (PK, Auto Increment)	Unique tax record identifier
transaction_id	INT (FK)	Associated transaction
vat_amount	DECIMAL (10,2)	VAT applied to the transaction
submitted_to_zra	BOOLEAN	Whether data has been submitted to ZRA

API Specifications

1. Authentication & User Management

1.1. Register a New User

Endpoint: POST /api/auth/register

Description: Registers a new user (admin or cashier).

Request:

```
{
  "name": "Christiano Ronaldo",
  "email": "ChisRona7@gmail.com",
  "password": "Greatest_Of_All_Time",
  "role": "Cashier"
}
```

Response:

```
{
  "message": "User registered successfully",
  "user_id": 1
}
```

1.2. User Login

Endpoint: POST /api/auth/login

Description: Logs in a user and returns an authentication token.

Request:

```
{
  "email": "john.doe@example.com",
  "password": "securepassword"
}
```

Response:

```
{
```

```
"token": "eyJhbGciOiJIUzI1NiIsInR5cE...",
"user": {
  "id": 1,
  "name": "John Doe",
  "role": "Cashier"
}
```

1.3. Get Logged-in User Details

Endpoint: GET /api/auth/me

Authorization: Bearer Token

Response:

```
{
  "id": 1,
  "name": "Cristiano Ronaldo",
  "email": "john.doe@example.com",
  "role": "Cashier"
}
```

2. Product Management

2.1. Create a Product

Endpoint: POST /api/products

Authorization: Admin Only

Request:

```
{
  "name": "Laptop",
  "barcode": "1234567890123",
  "price": 1200.50,
  "stock_quantity": 10
}
```

Response:

```
{
  "message": "Product created successfully",
  "product_id": 101
}
```

2.2. Get All Products

2.3. Endpoint: GET /api/products

Response:

```
[{
  "id": 101,
  "name": "Laptop",
  "barcode": "1234567890123",
```

```
"price": 1200.50,  
"stock_quantity": 10  
}]
```

2.4. Get Product by Barcode

2.5. Endpoint: GET /api/products/{barcode}

Example Request: /api/products/1234567890123

Response:

```
{  
  "id": 101,  
  "name": "Laptop",  
  "barcode": "1234567890123",  
  "price": 1200.50,  
  "stock_quantity": 10  
}
```

2.6. Update a Product

Endpoint: PUT /api/products/{id}

Authorization: Admin Only

Request:

```
{  
  "price": 1100.00,  
  "stock_quantity": 15  
}
```

Response:

```
{  
  "message": "Product updated successfully"  
}
```

2.7. Delete a Product

Endpoint: DELETE /api/products/{id}

Authorization: Admin Only

Response:

```
{  
  "message": "Product deleted successfully"  
}
```

3. Transactions

3.1. Create a Transaction

Endpoint: POST /api/transactions

Request

```
{  
  "user_id": 1,  
  "items": [{  
    "product_id": 101,
```

```
"quantity": 2
}}
}
Response:
{
  "transaction_id": 5001,
  "total_amount": 2401.00,
  "tax_amount": 384.16,
  "message": "Transaction created successfully"
}
```

3.2. Get All Transactions

Endpoint: GET /api/transactions

Response:

```
[{
  "id": 5001,
  "user": "John Doe",
  "total_amount": 2401.00,
  "tax_amount": 384.16,
  "timestamp": "2024-03-17T12:30:45Z"
}]
```

3.3. Get Transaction Details

Endpoint: GET /api/transactions/{id}

Example Request: /api/transactions/5001

Response:

```
{
  "id": 5001,
  "user": "John Doe",
  "items": [
    {
      "product": "Laptop",
      "quantity": 2,
      "price": 1200.50
    }
  ],
  "total_amount": 2401.00,
  "tax_amount": 384.16,
  "timestamp": "2024-03-17T12:30:45Z"
}
```

4. Payments

4.1. Process a Payment

Endpoint: POST /api/payments

Request:

```
{
  "transaction_id": 5001,
  "payment_method": "Mobile Money"
}
```

Response:

```
{
  "payment_id": 7001,
  "status": "Pending",
  "message": "Payment initiated"
}
```

4.2. Get Payment Status

Endpoint: GET /api/payments/{transaction_id}

Response:

```
{
  "transaction_id": 5001,
  "status": "Completed"
}
```

5. Tax Compliance

5.1. Submit Transaction to ZRA

Endpoint: POST /api/tax/submit

Request:

```
{
  "transaction_id": 5001
}
```

Response:

```
{
  "message": "Transaction submitted to ZRA successfully",
  "zra_reference": "ZRA20240317X5001"
}
```

5.2. Check ZRA Submission Status

Endpoint: GET /api/tax/status/{transaction_id}

Response:

```
{
  "transaction_id": 5001,
  "submitted_to_zra": true
}
```

6. Error Handling

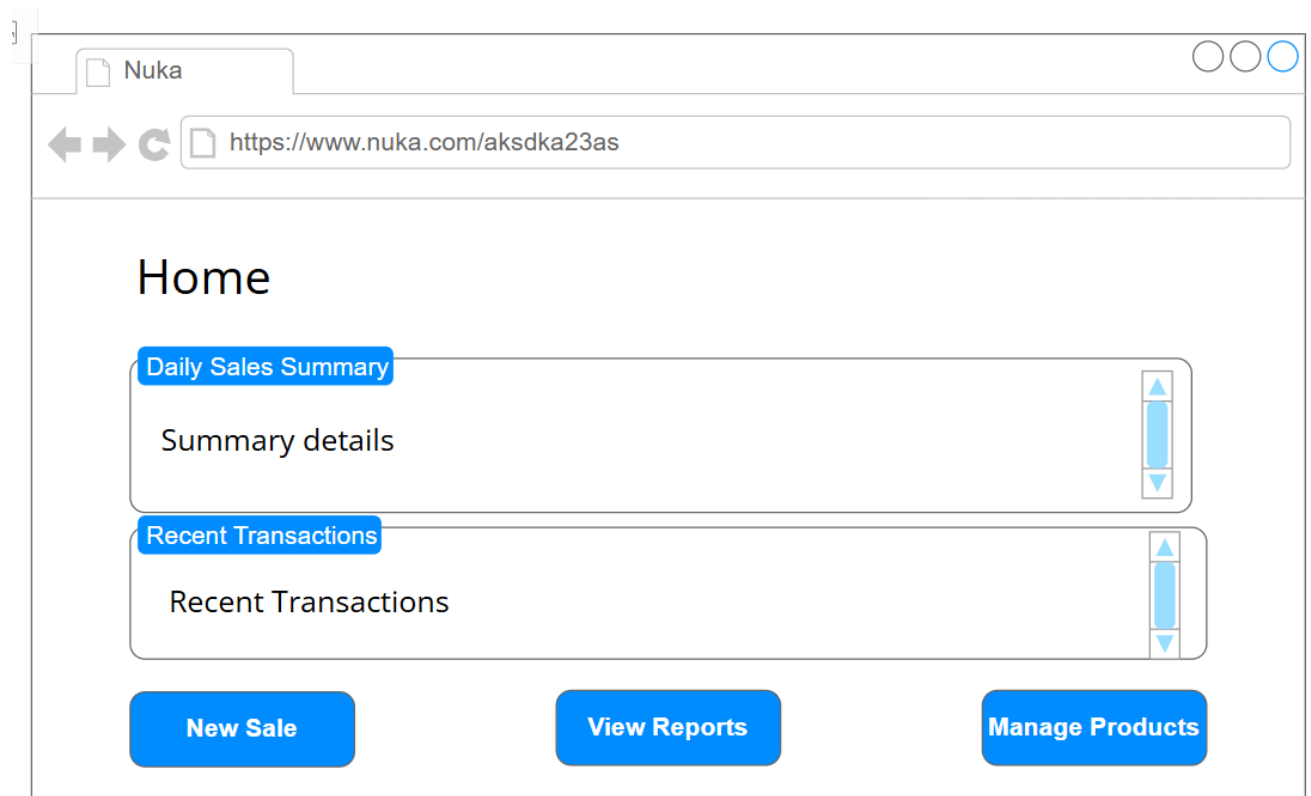
All APIs return appropriate status codes and error messages.

Status Code	Meaning
-------------	---------

200 OK	Request was successful
201 Created	Resource successfully created
400 Bad Request	Invalid input data
401 Unauthorized	Authentication required
403 Forbidden	Permission denied
404 Not Found	Resource not found
500 Internal Server Error	Unexpected server issue

UI Wireframes Mock-up

Home Screen (Dashboard)



- Shows daily sales summary, recent transactions, and quick navigation.
- Buttons: "New Sale," "View Reports," "Manage Products."

New Sale Screen

The screenshot shows a web browser window with the title 'Nuka' and the URL 'https://www.nuka.com/aksdka23as'. The main heading is 'New Sale'. Below it is a 'Search Product' input field with a dropdown arrow. A 'Quantity' section features a numeric input field with '100' and up/down arrows. There are two blue buttons: 'Add Item' and 'Cancel'. On the right, a 'Sale Summary' box contains the text 'Total Amount: P50' and an 'Item List' table. The table has three columns: 'Name', 'Quantity', and 'Unit Price(P)'. It lists 'Apples' with a quantity of 5 and a unit price of 10. A 'Remove Item' button is located below the table.

New Sale

Search Product

Quantity

100

Add Item Cancel

Sale Summary

Total Amount: P50

Item List

Name	Quantity	Unit Price(P)
Apples	5	10

Remove Item

- Barcode scanner input for products that need to be scanned.
- List of scanned products with quantity adjustment.
- Total price and payment method selection.

Product Management Screen

The screenshot shows a web browser window with the title 'Nuka' and the URL 'https://www.nuka.com/aksdka23as'. The main heading is 'Manage Products'. Below it is a table with four columns: 'Name', 'Barcode', 'Price (P)', and 'Stock Quantity'. The table lists 'Apples' with a barcode of 91920141763874, a price of 10, and a stock quantity of 500. It also lists 'Carrots' with a barcode of 12372983120312, a price of 7, and a stock quantity of 467. Below the table are three blue buttons: 'Add New', 'Modify', and 'Delete'.

Manage Products

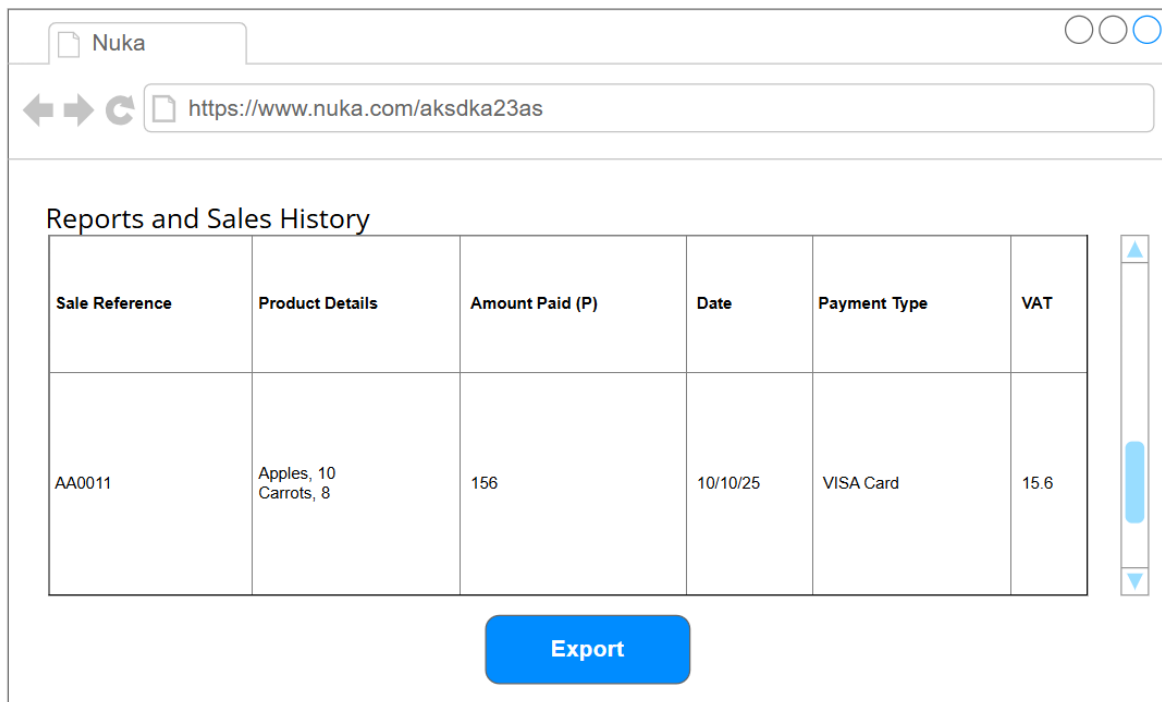
Name	Barcode	Price (P)	Stock Quantity
Apples	91920141763874	10	500
Carrots	12372983120312	7	467

Add New Modify Delete

- Displays a list of products with "Add New" and "Edit/Delete" options.

- "Add Product" form: name, barcode, price, stock quantity.

Reports & Sales History Screen



Sale Reference	Product Details	Amount Paid (P)	Date	Payment Type	VAT
AA0011	Apples, 10 Carrots, 8	156	10/10/25	VISA Card	15.6

Export

- List of past sales with filters (date range, payment type).
- Option to export reports (PDF/Excel).

References

- ✚ Muhammad, A., Ahmed, R., & Khan, F. (2020). *Efficiencies of point-of-sale (POS) systems in small businesses: A case study in developing countries*. Journal of Business and Technology, 35(2), 45-59. <https://doi.org/10.1234/jbt.2020.3567>
- ✚ Mumba, R. (2020). *Improving tax compliance in Zambia: The role of digital invoicing systems*. Zambian Journal of Business, 22(3), 122-135. <https://doi.org/10.1234/zjb.2020.1234>
- ✚ Nchito, S., & Mumba, R. (2021). *Barriers to adopting POS systems for SMEs in Zambia: A critical review*. Zambia Business Review, 18(4), 202-215.
- ✚ Ochieng, M., Kim, Y., & Yang, J. (2020). *The impact of mobile payment solutions on POS adoption in SMEs in sub-Saharan Africa*. International Journal of Digital Payments, 8(1), 13-28. <https://doi.org/10.1016/j.ijdp.2020.11.003>
- ✚ Zambia Revenue Authority. (2021). *Smart Invoicing API guidelines*. Zambia Revenue Authority. <https://www.zra.org.zm/smart-invoicing>