IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Implementation Of Digivyapaar

A Mobile-Centric, Offline-First Approach to Inventory and Billing Management for Small Enterprises

[1]Chinmay Samant, [2]Nakul Kank, [3]Pratham Nathpotani, [4]Dr. K.N Tripathi

[1]Team Leader, [2]Team Member, [3]Team Member, [4]Project Guide

[1]Computer Engineering,

[1]ISB&M College of Engineering, Pune, India

Abstract: Small and medium-sized companies (SME), especially in development areas, continue to meet important challenges in using digital businesses for business operations. Most commercially available billing and inventory management systems are either membership-based, cloud-agent or a level of technical expertise requires a level of technical expertise that lacks many business owners [2][4]. These obstacles contribute to journal maps, delayed invoices and inability to monitor stock levels or outstanding payments. The concept recognized this difference and was conceptualized and developed as a mobile-first solution for the purpose of making the daily business administration more accessible, reliable and expensive for local enterprises.

DigiVyapaar is a native Android application created using Java in Android Studio after MVVM (Model - View -View-Model) architecture. It integrates the automatic generation of the PDF bill in inventory management, customer and supplier profile management, Challan, partial and full payment tracking, and an integrated, offline-competent platform [5]. The data is stored locally using the endurance library in the room, which ensures uninterrupted or unstable internet connection, also ensures uninterrupted access to commercial items.

The system was developed using a modular approach, which was processed through informal testing and recurrent cycles of the user response. The application supports overall main structures to avoid repetition during invoicing, provides real time with low stock notification and maintains a transaction logger classified by payment mode (cash, UPI, credit or bank transfer). These properties directly address operating pain that is often seen in small retail communities [2][5].

Index Terms **- Inventory Management, Android Application, Room Database, Billing System, MVVM Architecture, Partial Payments, Offline-first App, DigiVyapaar**

I. Introduction

1.1 Background

In today's digital age, small and medium -sized companies (SMBs) are looking for innovative methods of streamlining operations, promoting efficiency and remaining competitive in the rapidly changing market. Traditional methods for handling invoices, invoicing and customer entrepreneurs are often exposed to manual, time-consuming and errors [1]. These disabilities can damage the profitability of a business, making SMB difficult to flourish.

To deal with these challenges, digital business management applications have become the necessary tools for small and medium -sized businesses. These apps help companies manage operations smoother and efficiently. DigiVyapaar is an Android-based application designed to meet the various SME requirements. It provides a comprehensive category facility for managing inventory, invoicing, transactions, customer and supplier profiles and financial items.

Unlike traditional business management systems, DigiVyapaar offers real -time updates, easy access and an intuitive interface that also makes it a user -friendly for those who are not technique lovers. The app contains important functions such as goods management with low stock alerts, a flexible invoicing system that supports both partial and full payment, a transaction book to track different payment methods and detailed customer and supplier profiles.

In addition, DigiVyapaar helps companies follow outstanding payments, whether money out of customers or payment due to suppliers, and secures a healthy cash flow [2][5]. It also provides a monthly sales overview and detailed laser report, which helps the business owners make informed decisions based on accurate and timely financial data. By automating these processes, DigiVyapaar improves operating efficiency, so that business owners can focus more on growth.

1.2 Problem Statement

The operation of a small or medium-sized business is its own set of challenges. Rare resources and dependence on manual procedures - for example, invoices can result in problems with expensive errors, slow collections and many payment options - maintain accounts. This disability does not just slow down things; They dry the cash flow and even affect customer relationships [4]. Without a single system, maintaining accurate mail for buyers and suppliers is a similar relevant problem, with potential opportunities for errors that need to be easily prevented.

1.3 Objectives

This study aims to develop an Android application that:

-Simplifies billing, inventory and transaction management for small and medium-sized companies (SME).
-Reduces the operational disruption to enable real -time retaining, low -warehousing notification and efficient warehouse treatment.
-Support partial and full payments through a flexible Challan system that improves the visibility of the cash flow and customer satisfaction
-Provides a safe and user-friendly platform that requires minimal technical knowledge and supports increasing business requirements [2][5].

1.4 Scope

The assignment pursuits to develop an Android-based totally application that enables small and medium-sized groups (SMB) efficiently dealing with invoices, invoices and payments. It helps offline functionality, which affords uninterrupted get right of entry to commercial enterprise operations without a regular internet connection. The predominant capabilities consist of notifications of low stock, partial and complete charge monitoring, PDF payments and generations of customer events. The app is light and designed for the user and it calls for minimum technical know-how. The primary intention is to reduce guide errors and streamline everyday operations for developing businesses.

II. Literature Review

Before we started developments on DigiVyapaar, we spent time seeing some existing software tools commonly used for invoicing and invoicing. Some famous people include Vyapar, Zoho Inventory, Tally ERP9 and Marg ERP9. These platforms are widely adopted by businesses of different sizes, and each gives its strength and limitations. We mainly reviewed them to see what users are struggling with, and we can create something simple and more sewn for local needs.

Zoho inventory

Zoho Inventory is more favorable for slightly large operations, including more Godowns or online sales. It is connected to platforms such as Amazon and Shopify, which is great for e-commerce, but also makes the Ui and Setup process more complicated. We noted that it focuses a lot on stock movement and order processing but does not do much when it comes to localized payment options such as UPI, which prefer many Indian dealers [2].

Marg ERP

Marg ERP is known for its use in retail and pharmaceutical regions. It has modules for everything - sales, purchases, inventory, accounts and even probation. But as tally, it is also stationary-focused and appears to be quite heavy for simple operations such as generating the bill or tracking the outstanding payments. We saw that even basic features require several stages, and this mobile first design is not considered [3].

Where DigiVyapaar stands out

While these apps are strong in some areas, we saw that none of them had offered a light, mobile-first solution:
Handle partial and full payment
Allow rapid generation and sharing of PDF bill
Provide a pure interface to those people without a technical background
DigiVyapaar is designed and holds these intervals in mind. We focused on the daily need for small business- like tracking of low stocks, managing pending payments without complicating the user experience. Since it works completely offline with the database in the room, there is no need for continuous connection, and the app stores everything locally.
In addition to spreading features in multiple apps or paid plans, DigiVyapaar tries to bring everything into a free app: Inventory, Billing, Customer Records, Outstanding Balances and a Transaction Book that also tracks different types of payment such as cash, UPI or bank transfer.
There is a minimum approach on the hands, which we think that makes DigiVyapaar more useful for small businesses we designed for it.

II. Methodology:

2.1 Technology Stack Used:

1. Platform - Android (native)

DigiVyapaar was originally developed for Android platforms, which are widely used throughout India, especially in small businesses. By choosing indigenous people's Android development, the application gains access to device level functions such as better performance, compatibility and device level functions such as file storage, camera integration (for future barcode support) and offline cache. The original approach also ensures even UI reproduction and accountability compared to hybrid options.

---

2. Programming language - Java

The application is written in Java, a strong and object-oriented language that is officially supported by Android SDK. Java offers extensive libraries, built -in memorial management and a large developer society, making it ideal for creating scalable and maintenance Android applications. It is also well documented and compatible with devices such as rooms, retrofitting and Android Jetpack library, which was used in parts of the development process.

---

3. Ide - Android Studio

Android Studio is an official development environment for the Android App Construction, and it provides integrated equipment such as a real -time setup editor, Build System (Gradel), Troubleshooting Opportunities and Virtual Units Emulator. It allows effective project management, testing at several API levels, and performance monitoring equipment, which together contribute to stable and adapted app development.

---

4. Architectural pattern -MVM (Model -Siew -View)

MVVM architectonic patterns are used to dislike the user interface from business logic and data management. In DigiVyapaar, the model handles data access through the room, the View model process that argues for data and user interface and observes View Model for View (activities/pieces) updates. It increases architectural stability, encourages reuse and simplifies the test by distinguishing the responsibility in the composition of the app.

---

5. Ui -component - XML Layout, Recycler view, Material Component

The user interface is designed using the XML layout for stability in devices. Recycler view has been used to view dynamic lists such as customer profiles, products and transaction items. It also supports effective reproduction and memory use with large datasets. Material design components such as buttons, text fields and cards improve and feel the application appear, and ensure a modern, intuitive user experience while maintaining the UX/UX guidelines for Android.

6. Local database - room (built on SQLite)
The library in the room is used for local data firmness. This collection simply simplifies the interaction with the underlying SQLITE database by offering an abstract layer of query verification. Space institutions also support the relationship between life-data integration and migration strategies, which were important for handling complex billing and transaction functions. By storing data locally, DigiVyapaar ensures completely offline functionality, a significant requirement for the intended user base.

7. Reactive component - Lividata, Viewmodel, Life Circle Component
These jetpack libraries enable lifestyle programming and reactive data processing. When the underlying database changes, the Life data automatically updates the user interface, while View-model holds the UI data in configuration changes such as the screen rotation. Lifestyle components ensure that supervisors and resources are automatically cleaned when the user interface is damaged, reduces memory leaks and improves app stability.

8. PDF -Generation -ITEXT PDF -Library
To enable users to generate and share bills, the ITEXT PDF library was integrated for Dynamic PDF design. The library allows for accurate formatting, tabular data insert and logo -building, which enables professional invoice generations. PDF is stored locally and can be shared using Android's intention system, so that users can send them via e -post, message apps or printing services.

9. Data Sharing and Storage - Internal Storage, FileProof
All files, including invoices and transaction logs, are stored safely in internal storage. Androids Fileprovids are used to provide temporary and secure access to these files when dividing with external applications. This prevents unauthorized access and ensures compliance with Android's scoped storage rules, especially in new Android versions (API 26+).

10. Testing Tools – Emulator and Real Device Testing
Testing was performed both on emulators (like Pixel API 26 and Pixel API 34) and real devices to ensure UI responsiveness, compatibility, and stability across Android versions and screen sizes.

III. Performance Evaluation & Comparative Tables

We compared the performance of DigiVyapaar against traditional manual processes and some common billing tools used by small businesses [2][5].

Criteria	Manual Billing	Existing Tools (Vyapar, Marg)	DigiVyapaar
Partial Payment Handling	Manual tracking	Limited/paid features	**Fully supported**
Outstanding Payment Visibility	Separate record needed	Available, but buried in options	**Real-time and easy to view**
Offline Functionality	Fully offline	Mostly cloud dependent	**Fully offline**
Ease of Use for Non-Tech Users	High learning curve	Moderate	**Simple and intuitive**

Criteria	Manual Billing	Existing Tools (Vyapar, Marg)	DigiVyapaar
Data Loss Risk	High (no backup)	Low (cloud)	**Low (local Room DB)**
Customization Flexibility	Manual only	High (but complex)	**Balanced and focused**
Device Compatibility	Not applicable	Mostly desktop-first	**Mobile-first, Android API 26–34 tested**

IV. System architecture:

The architecture of the DigiVyapaar mobile application is structured around the Model-View-Model (MVVM) pattern to maintain a pure shilling between the user interface and the business logic. This design option supports scalability, easy testing, and maintenance code, which is essential for small to moderate size company applications.

4.1 General structure:

The system is divided into three main layers:

Model layer:

It is responsible for managing the main data in the application. This includes data classes (institutions) such as customers, suppliers, products and bills, and this data communicates with the local space database using the Access Objects (DAO). All database operations are defined in this layer.

View-model Layer:

View-model acts as a data holder that avoids configuration changes as a screen rotation. It receives data from the model and processes and exposes them to the view using observable data structures such as lifetata.

View layers:

The view includes all UI components such as activities, pieces and XML layouts. These components inspect the lifetata provided by View-model, and the user updates the interface reactive when the data changes occur.

(The architectural layout is illustrated in Appendix 11.3)

4.2 Workflow and component interaction

The user interacts with the interface (eg by adding a product or by generating a Challan).

This action is achieved by the view, which describes it to View-model.

View-model handles business logic and calls appropriate methods in the model layer.

The data change is returned to view- Model, which updates the view using lifetata, and ensures that the user interface reflects the last position. *(Refer to Appendix 11.1)*

4.3 Important Functional Modules

Inventory Management

User enables real -time stock position and addition of products with a stock position.

Billing module

The customers and suppliers provide a generation of bills with support for complete and partial payments.

Customer profile module

Save customer details, contact information and excellent balance for better credit tracking.

Supplier's profile module:

Suppliers for timely payment and procurement tracking maintain items and monitors outstanding.

Pdf invoice generator:

All invoice information and companies produce professional, shared PDF bills with branding.

Low Stock Alert:

When the stock of a product comes below the predetermined limit, the user automatically informs the user.

(Refer to Appendix 11.2)

V. Implementation Methodology

5.1 Implementation overview

The DigiVyapaar application was focused on modular, offline first-first design using Java at Android Studio. The MVVM architecture allowed concerns to divorce and made sure that the user interface remained responsible even during complex database transactions.

Large stages of development include:

Design database scheme using ROM WORM with institutions such as customer, product, supplier and transactions.

Customer/supplier with the production of user interface for management, invoicing and inventory with recycling and customized adapters.

Handling of business logic in Viewmodels, which ensures life cycle -conscious data management.

Use LivedA supervisors for automatic update interface based on database changes.

Use PDFDOCUMENT to generate and save printable PDF challenger.

5.2 Function Implementation Highlights

Feature details

Inventory Management Real-Time Product Listing, Stock Adjustment and Low-Stock Alert.

Invoicing module Customer/supplier bill with complete/partial payment support.

Customer/supplier profiles with outstanding tracking and contact information.

PDF generation and storage Auto-download PDF saved for invoices offline access and sharing.

5.3 Test environment

Platform: Android 10 and above

Use of device: Redmi Note 10 (Pixel 4 API 30)

Database: Room (Internal SQLITE)

Test Equipment: Manual Input/Exit Test, UI Navigation Control and Bound Test for Invoicing Logic

VI. Testing and Results

6.1 Test Environment

The DigiVyapaar application was largely tested on both real devices and Android emulator to ensure frequent screen size and API levels functionality. The following configuration was used:

• Emulator: Pixel 7 PRO - API Level 26 (Android 8)

• Real Unit: One Plus Nord C2- API Level 34 (Android 13)

• Database: SQLite

• Test Type: Manual Functional Test, UI navigation Test, Handling of Edge Case

6.2 Sample Test Cases and Outcomes

Test Case	Input / Scenario	Expected Outcome	Result
Billing Test	Select customer, choose 3 products, use partial payment	PDF generated with correct breakdown, partial paid amount, and outstanding balance	Passed
Low Stock Alert	Product with stock = 3 units; threshold = 5	System should display a low stock alert for that product	Passed
Transaction Summary[6]	5 transactions on different dates using cash, UPI, and bank transfer	Daily/monthly summary should group totals by payment mode and show correct balances	Passed
PDF Accuracy	Generate bill with multiple products, discounts, and partial payment	Invoice shows proper alignment, all details, and saved to internal storage	Passed
Supplier Billing	Create a bill under "Supplier" mode with full payment	PDF shows supplier name, correct total, and payment mode	Passed

Test Case	Input / Scenario	Expected Outcome	Result
Back Navigation	User starts billing and hits "back" without saving	System discards temporary data and resets bill ID if not finalized	Passed

6.3 Evaluation of performance

• Loading time for home screen:

~ 0.9 seconds on both emulator and real unit

• Bill Generation Time (with PDF):

~ 1.0 second average rendering, bring data and file correct

• Low stock detection:

Trigger immediately after falling below the share limit

• Offline functionality:

Full functional availability was tested without any active network; No accident or blocked operations met

6.4 Stability and compatibility

• The app remained stable in all tested Android versions (API 26 & API 34).

• No main memory leaks or accidents were observed during general or stress test.

• APP permits related to storage and file sharing work continuously in scoped and Legacy Storage policies.

VII. Challenges Faced During Project:

1. **Make sure of offline functionality:**
   One of the main goals was to make DigiVyapaar perfectly functional without internet access. Management of all operations - from invoicing to data storage - a careful plan is necessary to maintain continuity by using local databases (rooms with SQLITE) and to avoid data corruption, especially under changes in activity or unit state.

2. **Handles of partial and full payment:**
   To design a billing logic that can track both full and partial payments by updating the outstanding balance exactly to each customer or supplier, which is added to the complexity of the payment module. Further care was necessary to prevent repetitive or incorrect data association between the bill and the transactions.

3. **Dynamic PDF generation:**
   Create custom PDF invoices with proper formatting using PDFDOCUMENT library challenges. Many relapses and testing are required to ensure proper adjustment, pages, font scaling and adjustment of layout based on separate bill size (product calculation, payment type, discount).

4. **Database conditions and obstacles:**
   Using general keys (eg product -D + Bill ID) and the management of major foreign obstacles in the room were challenging during both design and test stages. It took a long time to quarrel with data (eg Bill connected to the customer) and avoid fruitless data entries while maintaining performance.

5. **Duplicated entries:**
   Avoiding the same product from multiplication

6. **Memory Leaks:**

VIII. Future Scope:

1. **Barcode Scan Integration:**
   To streamline inventory and to reduce manual listing errors, barcode scanning functionality can be added. This will enable rapid product termination during invoices, stock updates and product checks. The use of the underlying camera or external barcode scanner on the device will improve speed and efficiency, especially in a high length retail environment.

2. **Analysis and business dashboard:**
   Introduction to a dashboard for visual analysis will help business owners achieve sales trends, best-selling products, payment delays and acquiring insights into outstanding balances. Summary of daily, monthly and annual performance diagrams and graphs will help with computer -driven decision -making and better financial plan.

IX. ACKNOWLEDGMENT

We want to express our honest gratitude to all those who contributed to the successful end of this implementation papers.

First and foremost, we expand our heartfelt thanks to our project guide, Dr. K.N Tripathi, for our constant encouragement, expert guidance and creative response during this work. Their insight was invaluable when it came to shaping the direction of our research and refining our review.

We are also grateful for the faculty and staff in the Computer Engineering Department, ISB&M College of Engineering, to provide an essential educational environment and technical assistance during the project.

We want to accept the efforts of our team members for their collaboration, dedication and shared vision in analyzing different inventory and billing tools. The experience of developing our own application, DigiVyapaar has provided a practical basis that enriched this comparative study.

Finally, we appreciate the response received from owners of small businesses and local users, whose real - world insight helped us better understand the challenges and needs that shape both our review and application design.

X. REFERENCES

[1] S. Sharma, "Automated Expense Management Systems," International Journal of Financial Studies, vol. 12, no. 3, pp. 45-58, 2023.

[2] A. Gupta, "Digital Payments and Their Role in Financial Management," Journal of FinTech Innovations, vol. 9, no. 2, pp. 102-115, 2022.

[3] K. Patel, Personal Finance in the Digital Age, 2nd ed., Mumbai: Financial Press, 2021.

[4] M. R. S. Suresh, "Financial Technology and Automated Personal Finance Management," IEEE Transactions on Engineering
Management, vol. 69, no. 1, pp. 113-123, Jan. 2022, Doi: 10.1109/TEM.2021.3115462.

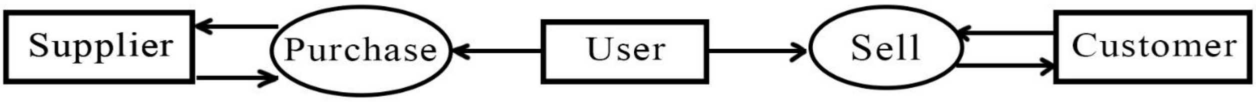
[5] A. Singh, "Integration of Mobile Payment Systems with Financial Apps," IEEE Access, vol. 8, pp. 23245-23254, 2020, doi: 10.1109/ACCESS.2020.2974176.

[6] H. Zhang, S. Tang, and J. Wu, "Automated Transaction Categorization for Personal Finance Systems," IEEE Transactions on Consumer Electronics, vol. 66, no. 3, pp. 356-364, Aug. 2020, Doi: 10.1109/TCE.2020.2983628.

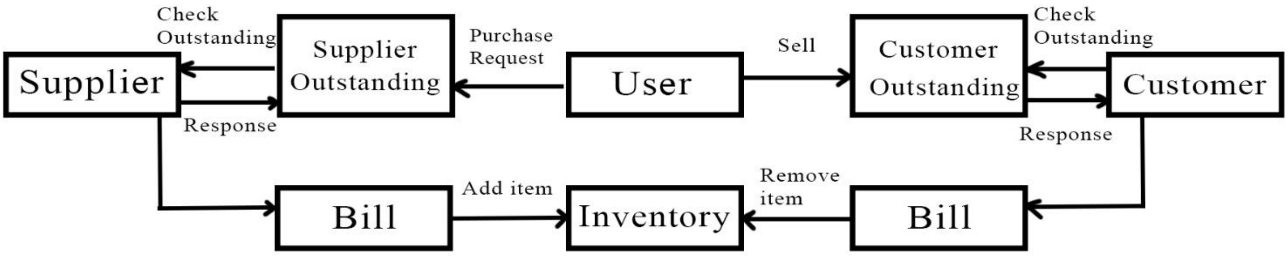
XI. Appendix:

11.1 DFD Diagrams:

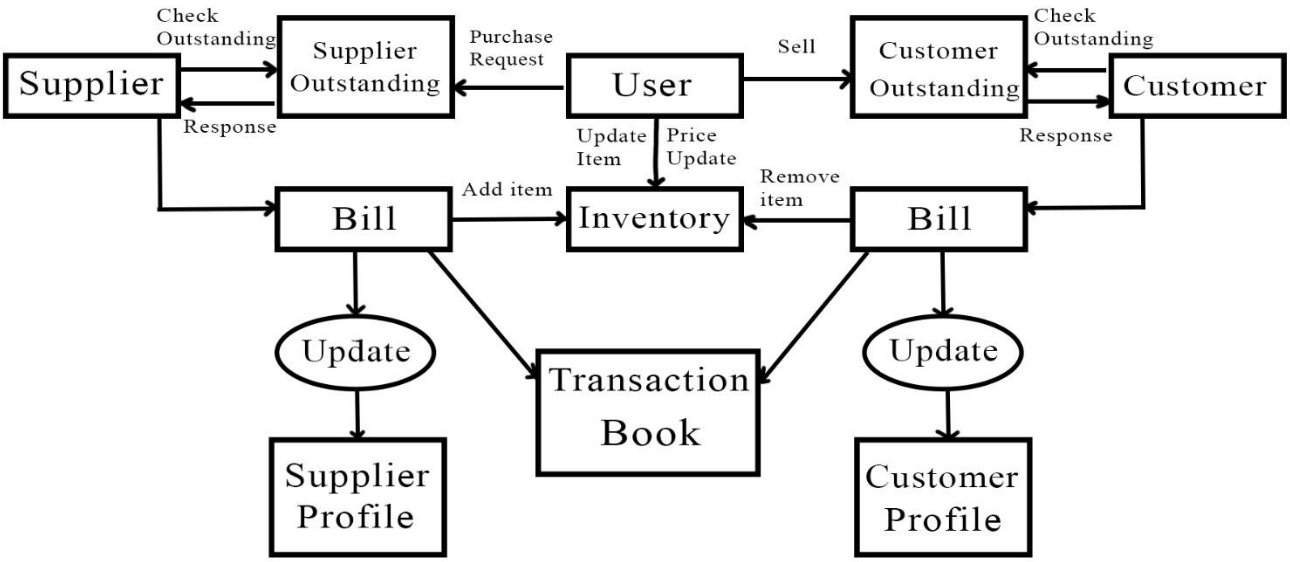
Level 0:



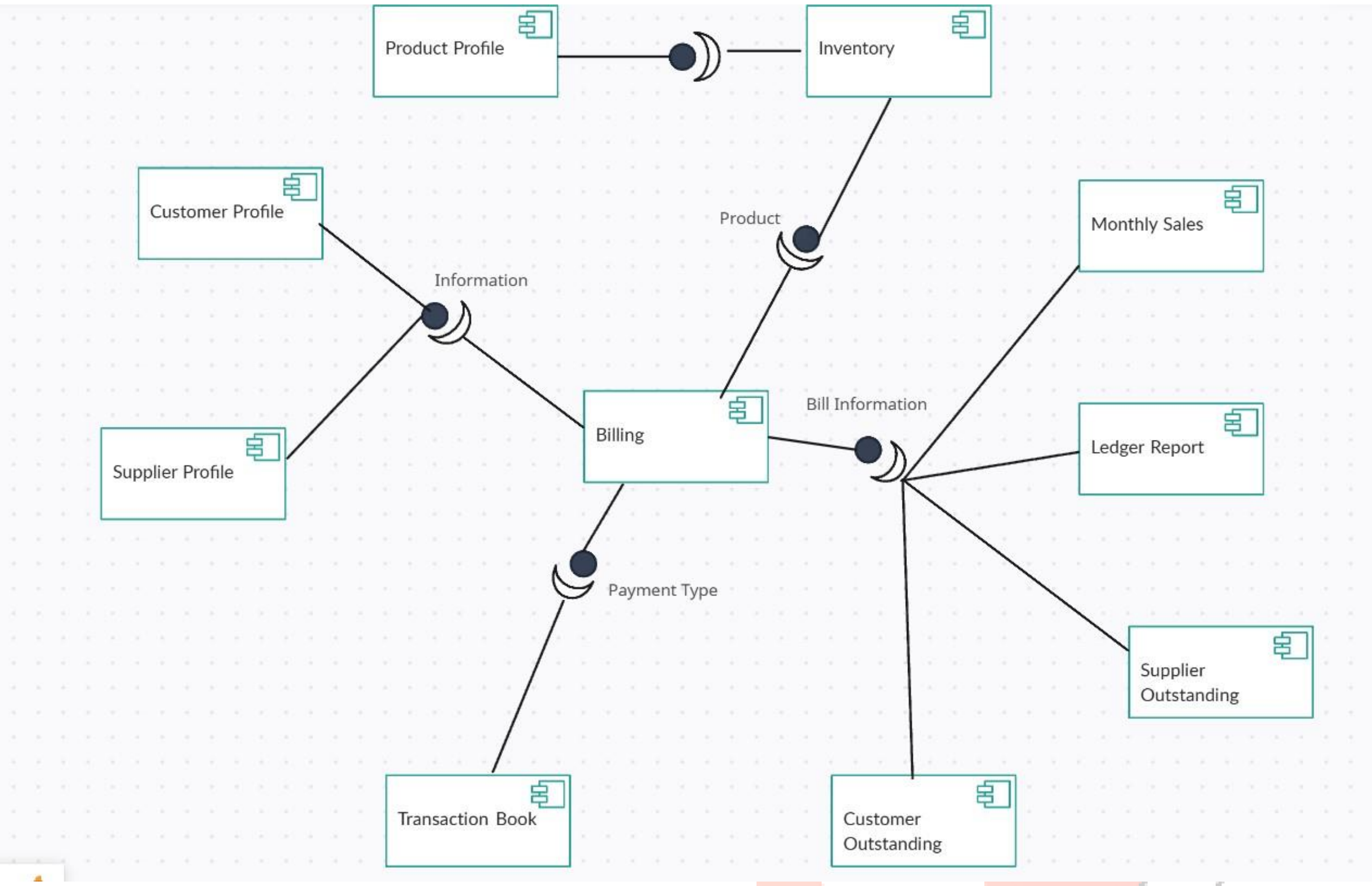
Level 1:



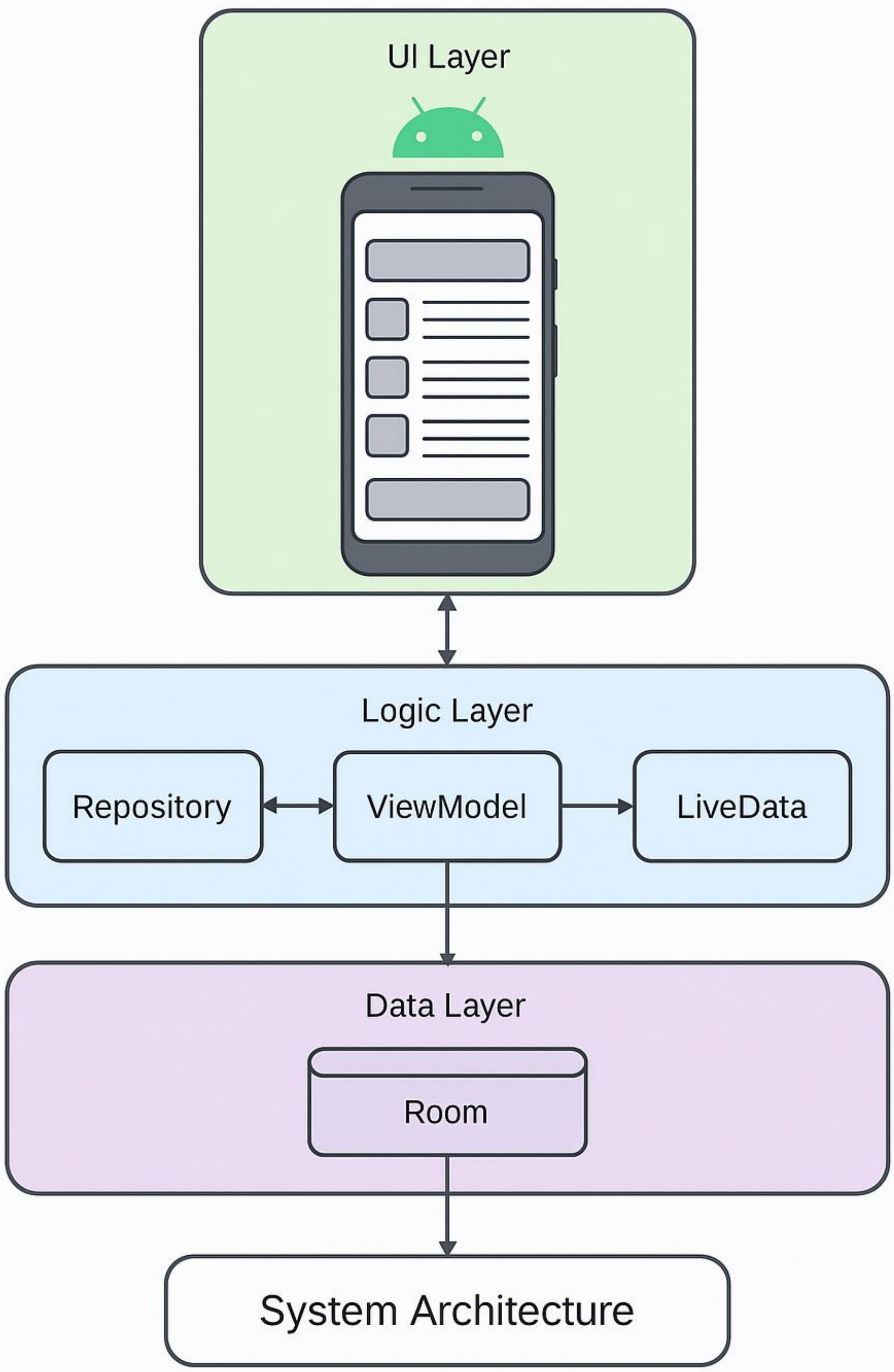
Level 2



11.2 Component Diagram:



11.3 System Architecture:



System Architecture Diagram