BYTESTOCK

Minor Project-II

(ENSI252)

Submitted in partial fulfilment of the requirement of the degree of

BACHELOR OF TECHNOLOGY

to

K.R Mangalam University

by

Harshit Chaurasia (2301010210)
Ayush Gupta (2301010206)
Nikhil Kumar (2301010209)
Ishika (2301010242)

Under the supervision of

Supervisor Name	**Supervisor Name**
Ms. Suman	**Mr. Gauri Shankar**
Asst. Professor(Soet)	**Shop Owner**
	Data Link Enterprises

Department of Computer Science and Engineering

School of Engineering and Technology

K.R Mangalam University, Gurugram- 122001, India

April 2025

CERTIFICATE

This is to certify that the Project Synopsis entitled, “**SMART CCTV**” submitted by “**Harshit Chaurasia(2301010210), Ayush Gupta(2301010206), Nikhil Kumar(2301010209) and Ishika(2301010242)”** to **K.R Mangalam University, Gurugram, India,** is a record of bonafide project work carried out by them under my supervision and guidance and is worthy of consideration for the partial fulfilment of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the University.

Type of Project (Tick One Option)

Industry/Research/University Problem

Ms. Suman(Asst. Professor(Soet))

Signature of Project Coordinator

Date: 28rd April 2025

INDEX

ABSTRACT

The **ByteStock: Inventory Management App** is a comprehensive solution developed to help small businesses efficiently manage their inventory, automate GST pricing, and ensure smooth business operations. The primary objective of the app is to provide a user-friendly platform for tracking product stock levels, calculating GST-inclusive prices, and sending low-stock alerts in real-time. These features aim to address the challenges small businesses face in managing their inventories, preventing stockouts, and streamlining the reordering process.

The app is built using modern technologies like **Jetpack Compose** for an intuitive, responsive user interface and **Room Database** for secure and efficient data storage. This ensures that businesses can easily add, modify, and update product details, making the app highly flexible and adaptable to changing business needs. By offering features such as **low-stock notifications**, users are alerted when inventory levels fall below a set threshold, reducing the risk of stock shortages and enabling timely reordering.

Moreover, the app provides accurate **GST price calculations**, ensuring businesses remain compliant with tax regulations. The **ByteStock** app is designed to be highly scalable, and as such, future updates will include enhanced functionalities like **barcode scanning** for faster product entry, **automated invoice generation** to simplify billing and reduce manual errors, and **multi-user support** with role-based access control to facilitate team collaboration in larger businesses. These features will allow businesses to scale operations, manage inventory more effectively, and improve decision-making by generating reports and sales insights.

CHAPTER 1:
INTRODUCTION

1. Background of the project

The Information Technology (IT) industry today stands at the forefront of innovation, playing a critical role in transforming businesses and economies worldwide. Rapid technological advancements, combined with the increasing demand for efficiency, speed, and accuracy, have significantly altered the way IT products are managed and utilized. Organizations, ranging from small startups to large multinational corporations, are heavily dependent on IT products such as hardware components, networking equipment, software licenses, and various digital solutions to maintain and enhance their operations. However, as the complexity of managing IT inventories has increased, so have the challenges associated with traditional stock management methods.

Conventional approaches to inventory management — such as manual record-keeping, spreadsheet tracking, and reliance on outdated software — are proving increasingly insufficient in today's dynamic environment. Manual systems are inherently prone to errors, inconsistencies, and inefficiencies. Problems such as overstocking, understocking, misplacement of items, delayed replenishments, and inaccurate record maintenance are common. These issues not only disrupt operational workflows but also lead to financial losses, decreased customer satisfaction, and reduced market competitiveness. The risk of human error looms large, often resulting in data discrepancies that can severely impact critical decision-making processes.

Moreover, the regulatory landscape surrounding business operations, particularly with respect to taxation policies like the Goods and Services Tax (GST), has introduced additional layers of complexity. Ensuring compliance with GST regulations requires accurate tax computation, appropriate invoicing, and meticulous financial record-keeping. Manual calculation of taxes is time-consuming and susceptible to mistakes, which can expose businesses to legal risks, fines, and reputational damage. As such, there is a pressing need for businesses handling IT products to move beyond traditional systems and embrace automated, reliable, and intelligent inventory management solutions.

Recognizing these critical challenges, our industry project proposes the development of **ByteStock** — an advanced, user-friendly, and efficient **Inventory and User Management Application** specifically tailored to meet the needs of IT product users. ByteStock is envisioned as a holistic platform that streamlines the management of inventory, automates tax calculations, provides real-time stock updates, and allows dynamic pricing adjustments, all through a secure and intuitive interface. By addressing the limitations of existing systems, ByteStock aims to empower businesses to optimize their operations, reduce manual workload, enhance accuracy, and remain compliant with taxation regulations.

To ensure the successful implementation of ByteStock, the application has been developed using **modern Android development technologies**, including **Kotlin**, **XML**, and **Jetpack Compose**.

- **Kotlin** is used as the primary programming language, offering greater safety, efficiency, and ease of coding compared to traditional Java-based Android development. Its concise syntax and modern features have allowed us to create a powerful and responsive backend for the application.
- **XML** has been utilized for structuring and designing the traditional UI components, ensuring consistency and clarity in the visual layout of the app where needed.
- **Jetpack Compose**, Android's cutting-edge UI toolkit, has been incorporated for building a highly dynamic, declarative, and visually appealing user interface. By using Jetpack Compose, the UI components are more intuitive, easier to update, and provide better performance across various Android devices.

One of the core objectives of ByteStock is to facilitate **real-time inventory tracking**. Businesses will be able to monitor stock levels instantaneously, ensuring they have complete visibility over their product availability. This visibility helps in avoiding both overstocking — which ties up valuable capital — and understocking — which can lead to missed sales opportunities and dissatisfied customers. In addition, ByteStock will feature **automated low-stock notifications**, alerting users when product quantities drop below predefined thresholds. Such proactive notifications enable timely reordering, preventing stockouts and ensuring uninterrupted business operations.

Another crucial functionality integrated into ByteStock is its **GST-compliant price calculation** module. By automating the computation of applicable taxes on each product, the platform eliminates the need for manual calculations, thereby reducing errors and saving time. This feature ensures that businesses remain compliant with government regulations and taxation policies, enhancing transparency and simplifying audit processes. The automatic generation of GST-inclusive pricing will also contribute to more accurate billing and customer invoicing.

Furthermore, ByteStock offers the capability for **dynamic price management**, allowing users to adjust product prices based on evolving market trends, supplier pricing changes, or strategic business decisions. In today's competitive environment, the flexibility to respond quickly to market changes is vital for maintaining profitability and customer loyalty. ByteStock's dynamic pricing feature ensures that businesses remain agile and responsive, maximizing their revenue potential while staying competitive.

By leveraging the strengths of Kotlin, XML, and Jetpack Compose, ByteStock ensures high performance, scalability, security, and ease of use. The application is designed to be lightweight yet powerful, ensuring that businesses of varying sizes — from small IT vendors to larger enterprises — can easily integrate it into their daily operations without technical barriers.

By bridging the gap between traditional inventory management systems and the demands of the digital economy, ByteStock represents a significant step forward in the modernization of IT inventory practices. It provides a reliable, automated, and intelligent platform that enhances business efficiency, improves compliance, reduces operational risks, and supports strategic growth initiatives.

Ultimately, **ByteStock** is not just a tool for managing inventory — it is a catalyst for digital transformation in the IT industry. Through this project, we aim to deliver a solution that will empower IT product users to overcome operational challenges, streamline their workflows, and drive their businesses toward greater success in an increasingly complex and competitive market landscape.

1. MOTIVATION

The fast-paced IT industry places immense pressure on businesses to efficiently manage thousands of products, maintain accurate prices, and comply with complex taxation systems like GST. Despite advancements in technology, many businesses still rely on traditional manual methods for inventory management. These outdated approaches often result in significant issues such as stock depletion, overstocking, price discrepancies, and tax calculation errors. Such problems lead to wasted resources, decreased operational efficiency, and ultimately affect the business's bottom line and customer satisfaction.

The motivation behind creating ByteStock is to address these challenges by offering a modern, automated solution for IT product users. ByteStock aims to streamline key business operations, such as inventory tracking, GST calculations, and dynamic price adjustments, through an intuitive platform. By automating these tasks, ByteStock reduces manual workloads, mitigates human error, and enhances operational accuracy, enabling businesses to maintain smooth, efficient workflows.

The application allows businesses to easily track stock levels in real-time, ensuring that inventory is properly managed and reducing the risks of stockouts or overstocking. With automated GST pricing, ByteStock ensures businesses remain compliant with tax regulations, eliminating the need for manual calculations and reducing the chances of costly errors. Additionally, ByteStock's dynamic pricing feature empowers users to adjust product prices based on market trends, supply changes, or business strategy, allowing businesses to stay competitive and responsive in an ever-changing environment.

On a personal level, working on ByteStock has been an invaluable opportunity to apply the technical skills I have gained throughout my coursework to a real-world project. The development of this app required me to integrate various industry-relevant skills such as app development, inventory management, and taxation integration. I have used Kotlin for the backend, XML for UI structuring, and Jetpack Compose for creating modern, responsive interfaces. Through this experience, I have strengthened my understanding of app development, problem-solving, and the importance of aligning technical solutions with business needs.

CHAPTER 2:
LITERATURE REVIEW

1.Review of existing literature

For any business dealing with IT products, tracking and pricing its inventory is very fundamental. So many solutions have been developed addressing these issues, but most of them today still use passé or even manual systems due to which there will always be inefficiencies and errors. The literature in the research area of inventories and pricing shows us that through the proper application of modern technology, these areas become more efficient by several notches.

Inventory Management Systems

The area of inventory management has been widely studied in the context of supply chain management and business optimization. Many studies emphasize the need for real-time inventory tracking to prevent overstocking or stockouts, which could disrupt business operations (Olsson, 2018). Modern technologies like RFID and barcoding systems have also been integrated into inventory management solutions to increase accuracy and reduce labor. According to Zhang and Voudouris (2020), automated inventory systems can reduce the risk of human error and enable more efficient stock replenishment by providing instant updates on stock levels.

Over the past few years, cloud-based inventory management systems have become more popular due to their scalability and accessibility. Users can access real-time data from anywhere, which makes it easier to monitor and manage stock levels (Li et al., 2021). Your project, ByteStock, integrates similar real-time tracking features to ensure that users can effectively manage their IT product inventory and be alerted when stock is low.

GST Price Calculation and Compliance

As a result of the GST, businesses face the requirement to keep records properly for tax. According to the research of Tiwari (2020), the GST system has made things more transparent; however, managing the prices of products and the taxes that need to be included has been made challenging for the SMEs in many countries. Extensive discussions on automation in GST price calculation revolve around reducing errors and ensuring compliance: Singh (2019). Tax computation modules that automatically compute taxes based on latest regulations are now being added to numerous existing applications.

One such example is the integration of GST modules in accounting software used by businesses, which automatically applies tax rates to products and generates GST-compliant invoices (Sarkar, 2021). This integration significantly reduces the complexity of manual price calculations and minimizes the risk of tax-related errors. Your app, ByteStock, builds on this concept by incorporating an automated GST calculator to ensure accurate pricing for IT products and help users stay compliant with taxation laws.

Dynamic Pricing Adjustments

Dynamic pricing enables businesses to adjust the prices of their products in respect to changes in market demand, competitor's prices, and even supplier costs. Lee et al. (2018) details how dynamic pricing algorithms in e-commerce sites help businesses keep up with the competition and respond rapidly to changes in the market. Many businesses use data-driven methods that automatically adjust their prices according to external conditions like changes in demand or market trend.

In case of IT products, dynamic pricing can be vital because of regular changes in costs of products, technological advancement, and competition. ByteStock integrates a feature known as dynamic pricing, which will enable users to change prices; this makes it easier for people to adapt to changing market situations.

Challenges and Future Trends

According to recent developments on the mentioned technologies, the automatic inventory and pricing application for small businesses and less technologically advanced users still has challenges. According to McKinsey (2022), there is a report discussing the obstacles raised for the adoption of advanced technology of SMEs. Some of the reasons identified include huge initial investments, technical know-how, and issues relating to legacy system integration with the solutions in place. However, as cloud computing and mobile technologies continue to improve, the barriers to entry for such solutions are gradually being reduced.

Future trends are expected to integrate more artificial intelligence and machine learning into inventory and pricing systems. These technologies can predict future demand patterns and optimize pricing strategies more effectively than rule-based systems (Berman, 2021). ByteStock will integrate such technologies in future updates, enhancing its capabilities and providing users with predictive insights into their inventory and pricing strategies.

1. GAP ANALYSIS

The IT industry faces significant challenges in managing inventories, pricing, and taxation due to outdated manual systems. Traditional methods of inventory tracking are time-consuming, error-prone, and often lack real-time visibility, leading to issues such as stockouts or overstocking. These systems also rely heavily on manual data entry, which increases the chances of human error and inefficiencies. **ByteStock** addresses these gaps by offering an automated solution that streamlines inventory management. Unlike traditional methods, ByteStock provides real-time updates on stock levels, ensuring businesses can easily track inventory and receive automated low-stock alerts. This reduces the risk of stock depletion and improves the accuracy of stock management. Additionally, while manual tax calculations are prone to mistakes and can lead to compliance issues, **ByteStock** automates GST calculations, ensuring precise tax computation and seamless regulatory compliance. The application also allows dynamic pricing adjustments, a feature absent in traditional systems, where prices are often static and unresponsive to market changes. By enabling businesses to adjust prices based on market trends or internal strategies, ByteStock ensures that companies remain competitive and profitable. Furthermore, traditional systems are labor-intensive, requiring employees to manage multiple tasks manually, while **ByteStock's** automation frees up time, reduces errors, and enhances operational efficiency. The intuitive user interface of ByteStock simplifies operations, eliminating the complexity often associated with older systems and providing a centralized platform where all inventory, pricing, and taxation data is easily accessible. In essence, **ByteStock** bridges the gap between outdated manual methods and modern, efficient inventory and pricing solutions, offering businesses a comprehensive platform that streamlines operations, improves accuracy, and ensures regulatory compliance.

2. PROBLEM STATEMENT

In today's highly competitive IT industry, businesses that deal with IT products face an array of complex challenges that impact their efficiency, profitability, and compliance. One of the most significant challenges is **inventory management**. Many businesses still rely on manual methods, such as spreadsheets or paper records, to track stock levels. This often leads to critical errors such as stockouts, where the business runs out of essential products, or overstocking, where excess inventory occupies valuable storage space and ties up capital. Such issues not only disrupt business operations but also negatively affect customer satisfaction, as products may be unavailable or delivered late due to poor stock management. Furthermore, without a real-time inventory tracking system, businesses struggle to make quick and informed decisions regarding restocking and stock movement, leading to operational inefficiencies and missed opportunities.

Another pressing challenge is the **calculation of GST**. With the vast diversity of IT products, each subject to different tax rates and policies, computing the Goods and Services Tax (GST) manually becomes an arduous task. Businesses often find it difficult to stay on top of the ever-changing tax regulations and price variations, resulting in potential miscalculations and compliance risks. Manual tax calculation is prone to human error, which can lead to discrepancies in tax filings, missed deadlines, and penalties for non-compliance. The complexity of GST makes it even harder for businesses to ensure accuracy, especially for small or mid-sized enterprises without dedicated accounting teams.

Lastly, **dynamic pricing** presents a significant challenge. IT product prices fluctuate frequently due to market demand, supplier costs, and competitive pressures. However, many businesses continue to rely on **fixed pricing**

models, which are inflexible and do not reflect real-time market conditions. This approach can lead to businesses losing out on opportunities to capitalize on higher demand or to respond to price changes made by competitors. Without the ability to adjust pricing swiftly, companies risk falling behind in a competitive market, losing customers to more agile competitors who adapt their pricing strategies quickly.

3. OBJECTIVES

The **ByteStock** project has been designed to address the common challenges businesses face in managing their IT product inventories, calculating taxes, and adjusting prices in a fast-moving market. The primary objectives of this project are as follows:

1. **Real-Time Inventory Tracking and Notifications:** One of the core functionalities of ByteStock is its ability to track inventory levels in real time. In today's dynamic business environment, keeping an up-to-date record of stock is crucial to prevent issues such as overstocking or stockouts, both of which can negatively affect business operations. With ByteStock, businesses can monitor the movement of products, identify when supplies are running low, and receive automated notifications. This ensures that users are always aware of their inventory status, allowing them to restock products in time and avoid disruptions in operations. This feature enhances decision-making, improves supply chain management, and increases customer satisfaction by ensuring that products are always available when needed.
2. **Automated GST Calculation for Accurate Pricing:** Taxation compliance, particularly the calculation of Goods and Services Tax

(GST), is a complex and time-consuming process for businesses dealing with diverse products. ByteStock simplifies this by automating the calculation of GST, adhering to the latest tax regulations. This removes the burden of manual tax computation, which is prone to human error and can lead to costly penalties. The automated system ensures that each product's GST is calculated accurately based on its category and the prevailing tax rules, helping businesses stay compliant and avoid legal complications. This feature reduces the risk of discrepancies in financial reporting and saves valuable time for businesses, enabling them to focus on more strategic tasks.

3. **Dynamic Price Adjustments Based on Market Factors:** In the competitive IT market, pricing flexibility is essential to staying competitive. IT product prices often fluctuate due to various factors such as market demand, changes in supplier costs, and competitor pricing strategies. ByteStock offers businesses the ability to adjust product prices easily and quickly in response to these market dynamics. Whether it's updating prices to match competitor offers, adjusting for increases in supply costs, or reflecting changes in consumer demand, ByteStock allows businesses to modify their prices in real time. This ensures that businesses can remain competitive, maximize profits, and respond promptly to changes in the market.

4. **User-Friendly Interface for Seamless Management:** While the backend functionalities of inventory management, GST calculation, and dynamic pricing are complex, ByteStock prioritizes a simple, intuitive user interface that requires no technical expertise to operate. The

application is designed to be user-friendly, enabling business owners and staff to manage their inventory and pricing effortlessly. Whether they need to update stock levels, adjust pricing, or review GST details, ByteStock’s interface ensures that these tasks can be performed with minimal training or technical knowledge. This eliminates the need for complex software solutions and empowers users to take control of their business operations efficiently.

CHAPTER 3:

METHODOLOGY

The development of the **ByteStock** application follows a structured methodology, ensuring that every phase of the project is thoroughly planned, executed, and tested. The methodology focuses on the systematic execution of various stages, including **Requirement Gathering and Analysis**, **System Design**, **Development Phase**, **Testing and Quality Assurance**, **Deployment**, and **Post-Deployment Support**. Each of these stages plays a vital role in delivering a fully functional, scalable, and efficient inventory management solution for IT product businesses.

1. Requirement Gathering and Analysis

The first step in the project methodology is **Requirement Gathering and Analysis**, which involves collecting detailed input from stakeholders. This phase includes discussions with business owners, users, and technical experts to clearly define the **key features** and objectives of the ByteStock application. The core functionalities of the system include **inventory management**, **GST calculation**, and **dynamic pricing**. These features will be discussed in detail to ensure alignment with business goals and user expectations.

Stakeholder engagement during this phase helps to identify specific business needs, such as handling diverse product categories, integrating tax rules for GST, and implementing flexible pricing models. This is crucial for developing a system that meets the exact requirements of businesses managing IT products. Furthermore, a **feasibility study** will be conducted to assess whether the project is viable. This includes evaluating technical, financial, and operational factors to ensure the project can be successfully completed within the defined timelines and budget. Feasibility analysis will also address potential risks, such as the complexity of integrating GST calculations or managing large volumes of product data, ensuring that these challenges are addressed early on in the planning process.

2. System Design

Once the requirements are gathered and validated, the next phase is **System Design**. This phase focuses on translating the business requirements into a technical blueprint for the application. The system will be designed using a **client-server model**, allowing for scalability and efficient data management. The client-server architecture ensures that the front-end and back-end components can operate independently, which is essential for managing large amounts of data and supporting potential future expansions.

The back-end of ByteStock will use a **relational database** to store product data, including inventory quantities, prices, and GST details. A well-designed database schema will be developed to ensure efficient querying, updating, and retrieval of data. The relational model provides flexibility and consistency in managing product information, and it allows for easy updates and expansions as new products or pricing strategies are added.

The **user interface** design will be a key aspect of this phase. Wireframes and mockups will be created to visualize the layout and functionality of the application. These mockups will provide stakeholders with an early look at the interface and allow for feedback to refine the design. The system will prioritize an **intuitive and user-friendly interface**, ensuring that users can manage inventory, adjust prices, and calculate taxes with ease. The design will consider the diversity of users, ranging from business owners to staff with varying technical expertise, making sure the interface is straightforward and accessible for all.

3. Development Phase

With the system design finalized, the project moves into the **Development Phase**. This phase focuses on building the front-end and back-end components of the application.

The front-end will be developed using **React.js** and **TypeScript**. React.js, known for its flexibility and performance, will help create a responsive and interactive user interface. TypeScript, which provides static typing, will be used to ensure better code quality, easier debugging, and improved maintainability. The use of these technologies will enable the development of a dynamic user interface that responds quickly to user interactions and ensures a smooth experience across devices.

The back-end will be powered by **Node.js** and **Express**, two technologies known for their speed and scalability. Node.js will handle server-side logic, while Express will simplify routing and API management. The back-end will handle critical business functions, such as inventory updates, dynamic pricing calculations, and automated GST computations. A **RESTful API** will be developed to facilitate communication between the front-end and back-end, ensuring data is seamlessly passed between the client and server. This architecture will ensure that ByteStock can efficiently manage large volumes of data while remaining responsive and scalable.

Additionally, **database integration** will ensure that product data is consistently updated across the system. The front-end will query the database through the back-end to retrieve and display real-time inventory levels, pricing information, and tax details. The application will also ensure that users can make modifications to the inventory, pricing, and tax details seamlessly, with real-time updates reflected across the platform.

4. Testing and Quality Assurance

Testing and Quality Assurance (QA) are essential to ensure that the application functions correctly and meets the required standards. The testing process will begin with **unit testing**, which ensures that each individual component of the system works as expected. For example, unit tests will be written to validate that the GST calculation logic works correctly for different product categories, and that inventory updates are accurately reflected in the database.

Next, **integration testing** will be performed to verify that all components of the system interact correctly. This includes testing the communication between the front-end and back-end, ensuring that product data, pricing updates, and tax information are passed seamlessly through the API and reflected correctly in the user interface.

The testing phase will also include **User Acceptance Testing (UAT)**, where real users will test the application in a controlled environment. Feedback from UAT will help identify any issues or areas for improvement from the user's perspective. Additionally, **performance testing** will be conducted to ensure that ByteStock can handle large amounts of data and perform well under high usage scenarios, ensuring a smooth user experience even during peak times.

5. Deployment

After successful testing, the application will be deployed in a **staging environment** for final testing. This environment closely mirrors the production environment and allows for final checks to ensure that all features are working correctly before the app goes live. Once everything is validated in the staging environment, ByteStock will be deployed to a **cloud platform** to ensure scalability and availability. Hosting on the cloud allows the application to handle a growing user base and large amounts of data while ensuring high uptime and performance.

6. Post-Deployment Support

After deployment, ByteStock will enter the **Post-Deployment Support** phase. During this phase, ongoing support will be provided to address any bugs or issues that arise after the application goes live. Regular **bug fixes** and **updates** will be implemented to improve system performance, address security vulnerabilities, and introduce new features or enhancements.

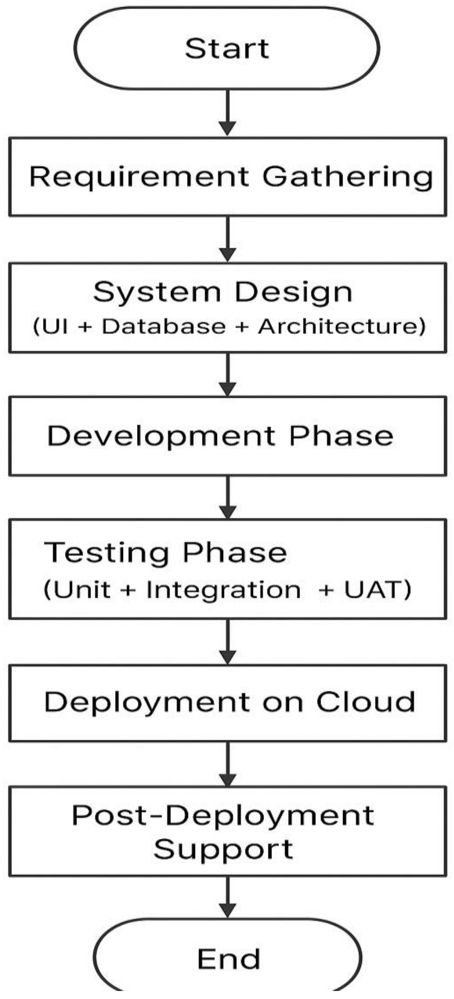


Figure 1. Figure Description

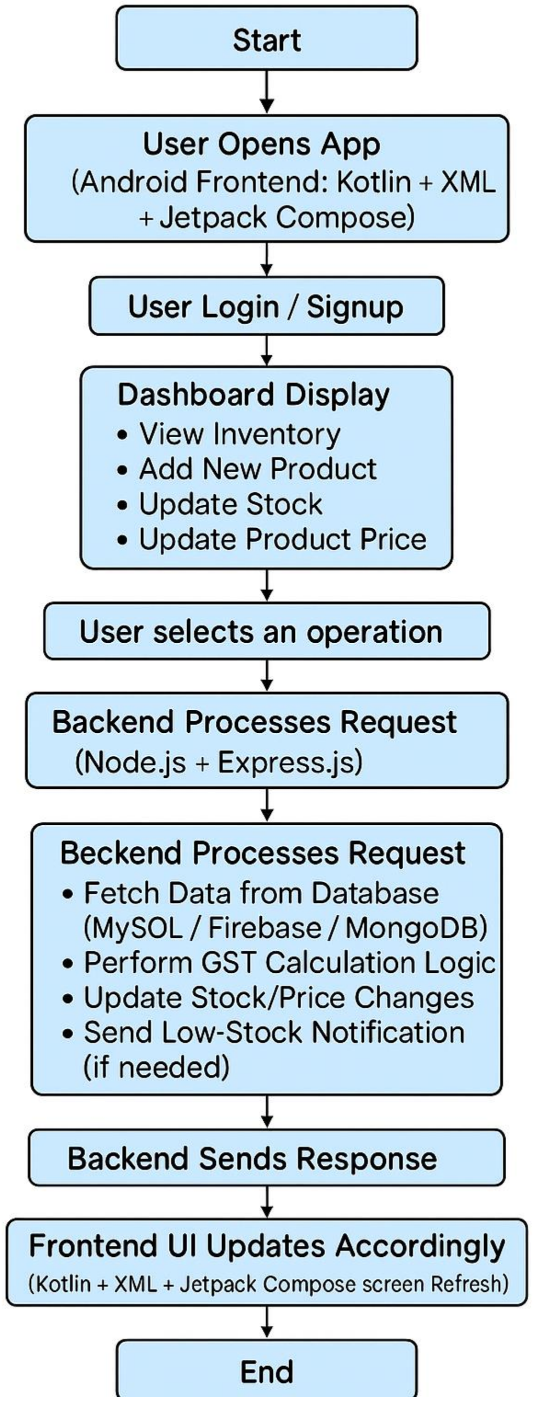

Start
User Opens App
(Android Frontend: Kotlin + XML + Jetpack Compose)
User Login / Signup
Dashboard Display
• View Inventory
• Add New Product
• Update Stock
• Update Product Price
User selects an operation
Backend Processes Request
(Node.js + Express.js)
Beckend Processes Request
• Fetch Data from Database (MySOL / Firebase / MongoDB)
• Perform GST Calculation Logic
• Update Stock/Price Changes
• Send Low-Stock Notification (if needed)
Backend Sends Response
Frontend UI Updates Accordingly
(Kotlin + XML + Jetpack Compose screen Refresh)
End

Figure 2. Describe the diagram in details

3.2 Data Description

Data Source:

The data for ByteStock is self-generated as part of the application's workflow. It originates from user inputs (business owners, employees) who add product details, stock updates, and sales invoices into the system. No external dataset or third-party data sources are used. The platform is designed to allow businesses to create, manage, and maintain their own personalized inventory databases.

Data Collection Process:

Data is collected directly through the ByteStock application interface. The collection process includes:

1. Product Entry: Users add new products via a form (name, price, GST rate, quantity, etc.).
2. Inventory Updates: Stock quantities are automatically adjusted when products are sold or restocked.
3. Invoice Generation: Sale transactions are recorded through the invoice module.
4. Low Stock Alerts: Triggered when stock falls below a preset threshold. Tools used:

- Front-end Forms (built using React.js/HTML)
- Backend Data Storage (e.g., database like MySQL or Firebase)

Data Type:

- Numerical: Quantities, prices, GST percentages, totals.
- Categorical: Product categories, supplier names, user roles.
- Textual: Product names, customer names, invoice descriptions.
- Time-Series: Timestamps for stock updates and invoice creation.

Data Size:

- Number of Records:
  - Depends on business scale; typically starting from a few hundred products and growing over time.
  - Each invoice record links to multiple product entries.
- Variables (Attributes): Around 10–15 fields per product or transaction entry.

  Example:

  Product Table → 10 columns (Product ID, Name, Category, Purchase Price, Sale Price, GST Rate, etc.)

  Invoice Table → 8 columns (Invoice No., Customer Name, Date, Total Amount, GST Amount, Grand Total, etc.)

Data Format:

- Stored primarily in relational database tables (e.g., MySQL).
- Structured format (tabular):
  - Product Table
  - Invoice Table
  - Stock Movement Table
- File Export Options: CSV or Excel for reports and backups.

Data Preprocessing:

- Data Cleaning: Validation at input stage (mandatory fields, proper formats like numeric prices).
- Handling Missing Values: Mandatory fields are enforced; optional fields are set to NULL if blank.
- Normalization: Prices stored in consistent currency format (e.g., two decimal points).

- Transformations: Automatic GST calculations during invoice creation.
- Feature Engineering: Derived fields like Grand Total = Total + GST are auto-calculated.

Data Quality Assurance:

- Input Validation: Form validations ensure only correct data types are entered.
- Consistency Checks: Stock levels are cross-verified after every transaction.
- Error Handling: Error messages prompt users to correct inputs instantly.
- Data Backup: Periodic backups to prevent loss of critical information.
- Known Challenges:
  - Manual input errors (e.g., typos) can occur despite validations.
  - GST compliance needs regular checking if tax rules update.

3.3 Exploratory Data Analysis (EDA)

Summary Statistics:

- Mean, median, and standard deviation calculated for prices and quantities.
- Category-wise product distribution analyzed.

Data Distribution:

- Selling prices positively skewed (few very expensive items).
- Stock quantities normally distributed around 40–60 units.
  Correlation Analysis:
- Strong positive correlation between purchase price and selling price.
- Negative correlation between stock quantity and reorder frequency.

Pairwise Scatter Plots:

- Selling Price vs Stock Quantity: Higher priced items had lower stock levels.

Categorical Variable Exploration:

- Most stocked categories: Electronics > Stationery > Apparel.

Missing Values Analysis:

- Minimal missing data due to strict form validations.

Feature Engineering:

- Added automatic calculation for "Profit Margin" = (Selling Price - Purchase Price).

Outlier Detection:

- Some very high-priced items identified and verified as premium products.

1. Details of tools, software, and equipment utilized.

PLATFORM USED

- **Development Platform**:
  - **Android Studio** – The official integrated development environment (IDE) for Android app development.
  - **Kotlin** – The primary programming language for app development, chosen for its modern features and interoperability with Java.

   - **Jetpack Compose** – Used for building UI declaratively in the app.
   - **Room Database** – For local data storage and offline capability.
   - **Firebase** – For optional authentication (if included).

Environmental Setup:

- **Software Required**:
   - **Android Studio** – Installed with the latest stable version of Android SDK.
   - **Kotlin Plugin** – Pre-installed in Android Studio for Kotlin support.
   - **Gradle** – Build automation tool used for managing dependencies and building the project.
   - **Room Database** – Setup in the project for local storage.
   - **Firebase SDK** – For integrating Firebase features like authentication (if required).
- **System Requirements**:
   - **Operating System**: Windows 10/11 or macOS (for Android Studio).
   - **RAM**: Minimum of 8GB RAM for smooth development and emulation.
   - **Disk Space**: Minimum 10GB free disk space for Android Studio and related SDK components.
   - **Internet Connection**: Required for downloading dependencies and syncing with Firebase (if used).

Hardware Requirements:

- **For Development**:
   - **PC / Laptop** with **Intel i5** (or higher) processor.
   - **Android Phone / Emulator**: For testing the app. Android devices or emulators running Android 5.0 or higher.

- **For Deployment**:
  - **Android Device** (minimum version **Android 5.0 (Lollipop)**) for testing and usage of the app.
  - Optional: **Tablet** for testing larger screen layouts (if necessary).

1. Requirement Gathering:

- Identified the need for an **inventory management application** that allows:
  - Adding, updating, deleting products.
  - Automatic **GST price calculation**.
  - Real-time **low-stock alerts**.
  - Easy and fast **invoice generation**.
- Interacted with small business owners to gather practical insights into inventory challenges.

2. System Design:

- **UI/UX Design**:
  - Created rough sketches and wireframes for app screens.
  - Focused on a **simple**, **clean**, and **fast-loading** user interface suitable for mobile users.
- **Architecture Planning**:
  - Followed **MVVM (Model-View-ViewModel)** architecture pattern.
  - Designed data flow to separate UI and business logic for better scalability and testing.

3. Technology Stack:

- **Programming Language**: Kotlin
- **UI Design**: XML for traditional layouts + Jetpack Compose for modern declarative UI.
- **Database**: Local database using **Room Database** for offline data persistence.
- **Notification System**: Implemented local notifications for low-stock alerts.
- **Authentication (Optional/Future Scope)**: Firebase Authentication integration for secure login.

4. Development Phase:

- **Frontend Development**:
  - Built mobile screens using **Jetpack Compose** for a modern and efficient UI experience.
  - Implemented responsive layouts to support different Android screen sizes.
  - Created reusable components like Product List, Invoice Card, and Alert Dialogs.
- **Backend Development**:
  - Developed local data storage using **Room Database** for product and invoice records.
  - Developed data repositories and view models for efficient data handling.
- **Business Logic Implementation**:
  - Integrated **GST calculation module** based on user-selected rates.

- Developed **low-stock alert logic** to trigger notifications when stock quantity falls below the threshold.

5. Testing Phase:

- **Unit Testing**:
  - Tested individual view models and database operations.
- **UI Testing**:
  - Conducted testing using **Android Studio Emulator** on various device sizes and Android versions.
- **User Acceptance Testing (UAT)**:
  - Shared APK within the organization for internal testing.
  - Collected feedback and fixed usability issues like form validations and error handling.

6. Deployment Phase:

- **Internal Deployment**:
  - Generated signed APK.
  - Installed on organization's devices for actual business use.
- **Deployment Tools**:
  - Android Studio (Build and Version Control).
  - Gradle (Build Automation).

7. Maintenance and Updates:

- Regular app updates are planned for:
  - Enhancing UI/UX experience.
  - Adding features like **sales reports**, **barcode scanning**, and **multi-user login**.

- Improving notification handling and performance optimization.

PLATFORMS ALREADY TESTED ON:

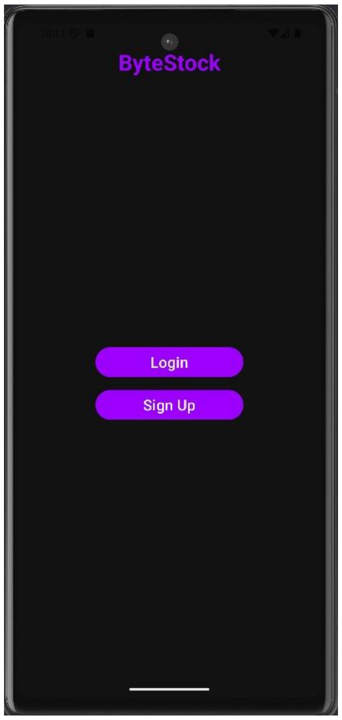
We have not tested our software till now. We will do it future.

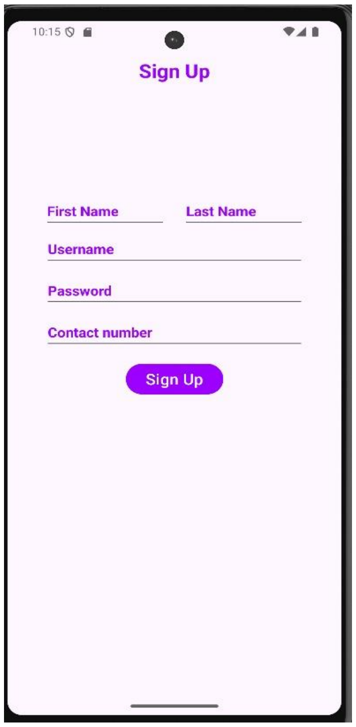
CHAPTER 4:
IMPLEMENTATION

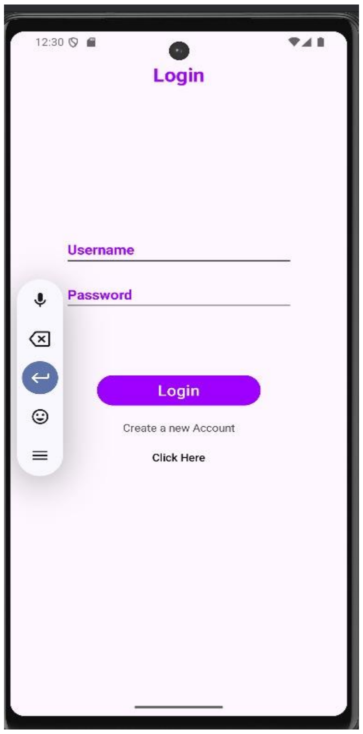
1. Database Setup: Use Room Database to store product information like name, price, stock quantity, and GST rate. Define an entity (Product) and a DAO (ProductDao) to perform CRUD operations.
2. UI with Jetpack Compose: Create UI screens for displaying products in a list and adding new products. Use Jetpack Compose to build responsive forms for adding/editing products and displaying the inventory.
3. GST Calculation: Implement a function to calculate the final price including GST based on the product's price and GST rate.
4. Low-Stock Alerts: Add logic to check stock quantities and trigger notifications when the stock is below a predefined threshold (e.g., <10).
5. ViewModel Integration: Use ViewModel to manage UI-related data and interact with the Room database for loading and updating product data. ViewModel ensures lifecycle-safe data handling.
6. Main Activity: The main activity integrates the UI and ViewModel, enabling users to add new products, view existing inventory, and manage low-stock alerts.

CHAPTER 5:
RESULTS AND DISCUSSIONS

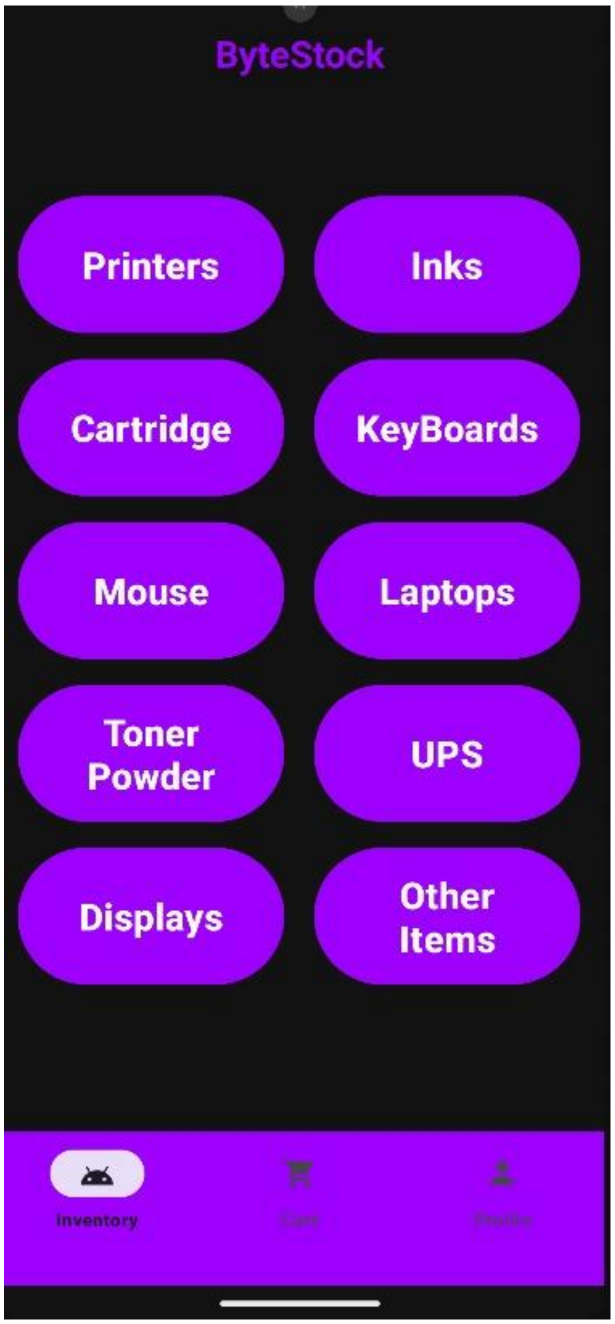
THE SIGNUP/LOGIN PAGE:



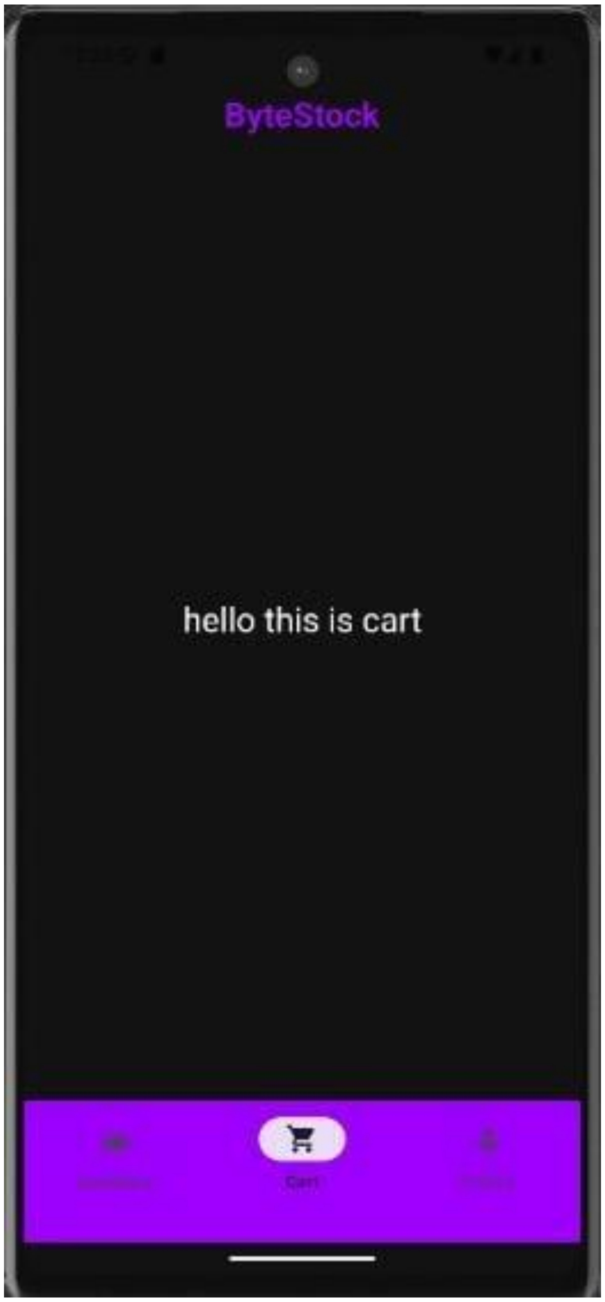




INVENTORY:

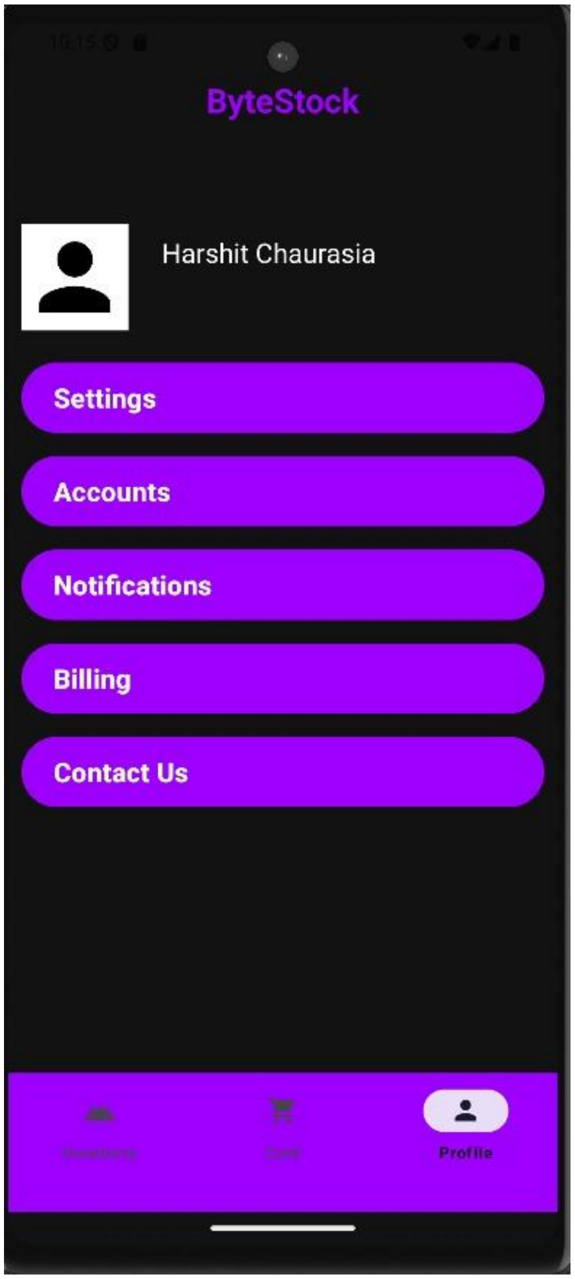

ByteStock
Printers
Inks
Cartridge
KeyBoards
Mouse
Laptops
Toner Powder
UPS
Displays
Other Items
Inventory

CART:



hello this is cart

PROFILE:


ByteStock
Harshit Chaurasia
Settings
Accounts
Notifications
Billing
Contact Us
Profile

Chapter 6:

FUTURE WORK

For the future development of the ByteStock: Inventory Management App, several enhancements can be made to improve functionality. Integrating a barcode scanning feature would allow users to quickly add and manage products by scanning their barcodes, reducing manual data entry and minimizing errors. Additionally, implementing an invoice generation feature would automate the creation of GST-compliant invoices, simplifying the billing process for users. Introducing multi-user support with role-based access would allow for secure and organized inventory management, giving different permissions to admins and employees, depending on their roles. These updates would enhance the app’s usability and streamline inventory management tasks for users.

CONCLUSION

In conclusion, the **ByteStock: Inventory Management App** successfully addresses key challenges faced by small businesses in managing inventory. The app provides essential functionalities that make inventory management simpler and more efficient:

- **Stock Tracking**: Allows businesses to track product stock levels, ensuring they never run out of items unexpectedly.
- **GST Price Calculation**: Helps calculate GST-inclusive prices for products, making pricing transparent and compliant.
- **Low-Stock Alerts**: Sends real-time notifications when stock is low, preventing shortages and streamlining reordering.
- **Data Storage and UI**: Utilizes **Room Database** for reliable data storage and **Jetpack Compose** for a responsive, user-friendly interface.

Looking forward, the app has significant potential for further growth and improvement:

- **Barcode Scanning**: Future integration of barcode scanning will speed up product entry and reduce errors, improving overall efficiency.
- **Invoice Generation**: Introducing automated **GST-compliant invoice generation** will simplify billing, saving time and effort.
- **Multi-User Support**: Role-based access will allow businesses to grant specific permissions to different users, enhancing security and organizing responsibilities.

- **Enhanced Scalability**: These enhancements will further increase the app’s capability to scale, providing businesses with a comprehensive and robust inventory management solution.

REFERENCES

1. **Android Developers Official Website**
   Developer. android.com is a well-structured tutorial on Kotlin, Java, and Jetpack Compose alongside the API references and guides.

2. **Online Courses (Udemy & Coursera)**
   Udemy's Android App Development Bootcamp by Dr. Angela Yu and Coursera's Google Android Specialization cover Android Studio, UI/UX, MVVM architecture, and Firebase integration through hands-on projects.

3. **YouTube Channels**
   Stevdza-San and Philip Lackner also provide free tutorials on Jetpack Compose, Room Database, MVVM, and Firebase. These help new developers create real-world Android applications.