



SYRACUSE UNIVERSITY

CIS 600: Android Programming

Bookstore Point Of Sale Mobile App

Kamaljit Aulakh - 758481537
Amit Prakash Vatyani - 224835474
Aditya Porwal - 249635066

Introduction

Bookstores that buy and sell inventory often face challenges in tracking their sales. Traditional point-of-sale systems typically require desktop scanners and other hardware, which can lead to increased operational overhead—particularly for small business owners operating on a thrifed model with generally low turnover per store. To eliminate the need for expensive computer systems, the ideal solution is a mobile application. Our app offers a simple yet effective platform for shop owners to streamline the inventory management and checkout processes.

This app is designed to provide technology-driven solutions for small business owners, allowing them to focus on enhancing the customer experience.

Proposed Solution

Our solution is divided into two main components:

1. Organizing and Storing Book Data in a SQL-based Database

We use a relational database to store key attributes of books, including: val title: String, val image: String, val status: String, val originalPrice: String, val discountedPrice: String, val isbn:

String, val id: String, val transactionCount: Int, val authors: String, val edition: String, and var availability: Boolean.

This data is served to the client side (the application's user interface) using Spring Boot, which provides Java-based RESTful APIs for seamless data interaction.

2. Mobile Application

An Android app built using Kotlin acts as a user-facing application. It handles the processing and display of records, while also synchronizing and updating transaction details in the SQL database through the APIs.

Key Features

1. Easy-to-navigate user interface with screens for displaying all the books, displaying details of the book, adding to the cart, and streamlined checkout.
2. Additional screen to add new books to the inventory.
3. Search across all the book data available through text-based input as well as QR code scanning.
4. App access is authorized only after a successful login.
5. Persistent storage of the logged-in user and the current items in the cart.

Competitive Analysis/Existing Solutions

While many apps in the market focus on buying and selling concepts, our app focuses on making the process faster and more efficient for bookstore owners.

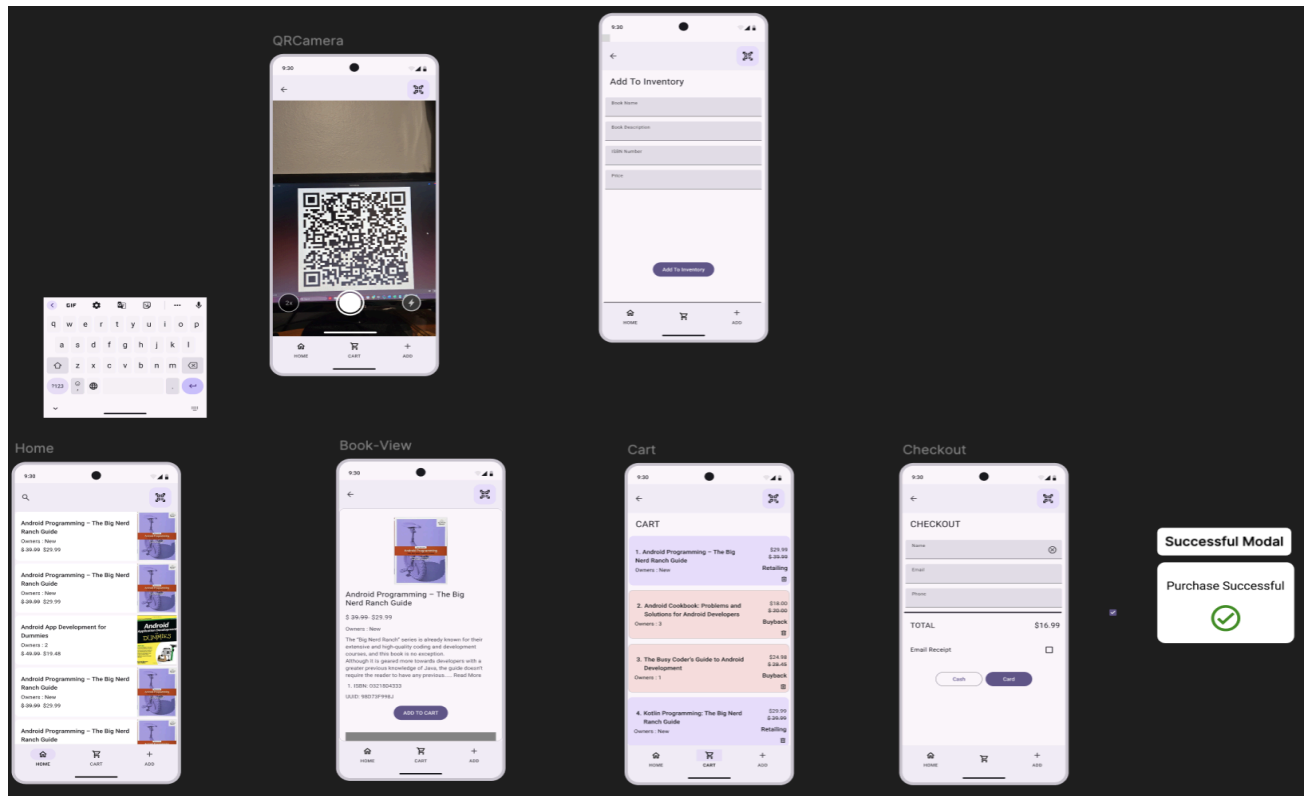
One of the leading apps in the market for books is BookScouter - Sell & Buy Books, which serves as a medium for users to buy and sell books from the comfort of their homes. However, it is exploited by users as they buy books at a cheaper price from elsewhere and sell them at a higher price.

A similar app is ThriftBooks: New & Used Books; however, this looks more like a clone of Amazon Kindle and doesn't offer an interactive user interface.

Also, many applications provide a plethora of features, including inventory management; however, this increases the overall complexity of the application, which we aim to avoid.

Initial UI Design

Below are the initial Figma wireframes which were presented as part of the design presentation. thus forming the basis of our app development for the next stages.



As per feedback from the design presentation, we implemented a login screen in our actual app discussed ahead.

Technical Stack Used

Mobile Application:

1. Android Studio
2. Kotlin
3. XML
4. Retrofit API
5. Room DB
6. Firebase Auth
7. Material UI
8. Gradle

Backend API's

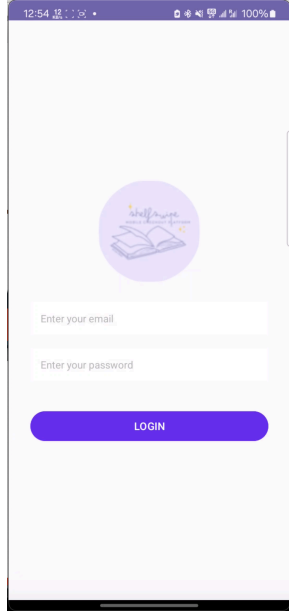
1. Java
2. Springboot
3. IntelliJ
4. Postman for testing
5. MySQL
6. AWS EC2 and AWS RDS for deployment

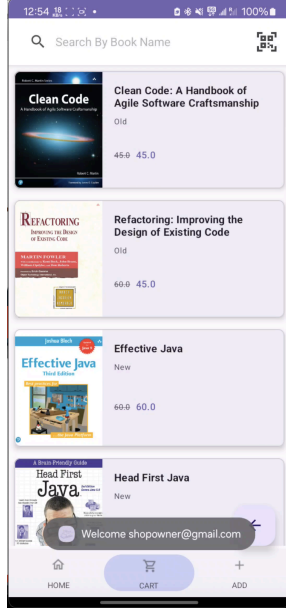
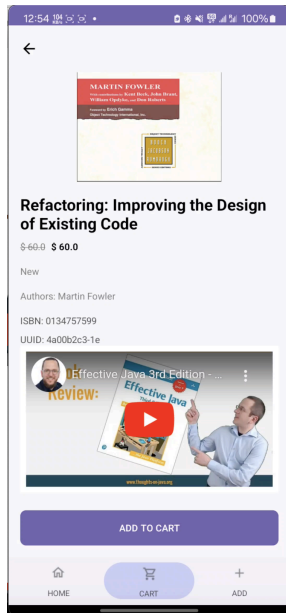
The code is available at

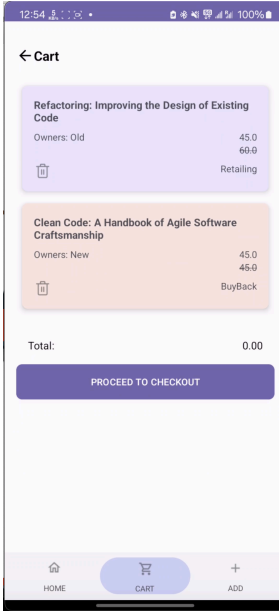
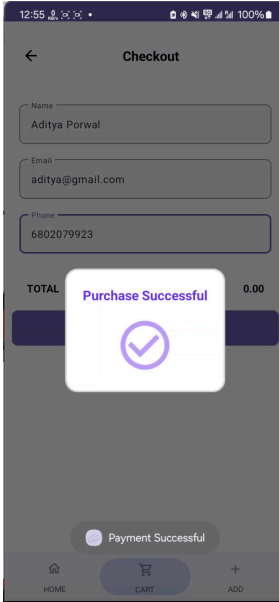
Android: <https://github.com/kamaljitkaur98/book-store-pos-android-app>

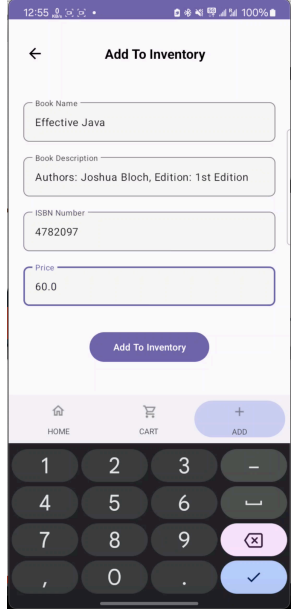
Backend API (Java): <https://github.com/adityap02/Bookstore>

Application Screens and Usage:

Screen Name	Usage	Screenshot	Feature Implemented
Login Screen	<p>Our application is specific to shop owners, not users, and thus does not have a registration page. Only users with granted credentials will be able to log in.</p> <p>Secure access is implemented to ensure that no one except the shop owner can log into the system.</p>		Firestore Auth Activity Button

<p>Home Screen</p>	<p>A list of all the books available for purchase is displayed.</p> <p>Label New/Old to help identify if it's a new copy or a preowned one.</p> <p>Each book card is clickable.</p> <p>On click, sign out the user's current session.</p> <p>Search with text and QR code.</p>		<p>Glide to display images</p> <p>Fragment for each book card</p> <p>Navigation using the bottom navbar</p> <p>Floating action button for signout</p> <p>Camera and Toolbar</p>
<p>Book Detail Screen</p>	<p>Displays all information about the book including a bonus video of a topic related to the book.</p> <p>Add to cart button</p>		<p>Bottom Navbar</p> <p>Glide</p> <p>Video Player</p>

<p>Cart Screen</p>	<p>Displays a list of all the books currently in the cart, along with a mapping to indicate whether they are being purchased or bought back by the user for buyback.</p> <p>The total amount is calculated based on the retail and buyback amounts.</p> <p>Even if the shop owner closes the app and returns, the card's content would be persistently stored in the app.</p>		<p>Bottom navigation bar.</p> <p>Fragment for each card item.</p> <p>Subtle animation when deleting each card item.</p> <p>Recycler View.</p> <p>Room DB for local persistence</p>
<p>Checkout Screen</p>	<p>Redirects the data from the card screen for the check.</p> <p>Will send the data to the API to update book details in the database.</p> <p>Confirmation pop-up showing if all API transactions are successful</p>		<p>Retrofit for API calls.</p> <p>Dialog with custom fades in animation.</p>

<p>Add to Inventory Screen</p>	<p>If a user wants to sell a book that is not already in the shopkeeper's database, it can be easily added.</p> <p>This only requires the unique ISBN of the book and other details will be pre-populated.</p> <p>The price can be adjusted according to the shopkeeper's understanding.</p>		<p>Bottom navigation bar.</p> <p>Retrofit for API calls.</p>
--------------------------------	--	--	--

Important Notes

1. Providing the app login credentials for testing purposes here

Username: shopowner@gmail.com

Password: changeme

2. IP and Port whitelisting:

Our backend data APIs are deployed on an AWS free tier account, blocked on the university network due to security issues. The data fetching should work on personal networks/carrier hotspots that don't block connection to port 5000.

For any other queries about local setup and deployment, contact

Kamaljit Aulakh: kkaulakh@syr.edu

Aditya Porwal: adporwal@syr.edu

Amit Vatyani: avatyani@syr.edu

Conclusion:

Through this project, we have been able to learn and build a strong grasp of the fundamentals of mobile application development using Kotlin. While there are many advanced features that can be implemented, such as recommending YouTube videos based on the contents of the book or a full-fledged inventory management system, our focus in this project was to build a minimal yet complete solution for a specific problem. Compared to our proposed design, we have completed nearly all the proposed features and incorporated the feedback provided by the professor.