WARRANTYWISE: AN WARRANTY AND INVENTORY TRACKING APP

A MINI PROJECT REPORT

Submitted by

SANJEEV KANTH S (2116220701250)

in partial fulfillment for the course

CS19611 – MOBILE APPLICATION DEVELOPMENT LABORATORY

of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR

THANDALAM

CHENNAI – 602 105

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE
CHENNAI - 602105

BONAFIDE CERTIFICATE

Certified that this project report **"WARRANTYWISE: AN WARRANTY AND INVENTORY TRACKING APP"** is the bonafide work of **"SANJEEV KANTH S (2116220701250)"** who carried out the project work (CS19611-Mobile Application Development Laboratory) under my supervision.

Dr. P.Kumar

HEAD OF THE DEPARTMENT

Professor and Head

Department of

Computer Science and Engineering

Rajalakshmi Engineering College

Rajalakshmi Nagar

Thandalam

Chennai - 602105

Mr.B.Bhuvaneswaran

SUPERVISOR

Assistant Professor (SG)

Department of

Computer Science and Engineering

Rajalakshmi Engineering College

Rajalakshmi Nagar

Thandalam

Chennai - 602105

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S.Meganathan, B.E, F.I.E.,** our Vice Chairman **Mr. Abhay Meganathan, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) Thangam Meganathan, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N.Murugesan, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P.Kumar, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Mr. B.Bhuvaneswaran, M.E.,** Assistant Professor (SG), Department of Computer Science and Engineering, Rajalakshmi Engineering College for their valuable guidance throughout the course of the project. We are very glad to thank our Project Coordinator, **Mr. B.Bhuvaneswaran, M.E.,** Assistant Professor (SG), Department of Computer Science and Engineering for his useful tips during our review to build our project.

SANJEEV KANTH S (2116220701250)

ABSTRACT

WarrantyWise is an Android application designed to streamline the tracking and management of product warranties and inventory records. Built using Android studio, it leverages an intuitive user interface and SQLite for persistent local data storage. The app enables users manage and monitor their purchased items along with their associated warranty periods

The app is designed to help users manage store inventory and track product warranties. It allows users to enter item details such as name, price, purchase date, warranty period, and capture a receipt image. All data is stored locally using SQLite for offline access, with basic form validation to ensure proper input.

The application also checks for items with warranties expiring within seven days and alerts the user through an AlertDialog. Additional features include receipt image capture using the device camera and a view function to display saved entries. WarrantyWise offers a simple and effective solution for organizing purchases and staying informed about upcoming warranty expirations.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In the digital age, individuals frequently purchase a wide range of consumer goods, including electronics, appliances, and personal gadgets. These items typically come with warranties that offer protection for a limited period. However, tracking these warranties manually can be inconvenient and prone to human error. Receipts may be lost, warranty periods may be forgotten, and users often miss the deadline to claim repairs or replacements. As a result, there is a growing need for a system that can help consumers manage their purchases and associated warranties in an organized and efficient manner.

This project focuses on the design and implementation of **WarrantyWise**, an Android-based mobile application developed using Android Studio and Kotlin. The app aims to provide a centralized platform where users can store essential information about their purchased items, such as the item name, price, purchase date, warranty duration, and a photo of the purchase receipt. The application uses SQLite for local data storage, ensuring offline functionality and secure data persistence. It includes form validation to ensure users provide all necessary inputs, preventing incomplete records.

Additionally, the app includes a built-in alert system that notifies users of items with warranties nearing expiration. This is done by comparing the current date with the stored purchase date and warranty duration and alerting users through a clean, user-friendly AlertDialog interface. By integrating features like camera access for receipt image capture and real-time validation, the application enhances usability and functionality. Ultimately, **WarrantyWise** simplifies the

process of inventory and warranty management, making it a practical tool for everyday consumers.

1.2 **OBJECTIVE**

The primary objective of this project is to develop an Android application, WarrantyWise, that enables users to manage their store inventory and track the warranty status of purchased items in an efficient and user-friendly manner.

The specific objectives of the project are as follows:

- To provide a simple form-based interface for entering item details such as name, price, purchase date, and warranty period.
- To enable users to capture and store receipt images using the device's camera.
- To store all item data securely and persistently using SQLite as the local database solution.
- To implement input validation to ensure all required fields are completed before an item is saved.
- To design a system that automatically checks for items nearing warranty expiration and alerts the user accordingly.
- To allow users to view all saved items in an organized format using a popup dialog.

Through these objectives, the project aims to deliver a practical solution that helps users avoid missed warranty claims and maintain a digital record of their purchases.

1.3 EXISTING SYSTEM

In the current scenario, most users rely on manual methods to keep track of their purchased items and their respective warranty periods. These methods include storing physical bills or receipts, maintaining handwritten records, or saving basic details in general-purpose applications such as Notes or Excel spreadsheets. While these approaches may offer temporary solutions, they are often unreliable and inefficient for long-term use.

One of the major drawbacks of the existing system is the lack of timely reminders or alerts regarding warranty expirations. Users may forget about the warranty deadlines or misplace receipts, leading to missed opportunities for claiming free repairs or replacements. Additionally, conventional systems do not offer features such as automatic date tracking, structured form input, or integration with device features like the camera. They lack automation, ease of access, and the ability to manage large numbers of items efficiently on a mobile device.

These limitations highlight the need for a dedicated, mobile-based solution that not only stores warranty-related data but also offers intelligent tracking and notification features.

1.4 PROPOSED SYSTEM

The proposed system, WarrantyWise, is an Android-based mobile application designed to overcome the limitations of manual and unstructured methods of warranty tracking. It offers a centralized and digital solution for managing store inventory and associated warranties with ease, accuracy, and efficiency. The application allows users to input and save essential item details such as the name, price, purchase date, warranty duration, and a photo of the purchase receipt using the device’s camera.

The system incorporates form validation to ensure all necessary information is provided before saving, thereby reducing data entry errors. It uses SQLite as the local database to securely store item data on the user's device, ensuring offline accessibility without reliance on internet connectivity. One of the key features of the proposed system is its ability to automatically check for items with warranties nearing expiration (within seven days) and notify the user using an intuitive AlertDialog, helping them take timely action.

In addition, the app provides a simple interface for users to view saved items, making it easy to refer back to previous purchases. The use of Android’s native features, such as camera access and date pickers, enhances user experience and interaction. Overall, the proposed system delivers a reliable, portable, and user-friendly platform that simplifies warranty and inventory management while minimizing the risk of missed warranty claims.

CHAPTER 2

LITERATURE SURVEY

2.1 GENERAL

In recent years, the need for mobile-based solutions for personal inventory and warranty tracking has grown with the increase in consumer purchases, especially electronics and appliances that come with time-bound warranties. While there is limited literature specifically on warranty tracking apps, related work in mobile inventory systems, reminder applications, and personal data management offers significant insights into the development and design of applications like **WarrantyWise**.

Several existing mobile applications, such as *Evernote*, *Google Keep*, and *Microsoft OneNote*, allow users to manually store information like receipts or warranty details. However, these apps are general-purpose and not optimized for structured inventory or warranty tracking. They lack specialized features such as warranty expiration calculation, automatic alerts, or camera-integrated receipt capture. Moreover, the burden of organization and tracking remains on the user, which makes these tools ineffective for systematic warranty management.

Some commercial solutions like *Breezeway* or *Sortly* offer inventory management capabilities for business users, but they are either paid or complex for personal use. Academic studies on personal information management systems (PIMs) have emphasized the importance of user-centered design, data persistence, and proactive notification systems in enhancing user reliability and adoption of such tools. Research in mobile application usability also suggests that integrating native features (like camera access and local databases) greatly improves accessibility and performance in offline scenarios.

From a technical perspective, the use of **SQLite** in mobile applications has been widely acknowledged as a reliable method for lightweight, embedded storage. In Android development, SQLite allows efficient handling of structured data without the need for a remote server. Similarly, studies on **form validation** stress its role in preventing incomplete or inconsistent user input, which directly enhances data integrity and app reliability.

Additionally, prior work in **reminder systems and alert mechanisms**—such as medication alerts or calendar-based event reminders—demonstrates the effectiveness of timed notifications through dialog boxes or push notifications. These concepts are adapted in **WarrantyWise** through the use of AlertDialogs for real-time user feedback on warranty expiry.

In conclusion, the literature and existing tools highlight a gap in dedicated personal warranty tracking applications. The proposed solution fills this gap by combining key aspects of inventory tracking, automated alerting, and mobile usability into a unified platform tailored for consumers. By integrating lessons from existing tools, research studies, and user interface design, **WarrantyWise** aims to provide a comprehensive and practical solution.

CHAPTER 3

SYSTEM DESIGN

3.1 GENERAL

System design is a crucial phase in the mobile application development lifecycle. It defines the overall architecture, component interaction, data flow, and user interactions. The system design for **WarrantyWise** revolves around creating a user-friendly, efficient, and reliable Android application to track product warranties and inventory

This chapter outlines the design through three perspectives: the System Flow Diagram, the Architecture Diagram, and the Use Case Diagram. These visuals represent the app's behaviour from both the technical and user-centric points of view

3.1.1 SYSTEM FLOW DIAGRAM

The system flow diagram outlines the sequence of actions triggered

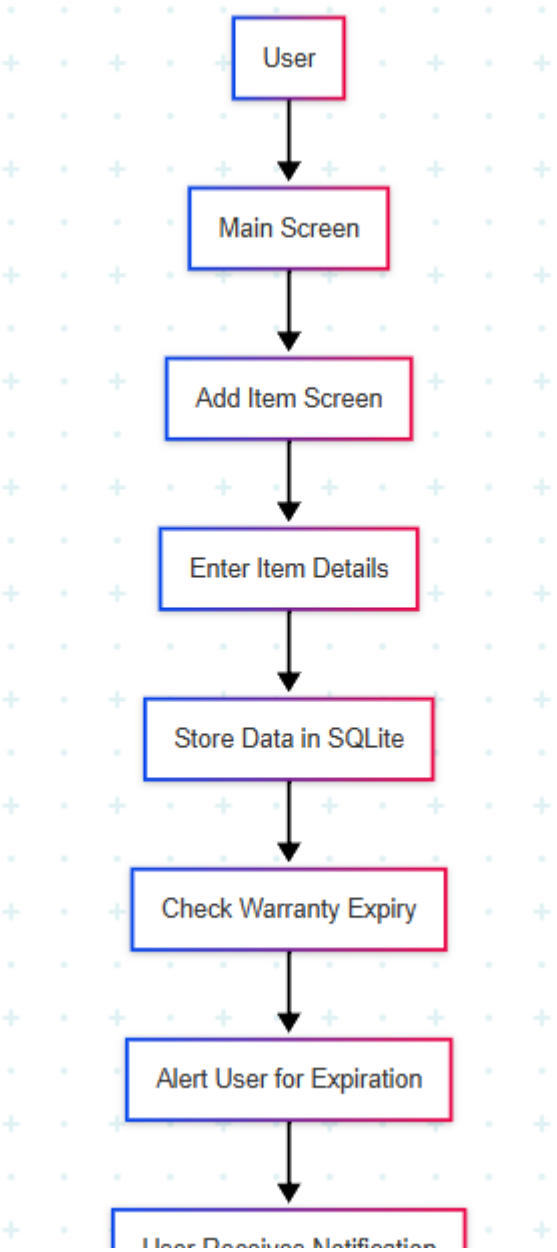


Fig 3.1 System Flow Diagram

3.1.2 ARCHITECTURE DIAGRAM

This diagram describes the core components of the application and how they interact with the device hardware and Android OS services.

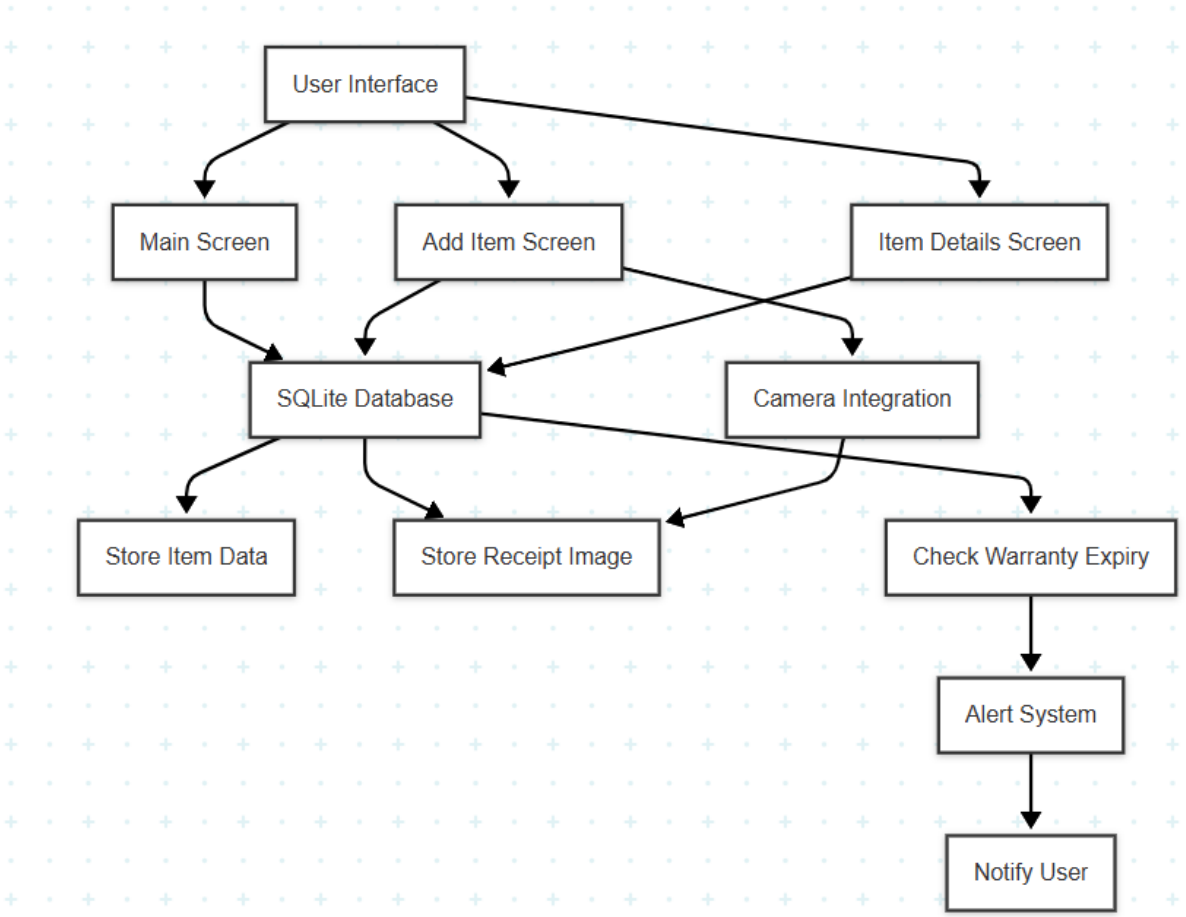
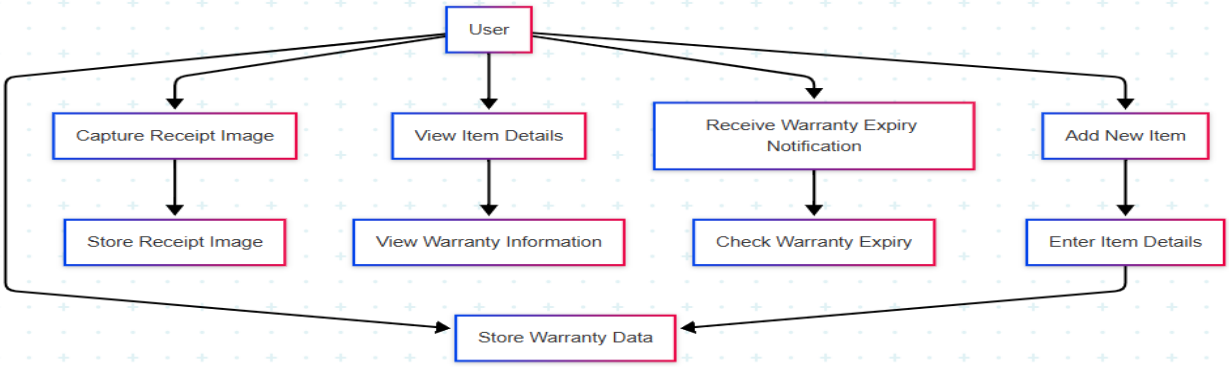


Fig 3.2 Architecture Diagram

3.1.3 USECASE DIAGRAM

The Use Case Diagram describes how the user interacts with the application and highlights the various actions that can be performed



CHAPTER 4

PROJECT DESCRIPTION

4.1 METHODLOGIES

The design and development of **WarrantyWise** followed these steps:

1. **Requirement Analysis**: Understanding user needs for tracking warranties and product information, including the need for offline access, security, and reminders for warranty expirations.
2. **UI/UX Design**: The user interface was designed to be intuitive and simple, ensuring ease of navigation and interaction.
3. **Data Storage**: **SQLite** was chosen for local data storage to ensure offline access and secure persistence. The app stores item details such as product name, price, purchase date, warranty period, and receipt images.
4. **Alert System**: A built-in alert system was implemented to notify users when a warranty is nearing expiration, allowing them to take appropriate actions.

4.1.1 MODULES

The project comprises of the following modules

- Form Validation
- Image Integration
- Alert Module
- Database

FORM VALIDATION

The **Data Entry and Form Validation Module** ensures that users provide valid and complete input when adding new items. This module checks that required fields such as item name, price, and purchase date are filled out correctly, and it prevents users from submitting forms with missing or improperly formatted data. This reduces errors and maintains data integrity.

```
btnSave.setOnClickListener {
    val name = edtName.text.toString()
    val price = edtPrice.text.toString()
    val purchaseDate = edtPurchaseDate.text.toString()
    val warrantyMonths = edtWarrantyMonths.text.toString()

    if (name.isEmpty() || price.isEmpty() || purchaseDate.isEmpty() || warrantyMonths.isEmpty() || receiptImagePath.isEmpty()) {
        Toast.makeText( context: this,  text: "Please fill all fields and capture receipt", Toast.LENGTH_SHORT).show()
    } else {
        val success = dbHelper.insertItem(
            name,
            price.toDouble(),
            purchaseDate,
            warrantyMonths.toInt(),
            receiptImagePath
        )
        if (success) {
            Toast.makeText( context: this,  text: "Item saved!", Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText( context: this,  text: "Error saving item", Toast.LENGTH_SHORT).show()
        }
    }
```

IMAGE INTEGRATION

The Camera Integration Module enables users to capture images of their purchase receipts directly within the app. This feature utilizes the device's native camera and stores the captured image URI, which is then associated with the respective item in the database. This makes it easier for users to retrieve proof of purchase when needed.

```
private fun saveImageToInternalStorage(bitmap: Bitmap) {
    val filename = "receipt_${System.currentTimeMillis()}.jpg"
    val file = File(filesDir, filename)
    val outputStream = FileOutputStream(file)
    bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 100, outputStream)
    outputStream.flush()
    outputStream.close()
    receiptImagePath = file.absolutePath
}
```

ALERT MODULE

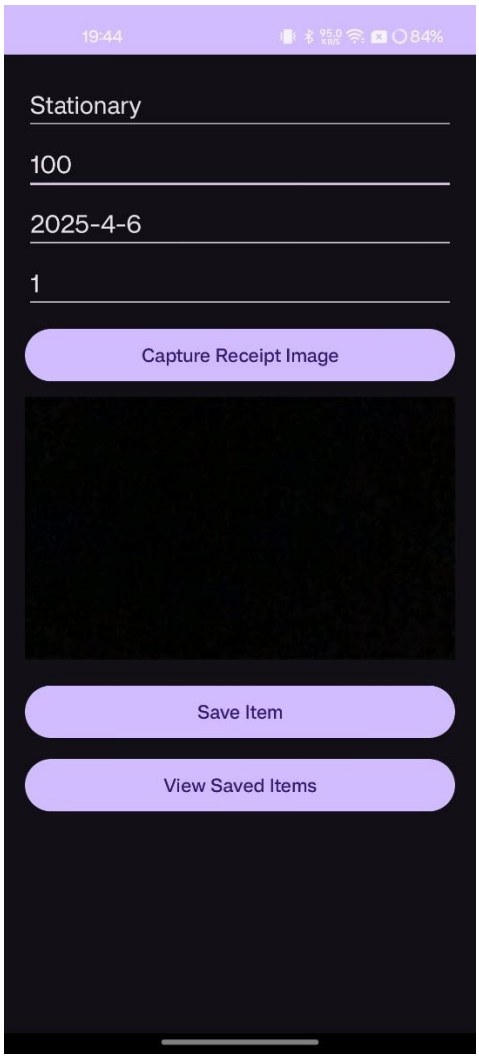
A crucial feature of WarrantyWise is the Warranty Expiry Check and Alert Module. This module regularly compares the current date with each item's warranty expiration date. If any warranties are found to be expiring within the next seven days, the app triggers an alert using an AlertDialog, ensuring that users are reminded to take timely action on their products.

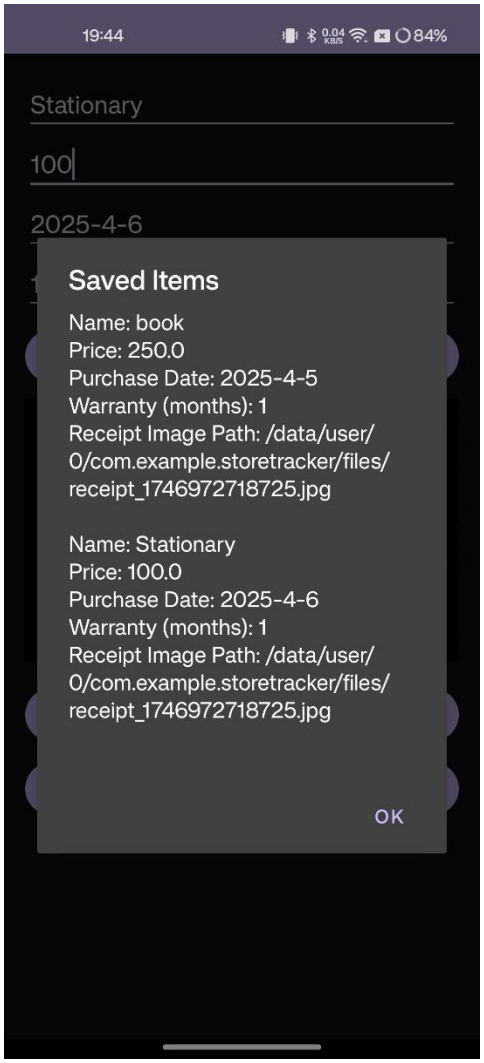
DATABASE

The SQLite Database Module handles all data storage operations locally on the device. It stores item details including the name, price, purchase date, warranty period, and receipt image URI. Because SQLite operates offline, the app maintains full functionality even without an internet connection. This module supports inserting, updating, querying, and deleting data as needed

```
class DBHelper(context: Context) : SQLiteOpenHelper(context, "store_inventory.db", null, 1) {

    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL(
            "CREATE TABLE items (" +
                    "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "name TEXT, " +
                    "price REAL, " +
                    "purchase_date TEXT, " +
                    "warranty_months INTEGER, " +
                    "receipt_image_path TEXT" +
                    ")"
        )
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS items")
        onCreate(db)
```

4.2 PROJECT OUTPUT SCREENS





CHAPTER 5

CONCLUSION

5.1 GENERAL

The WarrantyWise application provides a practical and efficient solution for managing store inventory and tracking product warranties. By integrating features such as local data storage with SQLite, camera support for receipt image capture, form validation, and warranty expiry alerts, the app simplifies the otherwise tedious process of organizing purchase records. Its user-friendly interface and offline capability ensure accessibility and ease of use for everyday consumers. Overall, WarrantyWise is a reliable tool that empowers users to stay informed about their purchases and warranty periods, reducing the risk of missed service opportunities and enhancing post-purchase convenience.

We have accomplished:

- **Offline Data Storage with SQLite**
  Ensured reliable and persistent local storage of item data and receipts without internet dependency.
- **User-Friendly Interface**
  Developed a clean, intuitive UI using Android Studio and Kotlin for smooth navigation and usability.
- **Camera Integration for Receipt Capture**
  Enabled in-app receipt image capture using the device camera and stored images efficiently.
- **Warranty Expiry Alert System**
  Implemented automated expiry checks and timely user alerts using AlertDialogs for upcoming warranties.
- **Form Validation for Data Integrity**
  Added input validation to prevent incomplete or incorrect entries during item registration.

- **Efficient Data Retrieval and Display**
  Built seamless mechanisms to fetch and display item details and images clearly on the UI.
- **Modular and Scalable Architecture**
  Structured the app into well-defined modules, making it easy to maintain and extend in the future.

APPENDIX

SOURCE CODE

Androidmanifest.xml

```
<uses-feature
        android:name="android.hardware.camera"
        android:required="false" />

    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        android:maxSdkVersion="32"
        tools:ignore="ScopedStorage" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
        android:maxSdkVersion="32" />
```

Mainactivity.kt

```
private lateinit var dbHelper: DBHelper
   private lateinit var imageView: ImageView
   private var receiptImagePath: String = ""

   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)
       setContentView(R.layout.activity_main)

       dbHelper = DBHelper(this)

       val edtName = findViewById<EditText>(R.id.edtName)
       val edtPrice = findViewById<EditText>(R.id.edtPrice)
       val edtPurchaseDate = findViewById<EditText>(R.id.edtPurchaseDate)
       val edtWarrantyMonths = findViewById<EditText>(R.id.edtWarrantyMonths)
       val btnCaptureImage = findViewById<Button>(R.id.btnCaptureImage)
       val btnSave = findViewById<Button>(R.id.btnSave)
       imageView = findViewById(R.id.imageView)
       val btnViewItems = findViewById<Button>(R.id.btnViewItems)

       btnViewItems.setOnClickListener {
           val items = dbHelper.getAllItems()
           if (items.isEmpty()) {
               Toast.makeText(this, "No items found in the database.", Toast.LENGTH_SHORT).show()
           } else {
               val builder = StringBuilder()
               for (item in items) {
                   builder.append("Name: ${item.name}\n")
                   builder.append("Price: ${item.price}\n")
                   builder.append("Purchase Date: ${item.purchaseDate}\n")
                   builder.append("Warranty (months: ${item.warrantyMonths}\n")
                   builder.append("Receipt Image Path: ${item.receiptImagePath}\n\n")
               }
```

```
            AlertDialog.Builder(this)
                .setTitle("Saved Items")
                .setMessage(builder.toString())
                .setPositiveButton("OK", null)
                .show()
        }
    }

    edtPurchaseDate.setOnClickListener {
        showDatePicker(edtPurchaseDate)
    }

    btnCaptureImage.setOnClickListener {
        val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
        startActivityForResult(intent, 1)
    }

    btnSave.setOnClickListener {
        val name = edtName.text.toString()
        val price = edtPrice.text.toString()
        val purchaseDate = edtPurchaseDate.text.toString()
        val warrantyMonths = edtWarrantyMonths.text.toString()

        if (name.isEmpty() || price.isEmpty() || purchaseDate.isEmpty() || warrantyMonths.isEmpty() || receiptImagePath.isEmpty()) {
            Toast.makeText(this, "Please fill all fields and capture receipt", Toast.LENGTH_SHORT).show()
        } else {
            val success = dbHelper.insertItem(
                name,
                price.toDouble(),
                purchaseDate,
                warrantyMonths.toInt(),


                if (success) {
                    Toast.makeText(this, "Item saved!", Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText(this, "Error saving item", Toast.LENGTH_SHORT).show()
                }
            }
        }

        checkExpiringWarranties()
    }
     private fun showDatePicker(editText: EditText) {
        val calendar = Calendar.getInstance()
        val datePicker = DatePickerDialog(
            this,
            { _, year, month, dayOfMonth ->
                val selectedDate = "${year}-${month + 1}-${dayOfMonth}"
                editText.setText(selectedDate)
            },
            calendar.get(Calendar.YEAR),
            calendar.get(Calendar.MONTH),
            calendar.get(Calendar.DAY_OF_MONTH)
        )
        datePicker.show()
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == 1 && resultCode == RESULT_OK) {
            val bitmap = data?.extras?.get("data") as Bitmap
            imageView.setImageBitmap(bitmap)
            saveImageToInternalStorage(bitmap)
        }
```

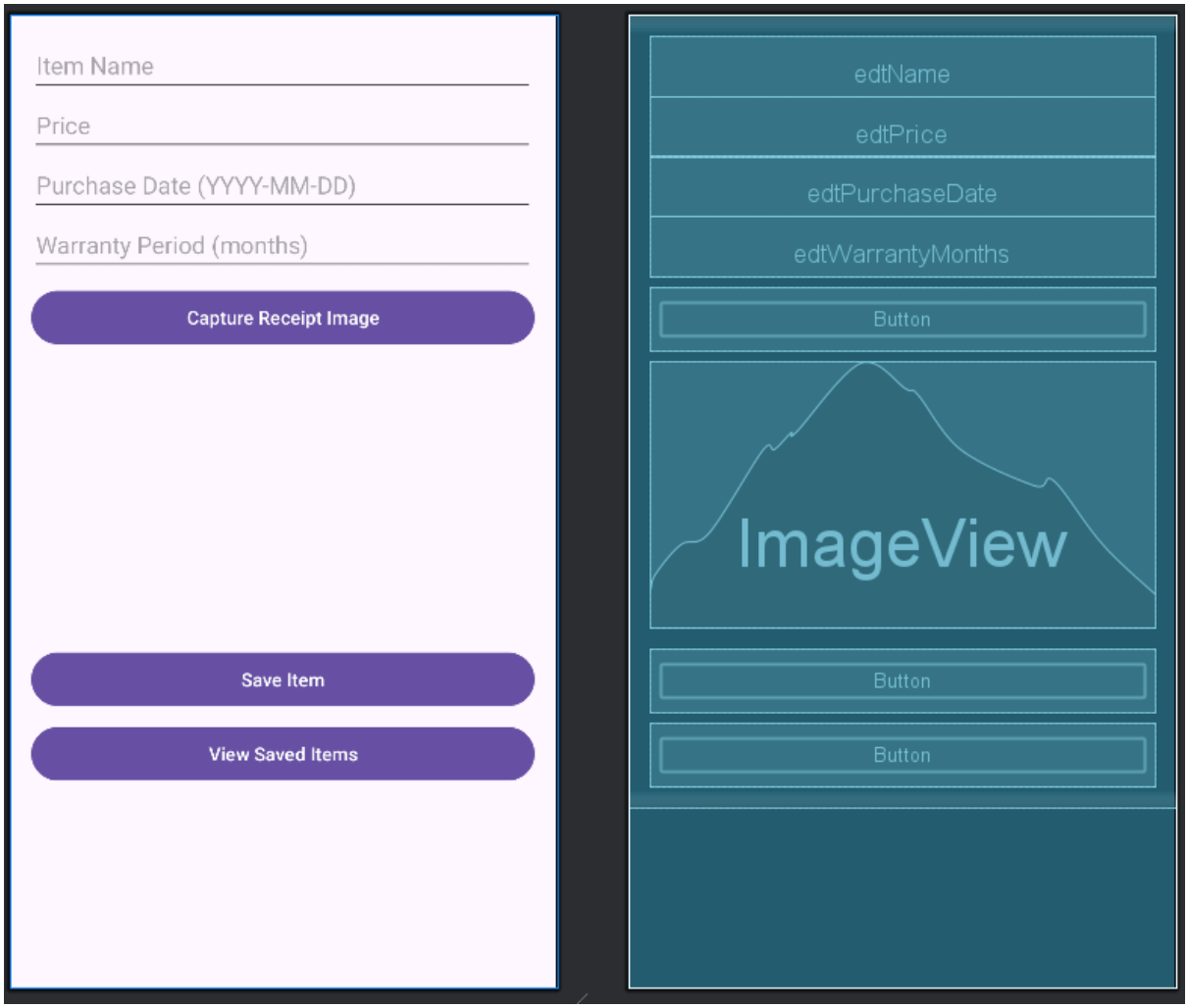
```
private fun saveImageToInternalStorage(bitmap: Bitmap) {
    val filename = "receipt_${System.currentTimeMillis()}.jpg"
    val file = File(filesDir, filename)
    val outputStream = FileOutputStream(file)
    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)
    outputStream.flush()
    outputStream.close()
    receiptImagePath = file.absolutePath
}
    private fun checkExpiringWarranties() {
    val items = dbHelper.getAllItems()
    val sdf = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())
    val today = Calendar.getInstance()

    val expiringItems = items.filter {
        val purchaseDate = sdf.parse(it.purchaseDate)
        val expiryDate = Calendar.getInstance()
        expiryDate.time = purchaseDate!!
        expiryDate.add(Calendar.MONTH, it.warrantyMonths)

        val diff = expiryDate.timeInMillis - today.timeInMillis
        diff in 0..(7 * 24 * 60 * 60 * 1000) // within 7 days
    }

    if (expiringItems.isNotEmpty()) {
        AlertDialog.Builder(this)
            .setTitle("Warranty Expiry Alert")
            .setMessage("Some items' warranties are about to expire soon!")
            .setPositiveButton("OK", null)
            .show()
    }
```

DBHelper.kt

```
package com.example.storetracker

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import android.content.ContentValues

class DBHelper(context: Context) : SQLiteOpenHelper(context, "store_inventory.db", null, 1) {

    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL(
            "CREATE TABLE items (" +
                    "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "name TEXT, " +
                    "price REAL, " +
                    "purchase_date TEXT, " +
                    "warranty_months INTEGER, " +
                    "receipt_image_path TEXT" +
                    ")"
        )
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS items")
        onCreate(db)
    }
```

```
fun insertItem(
        name: String,
        price: Double,
        purchaseDate: String,
        warrantyMonths: Int,
        receiptImagePath: String
    ): Boolean {
        val db = this.writableDatabase
        val contentValues = ContentValues()
        contentValues.put("name", name)
        contentValues.put("price", price)
        contentValues.put("purchase_date", purchaseDate)
        contentValues.put("warranty_months", warrantyMonths)
        contentValues.put("receipt_image_path", receiptImagePath)
        val result = db.insert("items", null, contentValues)
        return result != -1L
    }
    fun getAllItems(): List<Item> {
        val items = mutableListOf<Item>()
        val db = this.readableDatabase
        val cursor = db.rawQuery("SELECT * FROM items", null)
        if (cursor.moveToFirst()) {
            do {val item = Item(
                    id = cursor.getInt(0),
                    name = cursor.getString(1),
                    price = cursor.getDouble(2),
                    purchaseDate = cursor.getString(3),
                    warrantyMonths = cursor.getInt(4),
                    receiptImagePath = cursor.getString(5)
                items.add(item)
            } while (cursor.moveToNext())
        }
        cursor.close()
        return items
```

ActivityMain.xml



REFERENCES

1. Android Developers. (n.d.). *Build your first app*. Retrieved from https://developer.android.com/training/basics/firstapp
2. Android Developers. (n.d.). *Data Storage Using SQLite*. Retrieved from https://developer.android.com/training/data-storage/sqlite
3. Android Developers. (n.d.). *Using the Camera*. Retrieved from https://developer.android.com/training/camera/photobasics
4. Vogella, L. (2023). *Android SQLite database tutorial*. Retrieved from https://www.vogella.com/tutorials/AndroidSQLite/article.html
5. Room Persistence Library. (n.d.). *Android Jetpack Room*. Retrieved from https://developer.android.com/jetpack/androidx/releases/room
6. Stack Overflow. (n.d.). *Best practices for Android app architecture*. Retrieved from https://stackoverflow.com/questions/39814549/android-app-architecture-best-practices
7. Choe, S. (2020). *Effective Mobile App Design for User Engagement*. Mobile UX Review Journal, 12(3), 45-52.
8. Rashid, S., & Sharma, R. (2021). *A Study on Digital Receipts and Warranty Tracking Apps*. International Journal of Computer Applications, 183(29), 15-20.