University of Bahrain
College of Information Technology
Department of Computer Science

THIQA: AI-POWERED RECEIPT AND WARRANTY MANAGEMENT SYSTEM

Prepared by

Hussain Ali Nooh 202106117

Ali Abbas Ali 202107809

Haitham Monia 202107520

For

ITSE 498

Senior Project

Academic Year 2024-2025-Semester 2

Project Supervisor: Dr. Ahmed Fahad

Dr. Amal Shaheen

Date of Submission : 5/5/2025

Abstract

Managing physical and digital receipts as well as product warranties is quite difficult for small companies and people. Often laborious, error-prone, and resulting in lost financial data or missed warranty claim chances, manual tracking techniques are often time-consuming. Digital papers grow more complicated as receipts fade or disappear, warranty information is forgotten, and more of them accumulate. Proposed as an AI-driven mobile app, Thiqa is meant to solve these problems by offering a centralized, automated system for warranty and receipt management. Thiqa intends to simplify the whole procedure. Core functions are timely notifications and reminders for approaching warranty expirations, digital storage of warranty information, smart expense categorization, automatic extraction of important data points (vendor, date, total amount, line items), and capturing of receipt images. The design, implementation, and assessment of the Thiqa system are covered in this paper together with its architecture, technical decisions, and possible future growth.

Acknowledgments

We extend our gratitude to those who contributed to the successful completion of this project. We extend our deep appreciation to our co-supervisor, Dr. Amal Shaheen, and our supervisor, Dr. Ahmad Fahad, for their astute feedback, unwavering support, and invaluable guidance during all stages of our research and development. Their expertise and assistance contributed to the formation of this initiative. We extend our sincere gratitude to those who participated in the survey, as well as to individuals who generously contributed their time and perspectives during the requirement elicitation phase. We are also grateful to the retailers whose cooperation and insights proved invaluable during our research. Our comprehension of the problem domain and the specification of the system's requirements relied on their willingness to share their experiences and needs. Furthermore, we express our gratitude to anyone who offered assistance or advice throughout the process.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter presents the project problem statement, which addresses the ongoing issues people have in handling receipts and product warranties after purchase. Tangible receipts and warranty papers are often lost, damaged, or poorly organized despite their vital part in enabling returns, warranty claims, budgeting, and expense tracking. Users still find significant challenges in properly handling scattered digital receipts and warranty-related emails even with the many digital transactions. The project presents Thiqa, an AI-enhanced mobile app meant to offer a centralized, automated, user-centric platform for the management of receipts and warranties, so tackling these issues. Reducing manual effort through smart data extraction, improving financial record accuracy, and providing integrated tools for warranty monitoring and peer-to-peer warranty transactions are among the goals of the project. This project is important since it can enable consumers with better financial control and help to simplify post-purchase paperwork, so correcting major flaws in current solutions. Furthermore, by replacing antiquated, paper-based procedures with an intelligent, totally digital ecosystem for managing vital post-purchase documents, Thiqa helps to further the digital transformation vision. Finally, the report outline offers a summary of the main chapters comprising literature review, project methodology, system analysis and requirements, design and implementation, testing, and conclusion.

1.1 Problem Statement

Outdated, scattered systems cause consumers major difficulties in handling post-purchase paperwork. While digitalization is changing things, there is still no coherent, smart way to arrange, monitor, and relocate warranties or receipts. These problems can be divided into the following sub-problems:

- Receipts are still mostly issued in paper form, which are susceptible to loss or damage. Even when there are digital receipts, they are usually spread out over apps, emails, or platforms, which complicates retrieval and organization.
- Existing tools for managing receipts and warranties often fail to process diverse formats, low-quality images, and languages such as Arabic. This results in inaccurate data extraction and limited automation.
- There is no consistent, verified way to check ownership or warranty status, nor for safely issuing or transferring warranty entitlements during peer-to-peer product transfers. This fosters second-hand market inefficiencies and abuse.

1.2 Project Objectives

The aim of this project is to form, build, and evaluate Thiqa, an artificial intelligence mobile app simplifying warranty and receipt management to improve the post-purchase experience for both companies and consumers subsequently Thiqa directly significantly paper waste aligning with national sustainability goals. Developing a centralized platform for organizing and securing purchase-related documents, implementing an exclusive AI-driven OCR engine optimized for multilingual receipts (including Arabic) and handwritten input designing automated expense categorization and warranty tracking features, and enabling users to create, issue, and securely transfer authenticated warranties in peer-to-peer transactions are among the goals of the project. Other objectives included providing a user-friendly, scalable interface, guaranteeing little user input via smart automation, and including multilingual support. Moreover, the warranty system employs cryptographic authentication to prevent forgery, though compliance with local regulation remains open challenge and require future legal and standardizing considerations, Beyond commercial use, widespread adoption of the app could position it as a catalyst for green tech adoption in Bahrain.

1.3 Relevance/Significance of the project

The difficulties of managing receipts and warranties affect a broad spectrum of users from individual consumers to enterprises, therefore stressing the great demand for a quick, dependable solution. An artificial intelligence-powered mobile app called Thiqa address this gap by replacing scattered, paper-based methods with an intelligent, centralized platform for managing and storing post-purchase documents. Its support for secure, peer-to-peer warranty creation and transfer adds a unique feature missing from current solutions, therefore promoting trust and responsibility in informal and second-hand transactions. Although its fit with fintech and digital transformation helps it to be forward-looking tool in finance management, the app's scalability and multilingual make it available across areas. In the end, Thiqa's influence goes beyond convenience; it enables users more financial control, helps environmentally sustainable practices by lowering paper need contributing to the evolution of secure, paperless ecosystems.

1.4 Report Outline

Chapter 2 offers a Literature Review, therefore placing the study within current research and practices on receipt and warranty management following this introduction and acknowledgments. Project Management is covered in Chapter 3; it emphasizes project activities, the chosen process model, risk management, and the project activity plan. Chapter 4 then describes the Requirement Elicitation process, describing the approach, survey results, and the resulting functional and non-functional requirements. Presenting the selected microservices architecture, essential components, data design, and UML diagrams, Chapter 5 covers System Design. Covering system development, component integration, testing phases, findings, and a discussion of the system's strengths and weaknesses, including usability and user experience testing. Chapter 6 addresses System Implementation and Testing. Finally, Chapter 7 offers the Conclusion and sketches possible Future Work for the project.

Chapter 2
Literature Review

2.1 The Hidden Cost of Receipts: Post-Purchase Management Challenge

Particularly with regard to the efficient handling of receipts and product warranties, the time following a consumer finishes a purchase transaction is a vital but sometimes difficult phase for both consumers and companies. Necessary papers, receipts are proof of purchase for returns, exchanges, warranty claims, expense tracking, and personal budgeting (FasterCapital, n.d.). On the one hand, warranties are a vital consumer protection tool against product flaws as well as a promise from the manufacturer or retailer about product quality and support (FasterCapital, n.d.). However, conventional techniques depending on paper documentation are riddled with natural weaknesses. Physical receipts deteriorate, readily lost or misplaced, and recovering particular papers can be irritating and ineffective (Stephen & Mathivanan, 2017). This often results in customers being unable to exercise their rights for returns, access warranty services, or keep correct financial records. The scale makes the issue worse. Retailers in the United States alone produce millions of pounds of receipt paper each year, a large percentage of which turns straight waste (Stephen & Mathivanan, 2017). This not only raises environmental issues but also underlines the inefficiency of the conventional system. Thus, the fundamental issue is creating a strong recordkeeping system that improves the usability, accessibility, security, and sustainability of post-purchase documentation for consumers, businesses, and financial institutions (Stephen & Mathivanan, 2017). This literature review critically looks at the development of solutions tackling this issue, evaluates the features and constraints of present digital and AI-powered systems, and points out the precise gaps the suggested Thiqa system seeks to close.

2.2 The Evolution from Paper to Digital: Receipt Management Solutions

Information systems theory's framework helps to fully grasp the shift toward digital post-purchase management by stressing the role of technology in improving the accuracy, timeliness, and accessibility of both corporate and personal information. This enhancement enables better operational efficiency and decision-making. The need for improved efficiency, more accessibility, and better data management has driven the change from inefficient paper-based systems to streamlined digital solutions. Often as basic desktop apps or components of personal finance software like Quicken and Microsoft Money, early efforts at digital receipt storage

surfaced in the late 1990s and early 2000s, allowing restricted manual entry of expenses and scanned files but lacking integration, portability, or real-time processing capabilities.

2.3 Limitations of Traditional and Basic Digital Methods

Paper receipts are fragile and poorly manageable (Stephen & Mathivanan, 2017). Early digital workarounds, such as storing photos, lack organization and search capabilities, providing only marginal benefit over physical storage. Moreover, the manual transfer of data from paper receipts to digital formats (e.g., spreadsheets) is naturally labor-intensive, time-consuming, and error-prone (Lin, Liu, & Lee, 2022), therefore greatly blocking efficient financial tracking and record-keeping (FasterCapital, n.d.).

2.4 Rise of Dedicated Digital Receipt and Warranty Platforms

Identifying these limitations, various digital solutions have emerged. Expensify and Zoho Expense Concur are business oriented platforms that provide advanced capabilities including receipt scanning using Optical Character Recognition (OCR), expense categorization, report generation, and integration with accounting systems platforms. Consumer oriented apps like Smart Receipts and Foreceipt offer comparable scanning and organizing tools customized for personal finance. Becoming a foundation, OCR technology automated data extraction to some degree by converting scanned images into machine-readable text (Lin, Liu, & Lee, 2022). Particularly in industries like hospitality, centralized digital receipt systems have been suggested and built, showing advantages including lower operational costs, less paperwork, better record-keeping accuracy, quicker customer service, and environmental sustainability by means of lessened paper use (Gill, 2023). Often, these systems use cloud platforms for accessibility and scalability (Gill, 2023). Government initiatives, such as Bahrain's Ministry of Health Green IT program, also illustrate these environmental benefits through the elimination of physical documents and the digitization of services like birth certificate issuance (Ministry of Health, 2024).

2.5 Why Current Systems Still Fall Short

Yet, notable restrictions remain despite these developments. First, many sites still depend quite much on manual user involvement often requiring users to actively scan or photograph every receipt and sometimes manually verify or classify the extracted information (Lin, Liu, & Lee, 2022). This point of friction still prevents regular use. Secondly, although useful, conventional OCR technology has natural restrictions. Often lacking contextual awareness to distinguish between comparable data types e.g., dates vs. prices it struggles with different receipt formats, poor image quality, handwritten text, complicated layouts, and other languages (DocuClipper,

2025). Moreover, many current solutions run as data silos, locking user data inside proprietary ecosystems and so preventing a unified view of financial activities or asset ownership. Although critics claim more automation could lower user control or create privacy issues, research indicate that digital tools' user engagement rises when they strike a balance between automation and openness and user agency. The design of digital receipt systems has to therefore carefully consider the trade-off between the advantages of automation and users' need for control and confidence.

2.6 The Buyer Burden: When Warranty Platforms Prioritize Business Over Consumers

Current digital warranty management systems such as those provided by Syncron, Claimlane, Intelli Warranty, and PTC (Claimlane, 2025; Intellinet Systems, 2025; iWarranty, n.d.) are mostly meant for companies (manufacturers, retailers). Their characteristics usually consist of automated warranty registration facilitation, claims management processes, analytics to spot product defect trends, parts inventory control, and occasionally consumer portals for claim submission. The main objectives are to lower warranty expenses for the company, increase operational efficiency, find warranty fraud (a significant cost estimated at 2-4% of revenue), improve product quality depending on claim data, and control supplier recovery. Although some provide customer-facing portals, these are interfaces to the company's system rather than tools for consumers to control all their warranties from many different suppliers in one location. As a result, consumers are often burdened with managing multiple platforms each tied to a specific manufacturer or retailer just to access their warranty or receipt information which clearly leads to a fragmented and inconvenient user experience.

2.7 The Consumer Gap in Warranty Management

From the point of view of the consumer, handling warranties is still challenging. Many customers forgo registration entirely since the process can be laborious, sometimes requiring manual forms or negotiating complicated internet portals. Many retailers in Bahrain further complicate this process by requiring consumers to create individual accounts on their platforms in order to register and store warranties digitally within their systems. Common annoyances include tracking expiration dates, knowing coverage terms across several products and manufacturers, and finding the required papers (receipt and warranty information) when a claim is required (FasterCapital,n.d.). Yet, Many retailers in Bahrain further complicate this process by requiring consumers to create individual accounts on their platforms in order to register and store warranties digitally within their systems. Often, this disparity causes customer annoyance,

challenges in using warranty rights, and possible loss of brand loyalty resulting from subpar after-sales experiences (Kanoon360, 2024).

2.8 Overcoming OCR Limitations: From Fuzzy to Precise

Systems of AI-powered OCR significantly outperform conventional techniques. AI can attain much greater accuracy in extracting data from different and difficult inputs, including many fonts, languages (like Arabic), handwritten notes, low-quality photos, and complicated layouts usually found on receipts, by using deep learning models trained on large datasets. Tabscanner's 2023 benchmark study found that deep learning-based OCR systems extracted total amounts and tax values across 1,000 sample receipts with up to 94% accuracy, significantly outperforming rule-based OCR engines averaging 72%. Importantly, artificial intelligence adds contextual awareness so that systems can distinguish between data fields, verify information, automatically classify expenses or product categories, and even more precisely identify line items (Tabscanner, 2024). By simplifying the data collecting process, this lessens the need for manual correction and verification (Lin, Liu, & Lee, 2022).

2.9 The Environmental and Health Imperative: Beyond Paper Waste

The harmless paper receipt has a significant environmental and health impact that is becoming more and more impossible to overlook. Resource use is on a huge scale. Annually, in the United States alone, the manufacture of paper receipts is projected to use billions of gallons of water (perhaps over 20 billion gallons according to some estimates) and more than 3 million trees (with some estimates going as high as 10 million). Worldwide statistics greatly increase this effect by supporting deforestation and depleting water supplies. Apart from the environmental damage, the chemical coatings applied on most thermal paper receipts pose a substantial health hazard. Of these receipts, an estimated 86% to 93% are coated with Bisphenol-A (BPA) or its more frequent replacement, Bisphenol-S (BPS). Though they are also recognized as endocrine disruptors, these compounds serve as color developers. Human exposure primarily occurs through dermal absorption when handling the receipts. Alarmingly, the concentration of BPA on a single receipt can be up to 1,000 times greater than that found in the lining of a food can or a plastic bottle. Studies show that retail employees, who handle receipts frequently, have significantly higher levels (over 30% more) of BPA and BPS in their bodies compared to the general population. Research suggests that even brief contact, as little as 10 seconds, can result in skin absorption exceeding established safe harbor levels. Factors like using alcohol-based hand sanitizers can further increase the absorption rate.. The chemical coatings also create an end-of-life problem. BPA and BPS cause most thermal paper receipts to be non-recyclable.

Trying to recycle them puts these dangerous phenols into the recycling stream by tainting the whole batch of paper pulp. As a result, these receipts usually find their way to land lifts where the BPS and BPA could seep into groundwater and soil. This links directly the chemicals generating health issues with the aggravation of the paper waste problem. The same components that endanger handling also hinder sustainable disposal, so converting a large-volume paper product into a continuous source of pollution and waste (Green America, 2020).

2.10 Identified Gaps and the Opportunity for Thiqa

The studies examined show a clear fragmentation in current systems, with technological developments unevenly spread across consumer and business needs, little integration of artificial intelligence, and a lack of comprehensive, lifecycle-based post-purchase management tools. A close reading of the present literature and available solutions uncovers many ongoing gaps and unfulfilled needs in post-purchase management, therefore highlighting a clear chance for an inventive solution such as Thiqa: Many systems still need considerable manual user effort for data entry, scanning, and categorization despite digital tools, therefore hindering adoption and completeness (Lin, Liu, & Lee, 2022). Consumers lack integrated, consumer-centric warranty management systems that are thorough enough to let them easily handle all their warranties tracking expiry, terms, claims and related receipts from several vendors in one unified platform (Search 1, 3 findings). Though AI-OCR is available, many consumer-facing applications depend on simple OCR, therefore missing the full potential of artificial intelligence for very accurate, contextual data extraction from several formats (including regional specifics like Arabic) and automated categorization (Lin, Liu, & Lee, 2022; Search 2 results). Largely unaddressed remains the secure and official transfer of current manufacturer/retailer warranty rights during second-hand sales or gift giving. Moreover, there are no specific, organized digital tools to handle informal P2P agreements connected to purchases (e.g., loan terms, trial periods). Current P2P technologies draw attention to possible security and trust issues such as data integrity and fraud that need to be handled for sensitive transactions such warranty transfers. Information stays locked inside certain applications or ecosystems, therefore blocking a whole perspective. Negative consumer experiences, extended wait times, and annoyance are all caused by this fragmentation together with manual procedures and challenges obtaining warranty services (Gill, 2023). Current systems have not completely addressed the inefficiencies and environmental impact of laborious manual processes and paper receipts (Stephen & Mathivanan, 2017; Gill, 2023).

2.11 Positioning Thiqa: Addressing the Gaps

By including advanced technologies inside a consumer-centric framework, the suggested Thiqa system is meant to specifically solve these noted constraints. Thiqa intends to provide a new, integrated solution for receipt and warranty management by building on the advantages of centralized digital systems and using cutting-edge artificial intelligence. Thiqa intends to significantly cut manual data entry using Point-of-Sale (POS) integration for automatic digital receipt/warranty capture and, more importantly, by using advanced AI-powered data extraction for non-integrated or paper documents. Aiming to exceed conventional OCR constraints (Lin, Liu, & Lee, 2022), this AI engine automatically extracts and organizes important information (items, dates, warranty terms, vendor details) with great accuracy, including handling complicated formats and languages like Arabic, so reducing user input. Unlike business-oriented websites, Thiqa will offer a single dashboard for consumers to handle receipts and monitor warranty expiration dates, terms, and associated papers across all their purchases reliably and securely, therefore filling a major gap noted previously. A fundamental Thiqa innovation is the creation of a safe, verifiable system for the peer-to-peer transfer of formal manufacturer/retailer warranty entitlement. This function seeks to offer a consistent method for transferring leftover warranty rights for second-hand or donated items, a major unmet need, therefore addressing the trust and security issues natural in transactions. Thiqa also intends to provide users with tools to record and control their informal P2P agreements connected to platform transactions, therefore organizing where none now exists. Thiqa aims to provide a smooth, efficient, and seamless post-purchase management experience by automating data capture, centralizing information, offering timely alerts (e.g., warranty expiry), and enabling safe transfers, therefore solving recorded consumer pain points.

2.12 Thiqa vs. The Field: A Comprehensive Comparison of Solutions

Managing the aftermath of a purchase, from handling receipts to tracking warranties, presents ongoing challenges for consumers and businesses alike. While the digital age has offered alternatives to cumbersome paper trails, existing solutions often address only parts of the problem, leaving users to grapple with fragmented systems and manual effort. A review of the current landscape highlights the need for a more integrated and intelligent approach.

The proposed Thiqa system emerges as a solution designed to provide a unified, consumer-centric platform for post-purchase management, aiming to overcome the limitations of existing methods. The following table compares Thiqa with the various solution categories and specific examples discussed in the literature review:

Feature / Solution Category	Traditional Methods (Paper)	Basic Digital Methods (e.g., Photos, Spreadsheets)	Dedicated Digital Platforms (Business-Focused: Expensify, Zoho Expense, SAP Concur)	Dedicated Digital Platforms (Consumer-Focused: Smart Receipts, Foreceipt)	AI-Enhanced Solutions (General Capabilities)	Thiqa System (Proposed)
Receipt Management	Physical storage, prone to loss/degradation, manual sorting	Unorganized digital files, lack of searchability	Sophisticated scanning (OCR), expense categorization, reporting,	Basic scanning (OCR), personal finance organization	Advanced AI-OCR (higher accuracy, contextual understanding), automated categorization	Advanced AI-OCR (high accuracy, handles diverse formats/languages incl. Arabic), POS integration for automation, AI insights & categorization
Warranty Management	Manual tracking is easy to lose warranty cards/receipts	No dedicated system, reliance on external reminders	Primarily business-centric (claims processing, analytics for business); limited consumer interaction portals	Limited dedicated warranty features, focus often on expense tracking over warranty lifecycle	May include predictive analytics for businesses, automated claim workflows	Integrated consumer-centric hub for all warranties from various vendors, tracks expiry/terms, stores documents, provides alerts
Manual Effort Required	High (manual data entry, physical organization, searching)	High (manual photo taking, manual data entry into spreadsheets, manual organization)	Moderate (scanning, verification, categorization often needed)	Moderate (scanning, manual data entry/verification)	Reduced compared to basic OCR, but still may require some verification and manual input for features beyond extraction	Minimally low (automatic capture via POS or highly accurate AI extraction), aims to eliminate manual data entry through advanced automation
Data Extraction Accuracy & Context	Very Low (manual interpretation)	Very Low (no automated extraction)	Moderate (basic OCR limitations with varied formats, poor quality, lack of context)	Moderate (basic OCR limitations with varied formats, poor quality)	High (AI-OCR overcomes many traditional OCR issues, contextual understanding)	Very High (Advanced AI-OCR specifically designed for diverse formats, languages, and contextual understanding, including regional specifics like Arabic)
Integration & Data Siloing	No integration	No integration, data highly fragmented	Integrated with accounting/ERP (business-focused), data often siloed within platform	Data often siloed within app, limited integration with external services	Varies depending on the specific solution, some integration capabilities exist but not in Bahrain	Aims for a unified dashboard, reducing data siloing for the consumer across receipts and warranties
Consumer-Centricity	Very Low	Very Low	Low (tools primarily designed for business operations and efficiency)	Moderate (focused on personal finance/receipts, less on integrated warranty management across different products/brands)	Varies, not always consumer-focused for integrated post-purchase management	High (designed as a central hub for all consumer post-purchase documentation and management, prioritizing user experience and control)

Peer-to-Peer Functionality (e.g., Warranty Transfer)	Non-existent / informal (no secure mechanism)	Non-existent	Not applicable (business-focused)	Not applicable	Not typically a focus	Core innovation: Secure and verifiable mechanism for the peer-to-peer transfer of formal manufacturer/retailer warranty entitlement, tools for managing informal P2P agreements
Proactive Management / Alerts	None	None	Some (e.g., fraud detection for businesses)	Limited (e.g., basic expense reminders)	Potential for predictive analytics, automated alerts (often business-focused)	Yes (support customized notifications)
Sustainability	Very Low	Low (Still require paper receipt)	Low(Still require paper receipt)	Low (Still require paper receipt to extract info)	Low (Still require paper receipt to extract info)	High (Support POS integration promotes a paperless approach)

Table 1: comprehensive comparison

2.13 One System, All Proofs: The Vision for a Centralized Purchase Documentation Hub

No government-authenticated system exists for standardized digital receipt and warranty tracking, forcing reliance on fragmented solutions that operate in isolated silos without verification (Claimlane 2025). This fragmentation enables four continued problems: prevalent receipt/warranty fraud, cumbersome manual verification processes, inefficient warranty transfers and lack of ability to verify and validate ownership of receipts/warranties seamlessly. A government-backed digital record validated by precedents like South Korea's Cash Receipt System and EU's Digital Product Passport (Kim & Park 2020; European Commission 2023) would solve these issues through cryptographic Point Of Sale authentication, creating immutable, instantly verifiable records (Gill 2023; DocuClipper 2025). Such a system would deliver 5 transformative benefits:

1. Instant Verification: Real-time authentication by consumers, businesses and authorities.
2. Purchase records cannot be lost or deleted.
3. Seamless Transfers: Irrefutable warranty coverage proof for second-hand sales (Experian 2023).
4. Standardized Rights Enforcement: System-guaranteed validity replaces manual checks.
5. Fraud Elimination: Tamper-proof cryptographic security prevents receipt/warranty forgery (Intellinet Systems 2025).

This ecosystem would revolutionize post-purchase management by making every transaction automatically documented, protected, and universally verifiable resolving current inefficiencies while establishing new standards for commercial trust.

2.14 Conclusion

In conclusion, the literature highlights significant progress in digital receipt and warranty management but reveals critical gaps concerning automation, consumer-centric integration, the utilization of advanced AI, and secure P2P functionalities, particularly for warranty transfers. Thiqa aims to bridge these gaps by offering an intelligent, automated, and integrated platform that addresses the full lifecycle of post-purchase documentation from a consumer perspective, positioning it as a necessary advancement in the field.

Chapter 3
Project Management

3.1 Agile Model

A software process model also called the Software Development Life Cycle (SDLC) is a methodical framework that describes the phases in developing a software system from initial planning to final deployment and maintenance. The Agile software development process model is an incremental and iterative way of creating software. Unlike conventional linear approaches like waterfall, Agile focuses on flexibility, teamwork, and ongoing feedback, therefore enabling adaptation all through the development lifecycle. Among the main features of Agile are the division of labor into brief cycles called sprints, a strong emphasis on customer cooperation and input, the capacity to react to evolving needs, and a dedication to regularly provide functional software. Common Agile frameworks are Scrum and Kanban, each offering particular rules and practices for controlling the development process. Agile fundamental tenet is to provide value to the consumer in small, manageable increments, therefore promoting continuous improvement and guaranteeing the final product closely fits user needs. Agile natural flexibility and iterative character are especially well-suited for the dynamic environment of developing a warranties and receipts management system like Thiqa, where fast changing technologies, shifting user needs for managing various document types, and the need for quick adaptation to feedback are top priorities and fit exactly with Agile core principles. The Agile process model was chosen for the Thiqa application development because of its appropriateness for iterative development, fast feedback cycles, and ongoing user participation. We tracked and visualized our development process and tasks using Trello, a digital Kanban board tool. By categorizing activities into lists and cards, Trello helped us to effectively control our workflow. We used tools including checklists, due dates, and labels to track progress, establish priorities, and guarantee team responsibility. This interactive and visual strategy reinforced Agile values of openness, teamwork, and constant development. Below are the main steps usually included in an Agile iteration, described in the framework of creating Thiqa application.

3.1.1 Plan

Often part of iteration planning, this stage has the team choosing and prioritizing tasks from the product specification to be finished in the next sprint. The team defines the sprint goal, talks

about the needs, and divides the chosen items into smaller, actionable tasks. This planning establishes the emphasis for the iteration and is team-based. The planning stage for Thiqa had the team looking over the Trello boards. Specific features such "Implement secure user authentication" or "Add expense categorization options" would be chosen for the next sprint depending on the general project objectives and input from stakeholders. Represented as Trello cards, these features would be shifted onto the sprint's assigned list. Sub-tasks were specified by checklists inside these cards, and due dates were set to direct the team's work within the sprint time frame.

3.1.2 Design

Design tasks run parallel to development inside each Agile iteration. This means designing or improving the technical architecture required for the features being developed in that sprint, user experience (UX), and user interface (UI). Design is evolving, changing with product development and response to comments. The design stage of Thiqa included building mockups and prototypes for new screens, the warranty information view and improving the flow for user interactions, the receipt scanning process. The screens and their flow were created using Figma.

3.1.3 Build

The team actively puts the intended and designed features into use during the build phase. This calls for coding, component integration, and producing the operating software increment for the sprint. It is a practical stage aimed at transforming needs and designs into functional components of the application. Core development effort for Thiqa was the build phase. Trello cards in the "In Progress" or "Doing" list reflected the actively constructed tasks, therefore enabling the team and stakeholders to monitor progress toward finishing the sprint's features and view the continuous development work.

3.1.4 Test

Testing is a constant activity woven into the build phase and carried out officially at its conclusion. It means checking that the created features function properly, fulfill the criteria, and are bug-free. Different kinds of tests unit, integration, and functional are carried out to guarantee the quality of the software increment.

Testing for Thiqa included checking the operation of newly constructed features including making sure the AI correctly pulls data from several receipt formats or that warranty expiration reminders are sent correctly. Testing duties probably showed on Trello as individual cards shifted to a "Testing" list. This guaranteed that features were exhaustively verified before being deemed finished for the sprint. Agile meets the goals of the SDLC providing functional, high-

quality software via a structured, yet flexible, life cycle by means of these continuous cycles of planning, design, construction, and testing.

3.1.5 Review

The team reviews the finished work with stakeholders at sprint's end. This is collecting feedback and showing the functioning software increment. The review offers openness and comments on the development status as well as allows for useful contribution that guides the course of upcoming sprints and the whole product.

3.2 Risk Management

Although Agile presents several advantages development, it is also linked with some risks that must be carefully controlled. One frequent difficulty is the possibility of timeline delays. Though Agile is iterative, unexpected development complexity or project scope uncontrolled expansion (scope creep) can cause projects to run longer than first expected (ValueCoders, n.d.; ProjectManager.com, n.d.). Although Agile encourages adaptability in reaction to changes, these changes have to be properly controlled inside each sprint to prevent delays in the whole project. Another major danger in mobile application projects including those using Agile approaches is budget overruns (ValueCoders, n.d.; ProjectManager.com, n.d.). Delays or scope creep may cause timelines to be extended, which could raise the related development expenses, particularly if more resources are required. Preventing expenses from rising beyond the initial projections depends on good budget control all through the Agile process. Sometimes, the fast development cycles and focus on quickly delivering functional software in Agile cause technical debt to build up. Shortcuts in code quality or design to meet tight deadlines cause this backlog of technical problems that must be addressed later to ensure the long-term maintainability and scalability of the application. In Agile development, a major difficulty is balancing the need for speed with the value of code quality. User adoption issues are still possible even with the inclusion of user input all during the Agile process. Users might be hesitant to use the final system if it does not properly satisfy the needs or expectations of their target audience. Comprehensive user research and ongoing testing are absolutely necessary to reduce this danger and make sure the application appeals to its intended users. All mobile apps, especially those managing sensitive data like receipts and warranty information, are seriously concerned about security risks. Mobile platforms offer special security issues, and flaws in the app's design or implementation could be exploited. To safeguard user data and preserve confidence, it is first importance to include strong security policies all over the whole Agile development lifecycle. Changes in platform guidelines, the arrival of new devices with different

specs, and regular operating system updates can all increase the difficulty of mobile app development. These changes must be monitored by developers to guarantee that the application stays functional and compatible across a wide spectrum of devices and software versions, which can occasionally affect project deadlines and budgets.

3.2.1 Risk Identification

Risk Type	Possible Risks
People	• The absence of team members from certain critical workshops and meetings may affect the development and progress of the project and slow it down. • The scarcity of knowledge and skills about the project and its nature may result in misunderstandings and errors that might lead to delays and reworks and potential project failures. • Unrealistic workload expectations or overworking may lead to burnout, increased error rates, and decreased retention and commitment.
Technology	• Incompatibility between new and current systems can cause integration challenges and delays. • Dependence on unstable or immature technologies can result in unexpected failures or rework. • Using new technology for the team.
Requirements	• Ambiguous or incomplete requirements can result in inaccurate deliverables and the need for significant modifications. • Without clear priorities, the team may prioritize less important features first. This might cause delays in the delivery of essential functionalities and decrease the overall impact of the project. • If requirements are not clearly stated or understood, the solution offered may fail to satisfy expectations. This could result in costly changes and reduced stakeholder satisfaction.
Financial	• If the initial estimation of budget is incorrect, the project may run out of finances before completion. This may result in restricted scope or delays while extra finance is acquired. • Additional expenses that were not originally planned for can extend the budget and have an influence on other project areas. These hidden costs may result in reduced quality or delivery speed. • A limited budget may reduce the ability to attract skilled professionals and prevent the project from acquiring essential tools and resources.
Change management	• People may be unwilling to adopt new methods and technologies due to fear, habit, or a lack of obvious advantage. This risk has the ability to limit project success and block benefit realization. • Ineffective change management approaches can cause misunderstandings and miscommunications, that could end up in errors and rework.

Table 2: Risk Identification

3.2.2 Risk Analysis

Risk Description	Probability	Effect
The absence of team members from certain critical workshops and meetings may affect the development and progress of the project and slow it down.	Medium	Moderate Impact
The scarcity of knowledge and skills about the project and its nature may result in misunderstandings and errors that might lead to delays and reworks and potential project failures.	Medium	Significant Impact
Unrealistic workload expectations or overworking may lead to burnout, increased error rates, and decreased retention and commitment.	Medium	Significant Impact
Incompatibility between new and current systems can cause integration challenges and delays.	Medium	Significant Impact
Dependence on unstable or immature technologies can result in unexpected failures or rework.	Medium	Significant Impact
Using new technology for the team.	Medium	Moderate Impact
Ambiguous or incomplete requirements can result in inaccurate deliverables and the need for significant modifications.	High	Significant Impact
Without clear priorities, the team may prioritize less important features first. This might cause delays in the delivery of essential functionalities and decrease the overall impact of the project.	High	Moderate Impact
If requirements are not clearly stated or understood, the offered solution may fail to satisfy expectations. This could result in costly changes and reduced stakeholder satisfaction.	High	Significant Impact
If the initial estimation of budget is incorrect, the project may run out of finances before completion. This may result in restricted scope or delays while extra finance is acquired.	Medium	Significant Impact
Additional expenses that were not originally planned for can extend the budget and have an influence on other project areas. These hidden costs may result in reduced quality or delivery speed.	Medium	Significant Impact

A limited budget may reduce the ability to attract skilled professionals and prevent the project from acquiring essential tools and resources.	Medium	Significant Impact
People may be unwilling to adopt new methods and technologies due to fear, habit, or a lack of obvious advantage. This risk has the ability to limit project success and block benefit realization.	Medium	Significant Impact
Ineffective change management approaches can cause misunderstandings and miscommunications, that could end up in errors and rework.	Medium	Moderate Impact

Table 3: Risk Analysis

3.2.3 Risk Plan

Risk Type	Mitigation
People	Implement robust resource planning; Provide adequate training and skill development; Establish realistic workload expectations and monitor team well-being; Foster a positive team environment and encourage open communication.
Technology	Conduct thorough compatibility testing before integration; Evaluate the stability and maturity of new technologies; Utilize pilot programs for new technology adoption; Provide comprehensive training on new systems and tools.
Requirements	Implement detailed and iterative requirements gathering processes; Establish a clear and agreed-upon prioritization framework; Conduct regular stakeholder reviews and validation of requirements and deliverables; Consider prototyping or proof-of-concepts for complex requirements.
Financial	Develop detailed and accurate cost estimations with expert input; Include contingency reserves for unforeseen expenses; Implement rigorous expense tracking and control mechanisms; Secure a sufficient financial buffer or access to additional funding if needed.
Change Management	Develop a comprehensive change management plan with clear communication strategies; Provide adequate training and support for new processes and

	technologies; Identify and empower change champions; Establish feedback mechanisms to address concerns and resistance.

Table 4: Risk Plan

3.2.4 Risk Monitoring

Risk Type	Indictor
People	Increased absenteeism or service; Noticeable decline in team performance or quality of work; Observable signs of stress or burnout; High team member turnover rate.
Technology	Frequent system errors or crashes; Difficulties and delays during system integration; Degraded system performance; Increase in support requests related to technology usage.
Requirements	Numerous change requests after requirements sign-off; Rework or required on delivered outputs; Missed project milestones or deadlines; Expressed dissatisfaction from stakeholders regarding delivered functionality.
Financial	Significant variances between actual and budgeted costs; Occurrence of unexpected or untracked expenses; Frequent requests for additional funding; Delays in acquiring necessary resources due to budget constraints.
Change Management	Overt or subtle resistance from individuals or teams to new ways of working; Misunderstandings or confusion regarding new processes or technologies; Increase in errors related to the adoption of changes; Low levels of engagement or adoption of new tools/methods.

Table 5: Risk Monitoring

3.3 Project activities Plan

Sprint	Start Date	End Date	Main Goal
1	25 Jan 2025	30 Jan 2025	Get familiar with Firebase and Google Cloud services.
2	2 Feb 2025	2 Feb 2025	Conduct meeting with the supervisor to refine the project functionalities and objectives

3	3 Feb 2025	5 Feb 2025	Start designing the architecture and database of the system and what services will be used for that
4	6 Feb 2025	11 Feb 2025	Ensuring the team has understood all the requirements and the services that are going to be used and get feedback
5	12 Feb 2025	15 Feb 2025	Preparing the tasks and process in Trello and prioritizing them to address the tasks of each sprint
6	16 Feb 2025	16 March 2025	Start working on the tasks within each sprint and continually updating the architecture based on the progress and feedback
7	17 March 2025	21 March 2025	Testing and ensures all functionalities are working as intended
8	26 March 2025	31 March 2025	Enhancing and refining user interface for better UI/UX experience
9	1 April 2025	7 April 2025	Preparing documentation on the APIs created.
10	10 April 2025	2 May 2025	Making sure all documents required are ready to be submitted

Table 6: Activities Plan

Chapter 4
Requirement Collection and Analysis

In this chapter the requirement elicitation will be discussed. The functional and non-functional needs provided the basis for the information collected, which was then converted into precise, quantifiable objectives. A mix of observation and survey-based techniques to gather both qualitative and quantitative user insights were employed to reach this. Drawing on the information gathered, we created user personas fictional characters standing for usual system users and stakeholders. These personas were designed to represent various user categories, including daily consumers, second-hand purchasers, and stores. By stressing user goals, issues, and expectations, they helped to steer the design process. We also built system models to show the system's structure and behavior. This covered UML diagrams to characterize system user interactions including use case, sequence, and package diagram. These models helped us to guarantee the design satisfies all stated criteria and to have a clearer picture of the internal architecture.

4.1 Requirement Elicitation

The approach applied to collect and specify the needs for the suggested system is covered in this part. A key stage in system development is requirement elicitation, which guarantees that the end product really meets the needs and issues of its intended users. The approaches examined for requirements elicitation included direct observations and questionnaires stressing both quantitative and qualitative data collecting strategies. Observations about typical challenges people in the management and retrieval of personal receipts and warranties experience guided initial identification of the problem area. Ultimately, observations and a planned questionnaire survey were chosen as the main elicitation techniques for this project. While the survey allowed statistical analysis of user needs, observations gave qualitative insights into user behavior and environment. Often because of physical misplacement, damage, or disorganization, many people reported finding it difficult to quickly find or access these vital papers when required. The survey aimed to obtain comments straight from possible users about their experiences with receipt and warranty management as well as their expectations for a technological solution. This approach was selected to efficiently collect data from a possibly large and varied population, therefore enabling the identification of shared pain areas and preferred features. User expectations were found by statistical analysis of the survey responses.

The following study of the data gathered by this survey served as the basis for specifying the system's functional and non-functional needs. The next section provides a thorough description of the responses gathered. These insights were then converted into thorough system specifications guaranteeing the final design closely fits user-validated needs and preferences.

4.1.1 Survey Findings

Two hundred replies were gathered. Below are the results from related question. The survey results offer clear proof of user needs and strongly backs the first problem observations. For example, when asked How often do you need a receipt or warranty but cannot find it?, a notable percentage of respondents reported sometimes (44.5%) or often (28.5%). This emphasizes the common problem of finding papers as needed, therefore guiding the need for a readily searchable and available system.

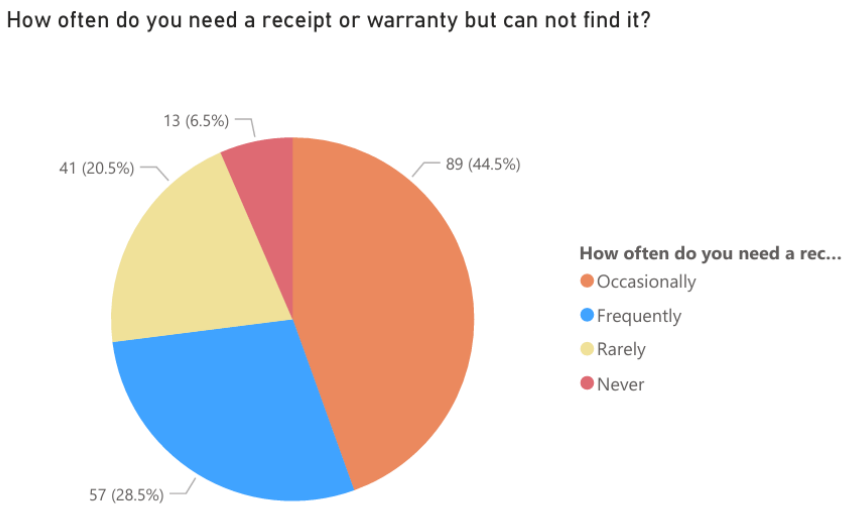

Figure 1: Survey question 1

The survey also looked at present practices for keeping and handling receipts and warranties. While 40% still depend on physical copies, the findings revealed that 29% use digital storage, 14% use emails, and 17% do not organize them at all. This varied terrain of present practices emphasizes the requirement of a flexible system that can fit different input techniques, e.g., scanning physical receipts, importing digital documents and emails and offer a centralized, ordered repository.

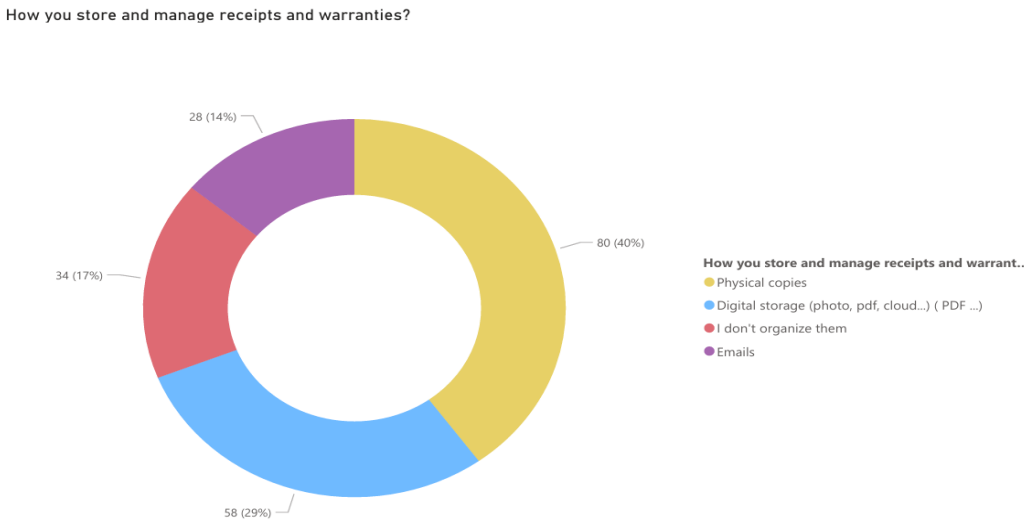

Figure 2: Survey question 2

The survey also measured the user's desire for a technical solution, which is important. Asked "How likely would you use a mobile app to manage your receipts/warranties?", a total of 79% of those polled said they were either very likely (42%) or somewhat likely (37%) to use such an app. This strong positive reaction confirms the need for a digital solution and supports the choice to create a mobile application as the main platform.

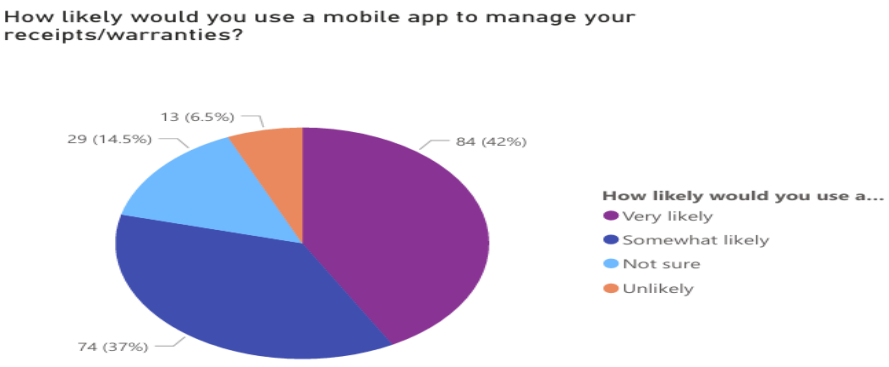

Figure 3: Survey question 3

The survey also questioned specific features desired by users for both receipt and warranty management systems:

6. Rank the features of a receipt system that would be MOST valuable to you:

- Receipts are instantly stored at the time of purchase: This feature was ranked as the most valuable by a significant margin, with 47% ranking it as their 1st choice. It also received substantial support as a 2nd choice (22%). This highlights the strong user need for effortless and immediate digital capture of receipts.
- Scanning physical receipts to be stored digitally with AI Auto-extraction: This came in as the second most valued feature, with 24% as a 1st choice and 26% as a 2nd choice. This indicates the importance of supporting existing physical receipts and the desire for automated data extraction to minimize manual entry.
- Quick search for specific receipts or items: Ranked as the 3rd choice by 25% of respondents, and receiving consistent support across other rankings, this confirms the critical need for efficient retrieval of documents.
- Detailed insights about spending (financial tracking): This feature garnered moderate interest, with 10% as a 1st choice and 21% as a 2nd choice, suggesting it is a valuable but not the most essential feature for the initial system.
- Categorisation of your items: This feature was ranked lower in priority compared to instant storage and scanning, with a higher percentage of respondents ranking it as their 4th (32%) or 5th (27%) choice. While still desired, it appears users prioritize the initial capture and searchability more highly.

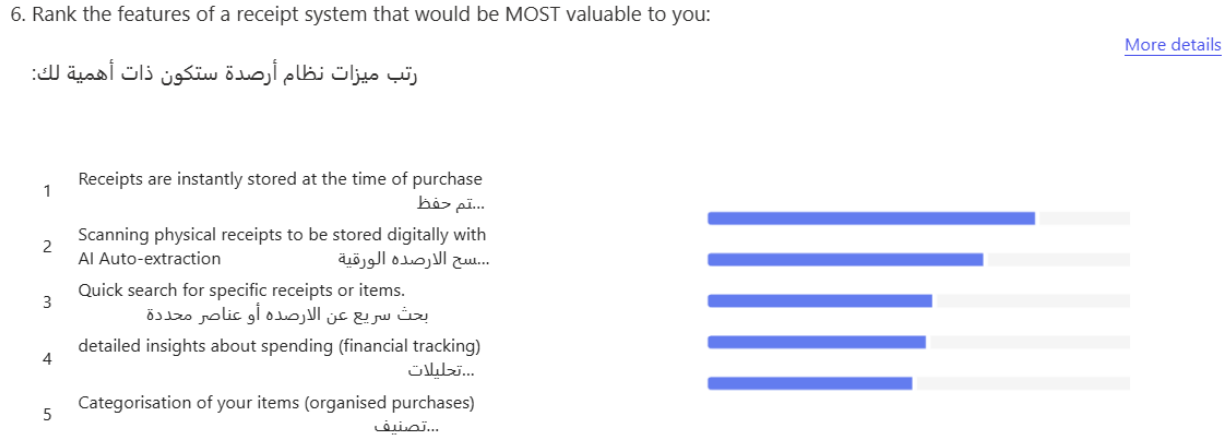

Figure 4: Survey question 4

7. Rank the features of a warranty system that would be MOST valuable to you?

- Warranties are never lost: This was overwhelmingly the top-ranked feature, with 60% of respondents listing it as their 1st choice. This directly addresses the core problem of misplacing or losing warranty information.
- Quick search for specific warranties: This feature was the second most valued, with a combined 59% ranking it as either their 1st (17%) or 2nd (42%) choice. This reinforces the need for easy retrieval of warranty details when required.
- Warranty expiry reminders**:** Ranked as the 3rd most valuable feature by 27% and receiving consistent support across other rankings, this indicates that users desire proactive notifications to avoid missing out on warranty claims.
- Option to transfer authenticated warranties: This feature was ranked lower in priority, with the highest percentages in the 4th (37%) and 3rd (28%) choice categories. While potentially useful, it is not perceived as a primary necessity by the majority of users.
- Option to issue P2P agreements (user to user): This feature received the lowest ranking among the warranty features, with a large majority ranking it as their 4th (53%) or 3rd (25%) choice. This suggests it is a niche feature not widely considered essential by potential users. possibly due to unfamiliarity with P2P warranty options, as this concept is relatively new in the field.

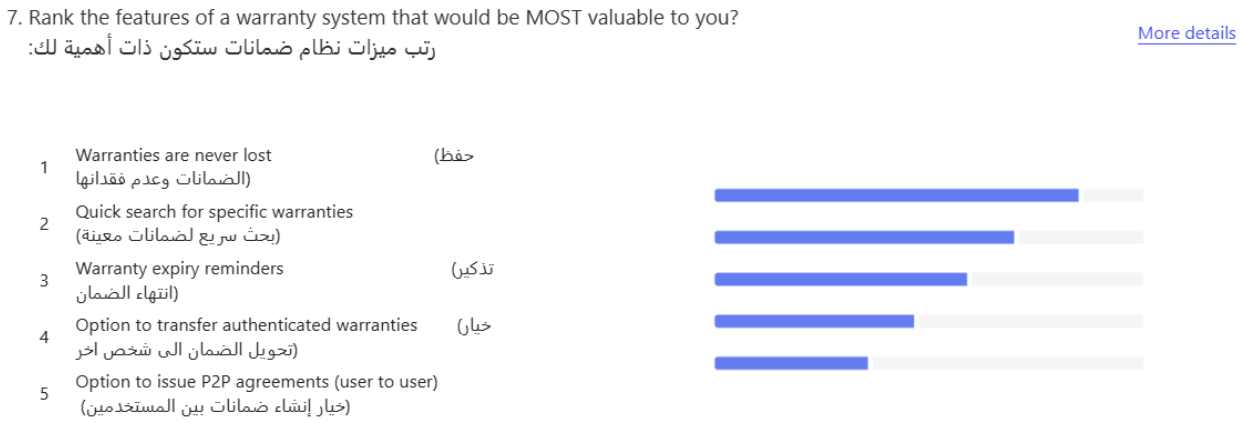

Figure 5: Survey question 5

8. How useful do you think this application would be?

With an average rating of 4.52 out of 5, the rating question on the perceived usefulness of the application produced a very positive response. Of the 200 people who answered, 132 (66%) rated the application very useful at Level 5; another 36 (18%) rated it Level 4. Just a tiny percentage of respondents said it was worse. The high average score and the strong clustering of responses at the upper levels suggest a clear perceived value and need for such a system among the target users.

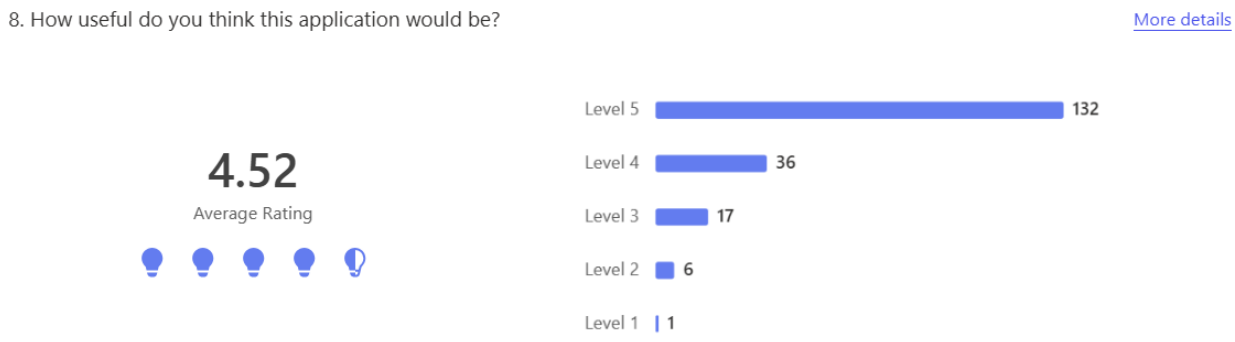

Figure 6: Survey question 6

The thorough ranking of features in questions 6 and 7 offered vital insights on user priorities. Regarding the receipt system, the focus on quick storage and simple scanning using artificial intelligence extraction directly reflects fundamental functional needs. There is also a clear order for the need of fast search capabilities. Although financial tracking and categorization are preferred, they seem to be secondary to the basic tasks of capture and retrieval. Likewise, for the warranty system, the main functions are dictated by the total importance given on avoiding loss and allowing rapid search. Another valuable aspect that ought to be included is warranty expiration alerts. Although maybe creative, characteristics like P2P agreements and warranty transfer are not regarded as necessary by most and might be looked at for later development stages. The project's possible influence is confirmed by the mostly favorable reaction to the perceived utility of the application in question 8, which also underlines the need of creating a system that properly handles the noted pain areas. Ultimately, the survey results offered priceless quantitative data that confirmed first impressions, highlighted important user issues, verified the need for a technological solution, and most importantly, gave priority to the particular qualities most valued by possible users. This information provides a strong basis for specifying the functional and non-functional criteria of the proposed receipt and warranty

management system, so guaranteeing it is created to satisfy the actual needs and expectations of its intended users.

4.2 System Requirements

The system requirements for the suggested receipt and warranty management application are described here. Produced straight from the knowledge acquired during the requirement elicitation phase, especially the results of the user survey described in Section 4.1. This section converts user needs and priorities into particular technical specifications. The two main categories of the requirements are functional ones, which specify the particular behaviors and functions the system must perform, and non-functional ones, which define the quality attributes, constraints, and characteristics of the system, such as performance, security, usability, and reliability. These criteria guide the design and development of a system that properly handles the issues and fulfills the expectations of its intended users. Building the first list and adding the user survey feedback, this part clarifies the functional needs. These criteria define the fundamental features and interactions the system has to enable to properly meet user needs and priorities.

4.2.1 Functional Requirements:

Based on the analysis of user needs and priorities, the system shall provide the following functional capabilities:

4.2.1.1 Receipt Management:

- FR-RM-1: The system shall let users manually upload digital receipt files (PDF, image formats like JPEG, PNG, JPG) from their device or capture photos of physical receipts using their device's camera. The system shall use artificial intelligence (AI) to automatically extract key information from scanned receipts, including but not limited to vendor name, date of purchase, total amount, individual items bought, and price per item.
- FR-RM-2: The system shall securely store all captured and uploaded receipts in a centralized digital repository.
- FR-RM-3: Receipt Viewing: The system shall allow users to view digital copies of their stored receipts within the application interface.
- FR-RM-4: Users should be able to find particular receipts depending on criteria including vendor name, date range, extracted item name keywords, or total amount using the system's quick and efficient search feature.

- FR-RM-5: The system shall analyze receipt data to provide users with detailed insights into their spending patterns. This includes generating summaries and visualizations based on categories, time periods, and vendors.
- FR-RM-6: The system should intelligently categorize receipt items using AI.

4.2.1.2 Warranty Management:

- FR-WM-1: Using their device's camera or manual upload of digital warranty documents (PDF, image formats including PNG, JPG, JPEG), the system shall let users capture photos of physical warranty documents. Key warranty information including but not limited to product name, warranty period, expiry date, vendor, and terms and conditions shall be automatically extracted by the system using artificial intelligence.
- FR-WM-2: The system shall securely store all captured and uploaded warranty documents in a centralized digital repository.
- FR-WM-3: The system shall allow users to view digital copies of their stored warranty documents within the application interface.
- FR-WM-4: The system shall provide a fast and efficient search function allowing users to locate specific warranties based on criteria such as product name, vendor, purchase date, or expiry date.
- FR-WM-5: The system shall allow users to set customizable reminders for upcoming warranty expiry dates. The system shall send notifications to the user prior to the expiry date.
- FR-WM-6: The system shall provide a mechanism for users to initiate a warranty claim, potentially by generating a summary of warranty details and providing options to attach relevant documents.
- FR-WM-7: Warranty Renew: The system shall provide a mechanism for users to renew their warranty.
- FR-WM-8: The system shall provide functionality for users to transfer authenticated warranties to another user within the system, subject to verification and agreement processes.
- FR-WM-9: The system shall provide functionality for users to create and manage simple peer-to-peer agreements potentially including basic warranty-like terms between individuals.

4.2.1.3 Retailer Functionality:

- FR-RT-2: The system shall allow retailers to digitally issue receipts and warranties directly to user accounts at the time of purchase.
- FR-RT-3: The system shall allow retailers to receive, review, and manage warranty claims submitted by users through the system (Accept/Reject).
- FR-RT-4: The system may allow authorized retailers to view receipts and warranties that they have issued to users.
- FR-RT-5: The system may allow authorized retailers to delete receipts and warranties that they have issued (subject to business rules and user notification).

4.2.1.4 Admin Dashboard Functionality:

- FR-AD-1: The system shall display key aggregate statistics on an administrator dashboard, including the total number of registered users, the total number of receipts stored, the total number of warranties stored, and the total number of warranty claims initiated.
- FR-AD-2: The system shall provide administrators with visual analytics (e.g., charts, graphs) offering insights into trends related to receipts and warranties, such as document capture over time, distribution of document types, or claim trends.
- FR-AD-3: The system shall identify and display a list of top users based on criteria such as the number of receipts stored. The system shall also display the contact details (e.g., email address, associated user ID) for these top users.
- FR-AD-4: The system shall allow administrators to manually trigger the generation of spending or document-related insights for specified time periods (e.g., monthly, yearly) or specific custom dates. This serves as a backup mechanism in case of automated process failures.
- FR-AD-5: The system shall provide administrators with tools to run fraud detection analysis. This includes identifying potential high-risk users based on configurable criteria (e.g., unusual claim patterns, excessive document uploads). The system shall display potential high-risk users and allow administrators to view their details and manually flag their accounts for further investigation.

4.2.2 Non-Functional Requirements:

This section outlines the non-functional requirements for the proposed receipt and warranty management system. These requirements define the quality attributes, constraints, and characteristics that the system must possess to operate effectively, reliably, and securely, complementing the functional requirements detailed in Section 4.2.1.

The system shall adhere to the following non-functional requirements:

- NFR-PERF-1: The system shall retrieve and display a user's list of receipts or warranties within a maximum of 3 seconds under normal load conditions.
- NFR-PERF-2: The system shall return search results for receipts or warranties within a maximum of 5 seconds, even with a large volume of stored documents (e.g., up to 1000 documents per user).
- NFR-PERF-3: The system shall process and extract data from a scanned physical receipt or warranty using AI within a maximum of 10 seconds.
- NFR-SEC-1: User data, including uploaded documents and extracted information, shall be encrypted both in transit (when being uploaded or accessed) and at rest (when stored on servers).
- NFR-SEC-2: The system shall implement secure user authentication mechanisms (e.g., password, multi-factor authentication) to prevent unauthorized access to user accounts.
- NFR-SEC-3: The system shall ensure that users can only access and manage their own receipts and warranties, and retailers can only access documents they have issued (if retailer functionality is implemented).
- NFR-USAB-1: The system shall have an intuitive and easy-to-navigate user interface, designed for mobile devices, allowing users to quickly understand how to capture, view, search, and manage their documents.
- NFR-USAB-2: New users should be able to understand the basic functions of capturing and viewing documents within 15 minutes of their first use without requiring extensive training.
- NFR-REL-1: The system shall be available to users at least 99% of the time.
- NFR-REL-2: The system shall implement regular data backups to prevent data loss in case of system failures.
- NFR-REL-3: The system shall handle errors gracefully and provide informative feedback to the user in case of issues (e.g., upload failed, search returned no results).
- NFR-SCAL-1: The system architecture shall be designed to accommodate a growing number of users and an increasing volume of stored documents without significant degradation in performance.

- NFR-MAINT-1: The system code and architecture shall be designed to be easily maintainable and modifiable to facilitate future updates, bug fixes, and feature enhancements.
- NFR-PORT-1: The system shall be developed to be compatible with major mobile operating systems (e.g., iOS and Android).

4.3 Personas

Fictional representations of the target users and system stakeholders, user personas Survey results and user behavior observations among other data gathered during the requirement elicitation phase drive their creation. Personas help to humanize the user base, so facilitating the understanding of their needs, goals, pain points, and motivations. Designing for these particular people will help us make the system user-centric and properly handle the practical problems the target audience experiences.
The following user and stakeholder personas have been created depending on the survey results, the defined problem area, and the enlarged scope including possible retailer and peer-to-peer interactions:

Persona 1: Fatima

- Background: Fatima is a 35-year-old marketing manager living in Riffa. She uses her smartphone for most daily tasks such as managing finances and online shopping. She is busy with work and family life and values efficiency and organization.
- Demographics: Age 25-44.
- Goals:
    - Quickly find receipts for returns, exchanges, or expense reporting.
    - Ensure warranties for electronics and appliances are easily accessible when needed for repairs or claims.
    - Reduce physical clutter from paper receipts and warranty cards.
    - Have a reliable digital backup of important purchase documents.
    - Receive timely reminders for warranty expiries.
- Pain Points:
    - Physical receipts are easily misplaced or faded over time.
    - Email receipts get buried in a crowded inbox.
    - Finding a specific receipt or warranty can be time-consuming and frustrating.
    - Worries about missing a warranty expiry date.
- Current Method: A mix of keeping some physical receipts in a drawer, leaving others in shopping bags, and having digital receipts scattered across email folders.

- Valued Features:
    - Instant digital capture of receipts at the time of purchase.
    - Scanning physical receipts with accurate AI extraction.
    - Quick and effective search functionality for both receipts and warranties.
    - Automated warranty expiry reminders.
    - Secure and reliable digital storage.

Persona 2: Ahmed

- Background: Ahmed is a 58-year-old retired accountant living in Manama. He is comfortable using technology for communication and browsing but is not an early adopter of new apps unless they provide clear practical benefits. He is meticulous about keeping records, especially for significant purchases.
- Demographics: Age above 45
- Goals:
    - Have a single, organized system for all receipts and warranties.
    - Ensure important documents are safe from physical damage or loss.
    - Easily access warranty information if a product needs repair.
    - Understand his spending patterns for personal budgeting.
- Pain Points:
    - Managing a growing collection of paper documents is becoming cumbersome.
    - Concerned about the security of digital documents.
    - Finding specific documents in his current filing system can take time.
- Current Method: Primarily relies on physical filing cabinets and folders for organizing paper receipts and warranty cards.
- Valued Features:
    - Scanning physical receipts and warranties with reliable AI extraction.
    - Secure digital storage that guarantees documents are "never lost."
    - Quick search capabilities to easily locate specific items.
    - Warranty expiry reminders.
    - Spending insights for financial tracking.

Persona 3: Noora

- Background: Noora is a 28-year-old graphic designer in Muharraq who frequently buys and sells electronics, furniture, and other items on online marketplaces. She is always looking for good deals and aims to get the best value when selling her pre-owned items.
- Demographics: Age 25-44.
- Goals:
    - Provide proof of purchase or remaining warranty to potential buyers.
    - Receive valid warranty information when buying secondhand items.
    - Track the purchase history of items she intends to sell.
    - Facilitate the transfer of remaining warranties to the new owner.
- Pain Points:
    - Original receipts and warranties for items being sold are often lost or hard to find.
    - Buyers are sometimes hesitant without proof of purchase or warranty.
    - Transferring physical warranties is inconvenient.
    - Uncertainty about the validity of warranties on secondhand goods.
- Current Method: Relies on finding original paper or email receipts if available, or simply stating the item's condition and age.
- Valued Features:
    - Ability to easily share digital receipts/warranties with others.
    - Functionality to formally transfer authenticated warranties.
    - Secure storage of purchase history for all items.
    - Option to create simple P2P agreements for items without formal warranties.

Persona 4: Ali

- Background: Ali is a 48-year-old owner of a small electronics store in Isa Town. He manages sales, inventory, and customer service. He is interested in improving customer satisfaction and streamlining business processes.
- Demographics: Age 45+

- Goals:
    - Issue digital receipts and warranties directly to customers.
    - Reduce the administrative burden of managing paper warranties and claims.
    - Improve the efficiency of handling warranty claims.
    - Maintain a digital record of issued documents.
    - Enhance customer loyalty through a modern service offering.
- Pain Points:
    - Printing and storing physical receipts and warranties is costly and cumbersome.
    - Tracking warranty periods for multiple products and customers is challenging.
    - Processing warranty claims can be a manual and time-consuming process.
    - Customers losing their proof of purchase.
- Current Method: Issues physical receipts and warranty cards. Manages warranty claims manually based on physical documentation.
- Valued Features:
    - Ability to digitally issue receipts and warranties at the point of sale.
    - Systematic process for receiving and managing warranty claims (Accept/Reject).
    - Digital access to receipts and warranties issued to customers.
    - Efficient document management features for issued documents.

These personas provide a comprehensive view of the different types of users and stakeholders who would interact with the system, guiding the design to meet their specific needs and ensure the system's overall success and adoption.

4.4 System Models

This Section shows the system behavioral modeling using UML Use Case Diagrams and process specifications. Highlighting the fundamental system features, the Use Case Diagram offers a high-level view of the interactions between key actors consumers, retailers and the system. Section 4.4.2 elaborates on this with a sequence diagram's thorough scenarios for chosen use cases. Every use case offers a thorough knowledge of how users will interact with the system under normal and edge-case conditions by including the goal, involved actors, step-by-step flows, and exceptional situations.

4.4.1 UML Use Case Diagram

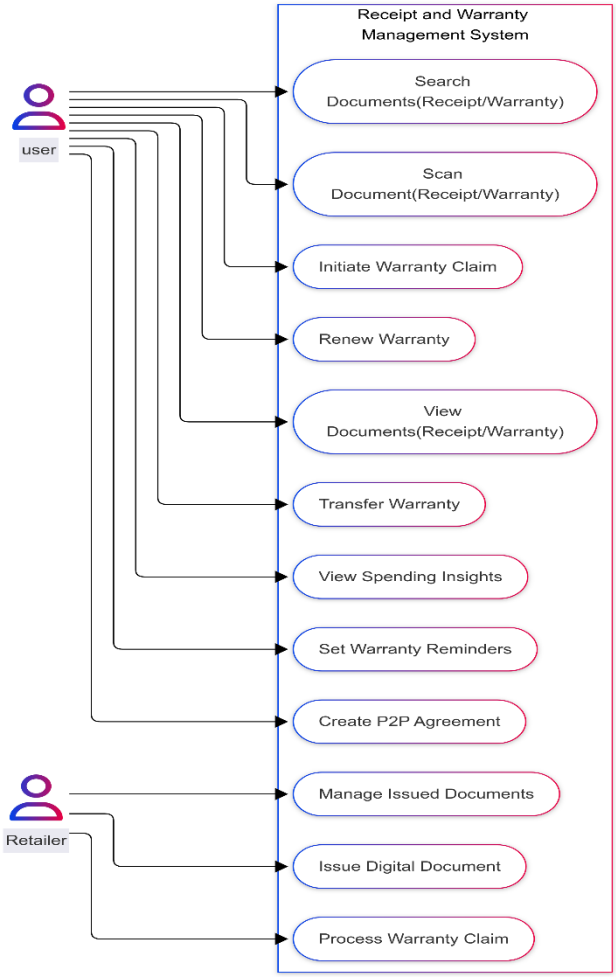

Figure 7: UML use case diagram

In software engineering, a Use Case Diagram is a behavioral diagram that describes a set of actions use cases that a system can perform in cooperation with outside users (actors). It offers a high-level perspective of the system features and helps to specify the system range. The diagram shown here shows the interactions between the main actors and the system. A rectangle denotes the system boundary, which the diagram shows. All the use cases included in the system functionality are contained within this boundary. Inside this boundary are the use cases shown by ovals. Every use case denotes a particular objective an actor wishes to reach with the system.

4.4.2 Process Specification

Key use cases found in the UML Use Case Diagram (Section 4.4.1) are thoroughly described and illustrated in this part. Every use case description specifies the objective, the actors engaged, the usual flow of events, and possible exceptional situations.

Use Case: Scan Document (Receipt/Warranty)

- **Goal:** The user wants to add a new receipt or warranty document to the system.
- **Actors:** User
- **Description:** This use case covers the different methods a User can employ to input receipt or warranty information into the system, including scanning physical documents, uploading digital files, or potentially forwarding emails. The system will process the input, extract relevant data using AI (for scans and emails), and store the document.
- **Scenario (Scanning Physical Receipt):**
    1. The User opens the system's mobile application.
    2. The User chooses the "Scan Physical Document" method.
    3. The system activates the device's camera or select from local storage.
    4. The User aligns the physical receipt within the camera frame and takes a picture.
    5. The system processes the image, performs AI extraction.
    6. The system displays the extracted information for review.
    7. The User reviews and confirms or edits the extracted data.
    8. The system saves the digital receipt and extracted information to the user's account.
- **Exceptional Scenarios:**
    - **ES-CD-1: Poor Image Quality:** If the scanned image is blurry, poorly lit, or the text is faded, the AI extraction may fail or produce inaccurate results. The system should notify the user and allow manual data entry or rescanning.
    - **ES-CD-2: Unsupported File Format:** If the user attempts to upload a digital file in a format not supported by the system, the system should display an error message and list supported formats.
    - **ES-CD-3: Extraction Errors:** Even with a clear image, the AI may misinterpret certain characters or fields. The system must allow the user to easily edit the extracted data before saving.
    - **ES-CD-4: Network Connection Failure:** If the network connection is lost during upload or processing, the system should inform the user and allow them to retry the operation when the connection is restored.

- **ES-CD-5: Storage Limit Reached:** If the user has reached their allocated storage limit, the system should notify the user and prevent further uploads until space is freed up or the limit is increased.

Figure 8: Scan Document sequence diagram

Use Case: View Documents (Receipt/Warranty)

- **Goal:** The user wants to view their stored documents.
- **Actors:** User
- **Description:** This use case allows the user to interact with their collection of stored receipts and warranties. They can browse their documents and view detailed information.
- **Scenario:**
    1. The User logs into the system.
    2. The User navigates to their list of stored receipts.
    3. The User selects a specific receipt.
    4. The system displays the details of the selected receipt.
- **Exceptional Scenarios:**
    - **ES-MD-1: Document Not Found:** If a document is somehow missing or corrupted, the system should display an error when the user attempts to view or manage it.

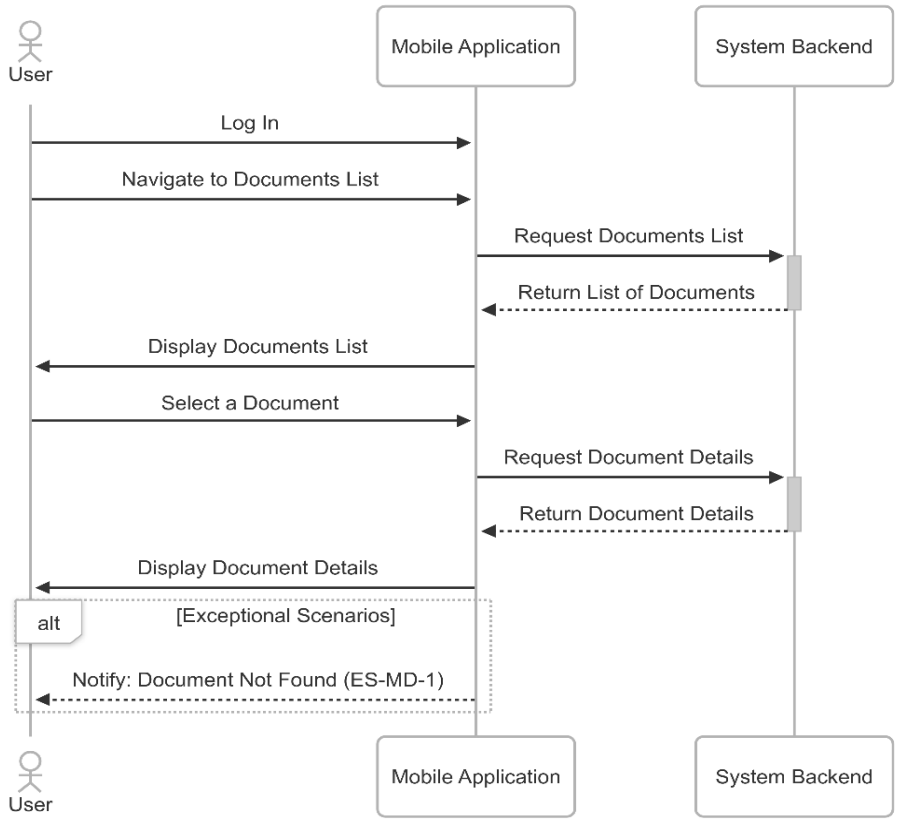

Figure 9: View document sequence diagram

Use Case: Search Documents (Receipt/Warranty)

- **Goal:** The user wants to quickly find a specific receipt or warranty.
- **Actors:** User
- **Description:** This use case provides the functionality for the User to search their stored documents based on various criteria such as vendor name, date, keywords from items, or expiry dates. The system should return relevant results efficiently.
- **Scenario (Searching for a Warranty):**
    1. The User logs into the system.
    2. The User navigates to the search function.
    3. The User enters a search term (e.g., "Samsung TV").
    4. The User specifies the document type (e.g., "Warranty").
    5. The User initiates the search.
    6. The system searches the user's stored warranties for matching documents.
    7. The system displays a list of search results.
    8. The User can select a result to view the document details.
- **Exceptional Scenarios:**
    - **ES-SD-1: No Results Found:** If the search criteria do not match any stored documents, the system should clearly indicate that no results were found and perhaps suggest alternative search terms or methods.
    - **ES-SD-2: Slow Search Performance:** With a very large number of documents, the search might take longer than expected. The system should provide feedback to the user that the search is in progress.

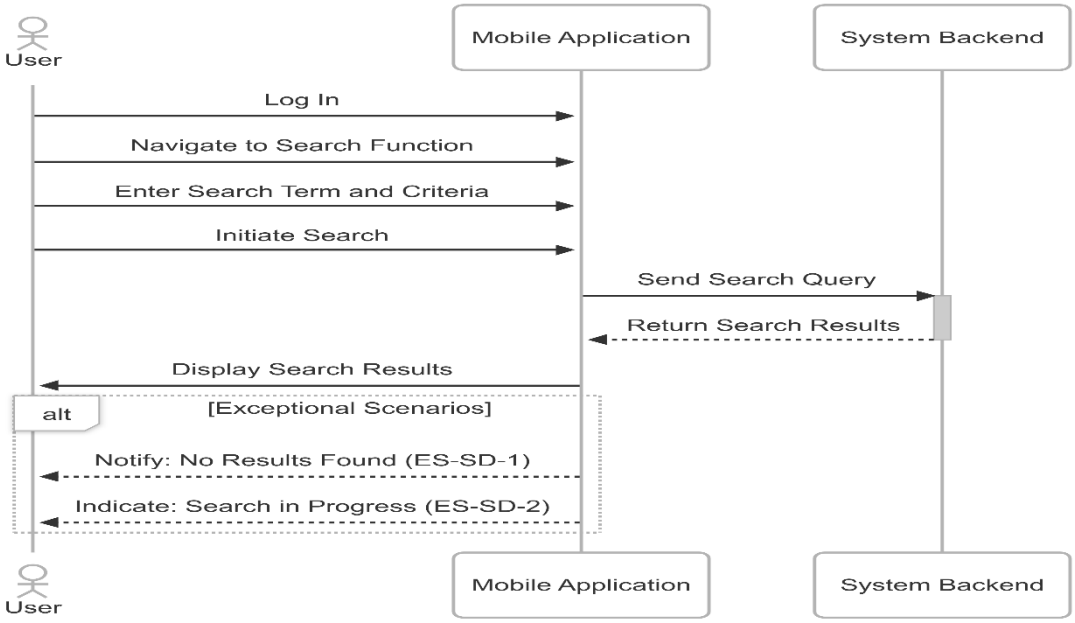

Figure 10: Search document sequence diagram

Use Case: View Spending Insights

- **Goal:** The user wants to understand their spending patterns.
- **Actors:** User
- **Description:** This use case enables the User to access visualizations and summaries of their spending based on the data extracted from their stored receipts. Insights can be presented by category, time period, or vendor.
- **Scenario (Viewing Monthly Spending by Category):**
    1. The User logs into the system.
    2. The User selects the "Spending Insights" feature.
    3. The system presents options for viewing insights (e.g., by month, year, category, vendor).
    4. The User selects "View by Month" and "By Category" for the current month.
    5. The system processes the receipt data for the current month.
    6. The system displays a chart or summary showing spending breakdown by category for the selected month.
- **Exceptional Scenarios:**
    - **ES-VSI-1: Insufficient Data:** If the user has very few stored receipts, the spending insights may be limited or not meaningful. The system should inform the user if there isn’t enough data to generate comprehensive insights.

- **ES-VSI-2: Data Inconsistencies:** If there are errors or inconsistencies in the extracted receipt data, the spending insights may be inaccurate. The system could potentially flag data points with low confidence or provide a way for users to correct data that impacts insights.

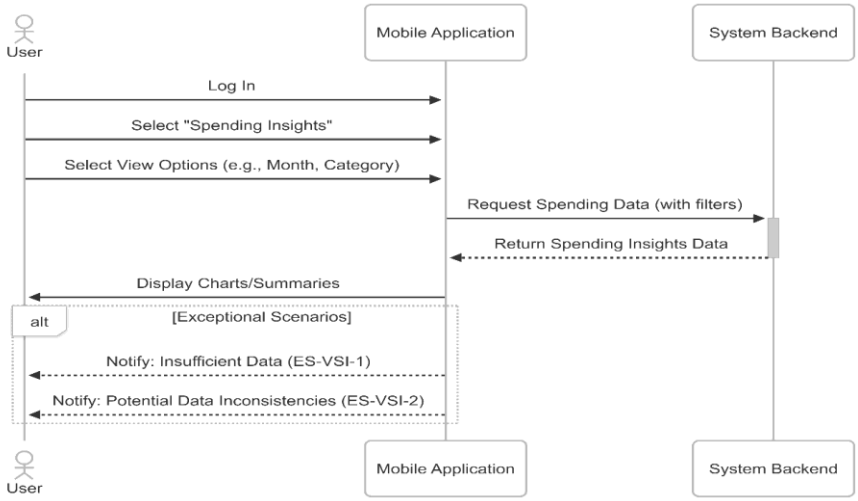

Figure 11: View spending insights sequence diagram

Use Case: Set Warranty Reminders

- **Goal:** The user wants to be reminded before a warranty expires.
- **Actors:** User
- **Description:** This case allows the User to set up customizable notifications for their stored warranties. The system will send reminders via the app or other configured channels as the warranty expiry date approaches.
- **Scenario (Setting a Reminder):**
    1. The User logs into the system.
    2. The User navigates to their list of stored warranties.
    3. The User selects a specific warranty.
    4. The system displays the warranty details, including the expiry date.
    5. The User selects the "Set Reminder" option.
    6. The system presents options for reminder frequency and timing (e.g., 1 week before, 1 month before).
    7. The User configures the desired reminder settings.
    8. The system confirms that the reminder has been set.
- **Exceptional Scenarios:**

- **ES-SWR-1: Invalid Expiry Date:** If the extracted or manually entered expiry date is invalid or missing, the system should not allow setting a reminder and prompt the user to correct the date.
- **ES-SWR-2: Notification Delivery Failure:** While the system sends reminders, external factors (e.g., user's device settings, network issues) might prevent the notification from being delivered. The system should ideally have a way to confirm notification delivery or provide an in-app notification center.

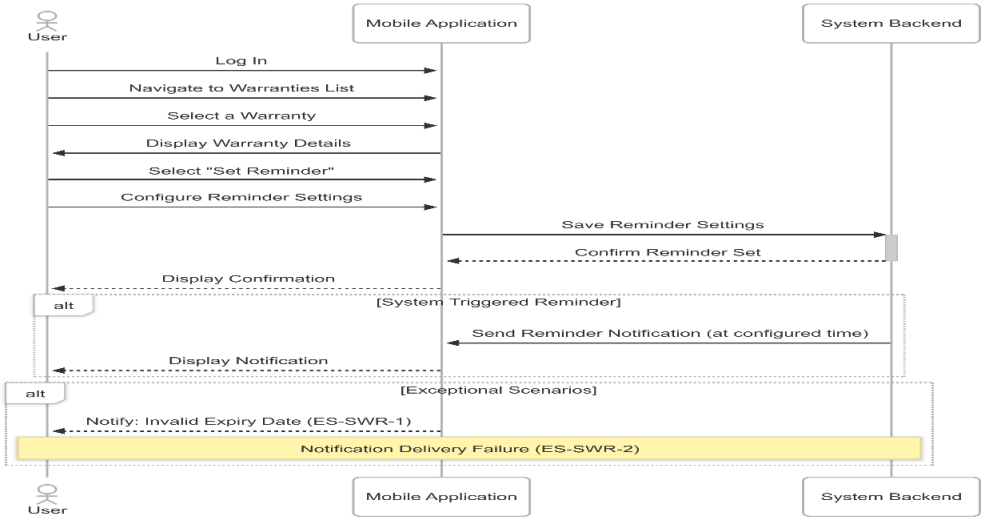

Figure 12: Set warranty reminder sequence diagram

Use Case: Initiate Warranty Claim

- **Goal:** The user wants to start the process of making a claim under a warranty.
- **Actors:** User
- **Description:** This use case provides a guided process for the User to initiate a warranty claim. It may involve gathering necessary information, accessing the warranty document, and potentially generating a summary for submission.
- **Scenario (Initiating a Claim):**
    1. The User logs into the system.
    2. The User navigates to their list of stored warranties.
    3. The User selects the warranty for the item they wish to claim against.
    4. The system displays the warranty details.
    5. The User selects the "Initiate Claim" option.
    6. The system may prompt the user for details about the issue.
    7. The system may generate a summary document containing key warranty information.
    8. The system may provide options for contacting the retailer or manufacturer.

- **Exceptional Scenarios:**
    - **ES-IWC-1: Expired Warranty:** If the warranty has already expired, the system should inform the user and prevent them from initiating a claim through this process.
    - **ES-IWC-2: Missing Information:** If essential information required for the claim (e.g., proof of purchase date) is missing from the stored warranty, the system should prompt the user to provide it.
    - **ES-IWC-3: Retailer Not Integrated:** If the retailer is not integrated with the system for claim processing, the system should provide the user with the necessary warranty information to pursue the claim through external channels.

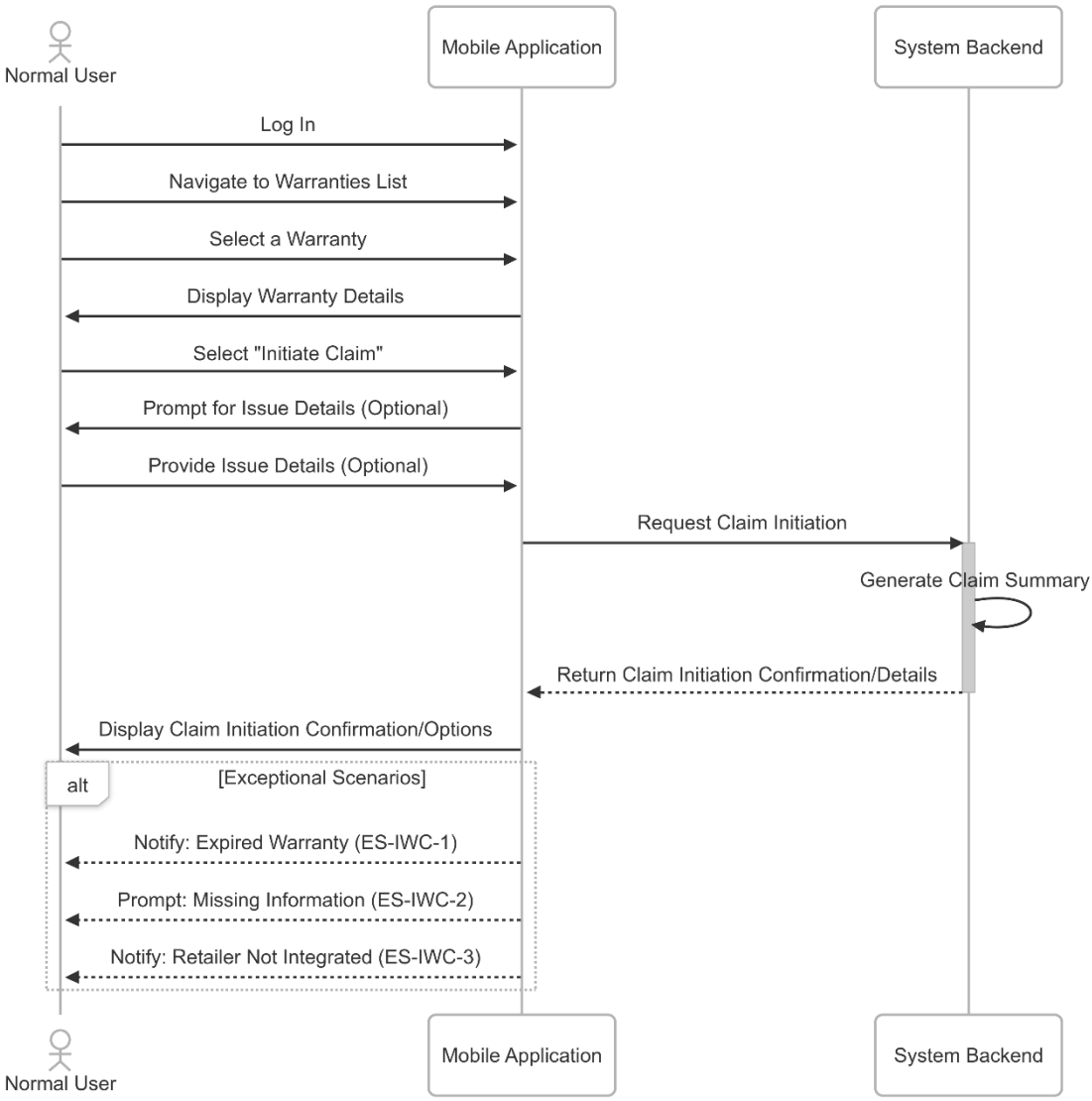

Figure 13: Warranty Claim sequence diagram

Use Case: Renew Warranty

- **Goal:** The user wants to extend the coverage period of a warranty.
- **Actors:** User
- **Description:** This use case allows the User to explore options for renewing an existing warranty through the system, if the retailer or manufacturer supports this functionality and integrates with the system.
- **Scenario (Exploring Renewal Options):**
    1. The User logs into the system.
    2. The User navigates to their list of stored warranties.
    3. The User selects a warranty that is eligible for renewal.
    4. The system displays the warranty details.
    5. The User selects the "Renew Warranty" option.
    6. The system checks if renewal options are available through integrated retailers/manufacturers.
    7. The system displays available renewal plans or directs the user to the relevant external process.
- **Exceptional Scenarios:**
    - **ES-RW-1: Warranty Not Renewable:** If the specific warranty is not eligible for renewal according to the terms or the retailer/manufacturer's policy, the system should inform the user.
    - **ES-RW-2: Retailer/Manufacturer Not Integrated for Renewal:** If the relevant party does not support renewal through the system, the system should inform the user and provide available contact information if known.

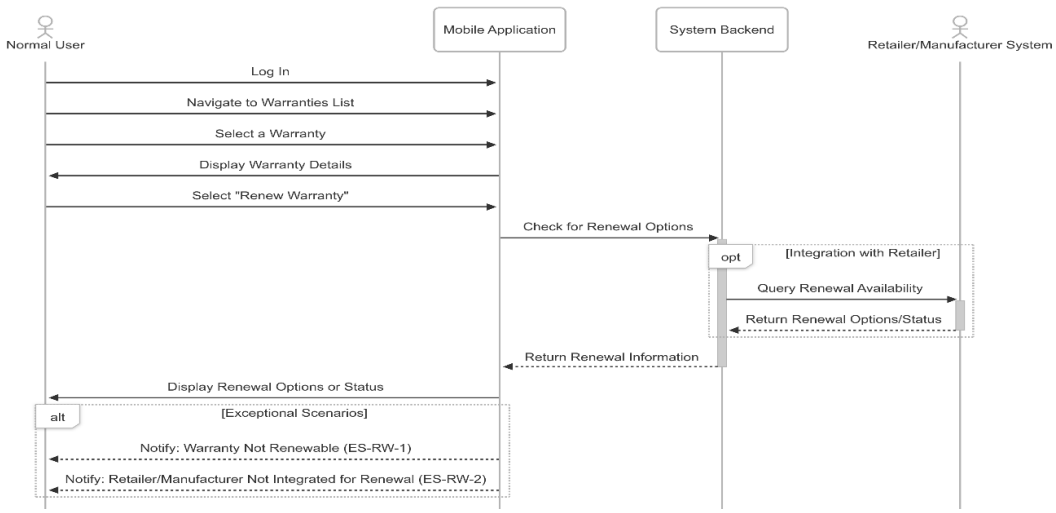

Figure 14: Renew warranty sequence diagram

Use Case: Transfer Warranty

- **Goal:** The user wants to transfer ownership of a warranty to another individual.
- **Actors:** User
- **Description:** This use case facilitates the transfer of an authenticated warranty from one User to another within the system. This is particularly relevant for secondhand sales. The process should ensure the validity and authenticity of the transfer.
- **Scenario (Transferring a Warranty):**
    1. The User (Seller) logs into the system.
    2. The User (Seller) navigates to their list of stored warranties.
    3. The User (Seller) selects the warranty to be transferred.
    4. The system displays the warranty details.
    5. The User (Seller) selects the "Transfer Warranty" option.
    6. The system prompts the User (Seller) to identify the recipient (e.g., via email or system username).
    7. The system initiates a transfer request to the recipient.
    8. The recipient (User) receives a notification and accepts the transfer.
    9. The system updates the warranty ownership in the database.
- **Exceptional Scenarios:**
    - **ES-TW-1: Warranty Not Transferable:** If the terms of the warranty explicitly state it is non-transferable, the system should prevent the transfer and inform the user.
    - **ES-TW-2: Recipient Not a System User:** If the intended recipient is not a registered user of the system, the system should inform the sender and potentially offer an invitation option.
    - **ES-TW-3: Recipient Rejects Transfer:** If the recipient declines the transfer request, the system should notify the sender.
    - **ES-TW-4: Transfer Authentication Failure:** If the system cannot verify the authenticity or eligibility of the warranty for transfer, the transfer should be blocked.

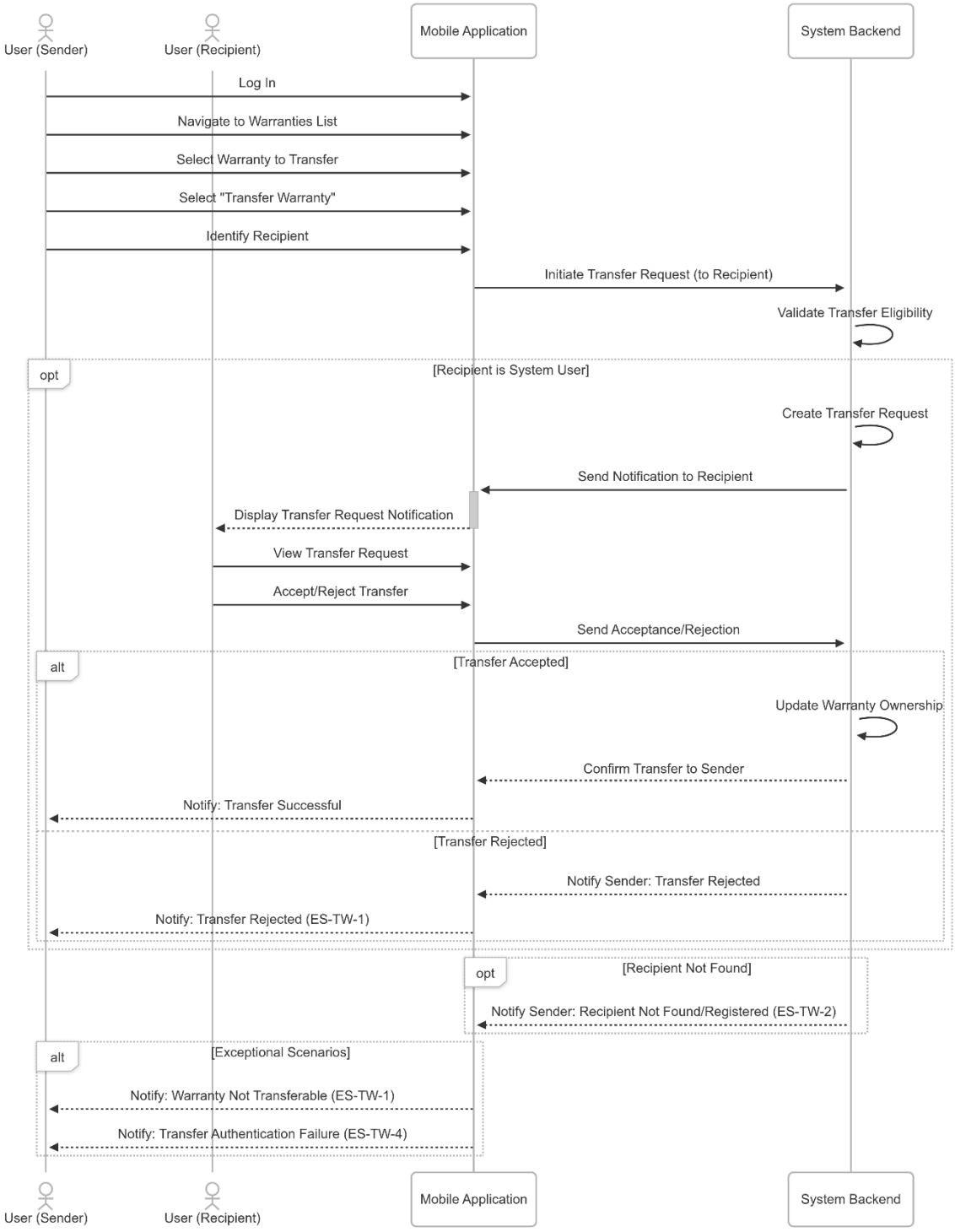

Figure 15: Transfer warranty sequence diagram

Use Case: Create P2P Agreement

- **Goal:** The user wants to create a simple agreement for an item exchanged with another individual.
- **Actors:** User
- **Description:** This use case allows Users to create basic peer-to-peer agreements within the system, potentially including simple terms similar to a warranty, for items exchanged outside of formal retail channels.
- **Scenario (Creating a P2P Agreement):**
    1. The User logs into the system.
    2. The User selects the "Create P2P Agreement" option.
    3. The system prompts the user to enter details about the item and the terms of the agreement (e.g., duration, covered issues).
    4. The system prompts the user to identify the other party involved.
    5. The system generates the P2P agreement document.
    6. The other party receives a notification to review and accept the agreement.
    7. Upon acceptance, the system stores the P2P agreement for both parties.
- **Exceptional Scenarios:**
    - **ES-P2PA-1: Other Party Rejects Agreement:** If the other party declines to accept the P2P agreement, the system should notify the creator.
    - **ES-P2PA-2: Invalid Agreement Terms:** The system may include basic validation to prevent obviously nonsensical or malicious terms, informing the user if the terms are invalid.

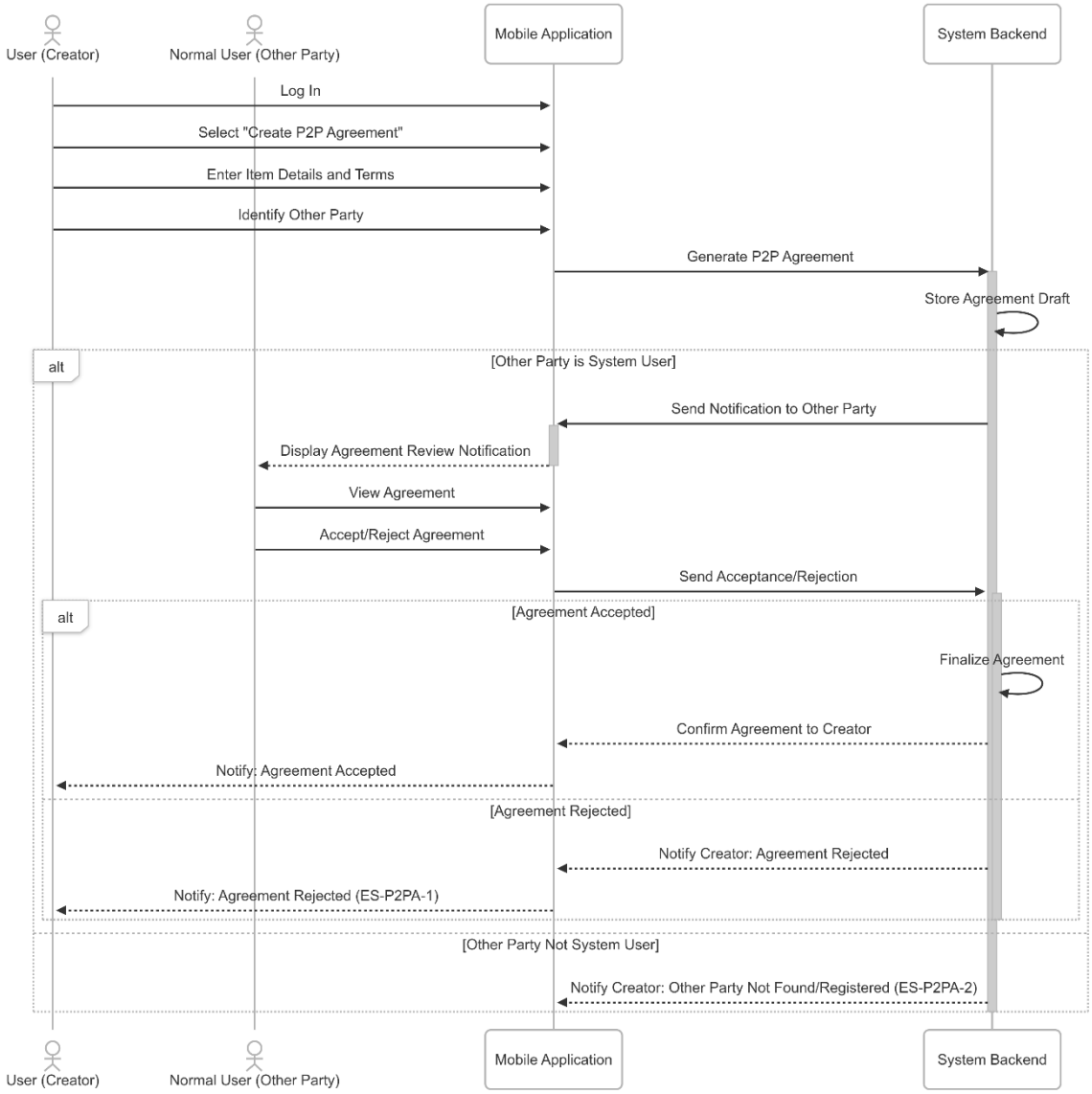

Figure 16: P2P agreement sequence diagram

Use Case: Issue Document

- **Goal:** The retailer wants to provide a digital receipt or warranty to a customer.
- **Actors:** Retailer
- **Description:** This use case allows authorized Retailers to generate and issue digital receipts and warranties directly to a customer's account within the system, ideally at the point of sale.
- **Scenario (Issuing a Digital Receipt):**
    1. The Retailer logs into their account on the system (or integrated POS).
    2. The Retailer completes a transaction.
    3. The Retailer selects the option to issue a digital receipt via the system.

4. The system prompts the Retailer to identify the customer (e.g., via email or system ID).
5. The system generates a digital receipt containing transaction details.
6. The system sends the digital receipt to the customer's User account.

- **Exceptional Scenarios:**
  - **ES-IDD-1: Customer Not Found/Registered:** If the customer cannot be identified or is not a registered user of the system, the system should inform the Retailer and potentially offer an option to invite the customer.
  - **ES-IDD-2: Integration Failure:** If there is an issue with the integration between the Retailer's POS/system and the Receipt and Warranty Management System, the digital issuance may fail. The system should provide an error message to the Retailer.
  - **ES-IDD-3: Data Entry Errors:** If the Retailer enters incorrect transaction details, the issued document will be inaccurate. The system may have validation rules but ultimately relies on accurate input.

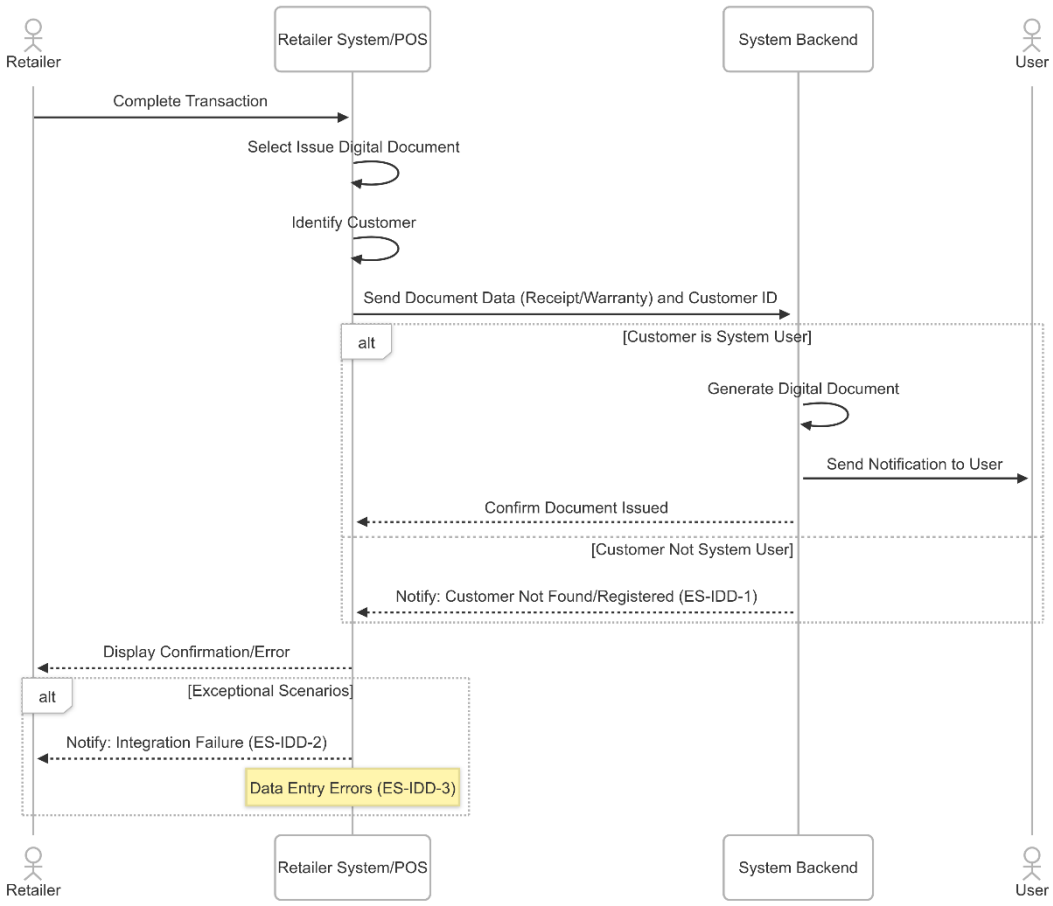

Figure 17: Issue document sequence diagram

Use Case: Process Warranty Claim

- **Goal:** The retailer wants to review and respond to a customer's warranty claim.
- **Actors:** Retailer
- **Description:** This use case provides the functionality for Retailers to receive, review, and manage warranty claims submitted by Users through the system. The Retailer can accept or reject the claim and communicate the decision.
- **Scenario (Accepting a Warranty Claim):**
    1. The Retailer logs into their account on the system.
    2. The Retailer navigates to the list of pending warranty claims.
    3. The Retailer selects a specific claim submitted by a User.
    4. The system displays the claim details, including the associated warranty and user-provided information.
    5. The Retailer reviews the claim and the warranty terms.
    6. The Retailer selects the "Accept Claim" option.
    7. The system updates the claim status to "Accepted".
    8. The system sends a notification to the User informing them that their claim has been accepted.
- **Exceptional Scenarios:**
    - **ES-PWC-1: Invalid Claim Information:** If the submitted claim is missing required information or appears fraudulent, the Retailer may reject it. The system should facilitate communication of the reason for rejection.
    - **ES-PWC-2: Associated Document Missing:** If the warranty document associated with the claim is somehow missing or inaccessible to the Retailer, the system should flag this issue.
    - **ES-PWC-3: Retailer System Issues:** If the Retailer's internal systems required to verify the claim are down or inaccessible, the processing of the claim may be delayed.

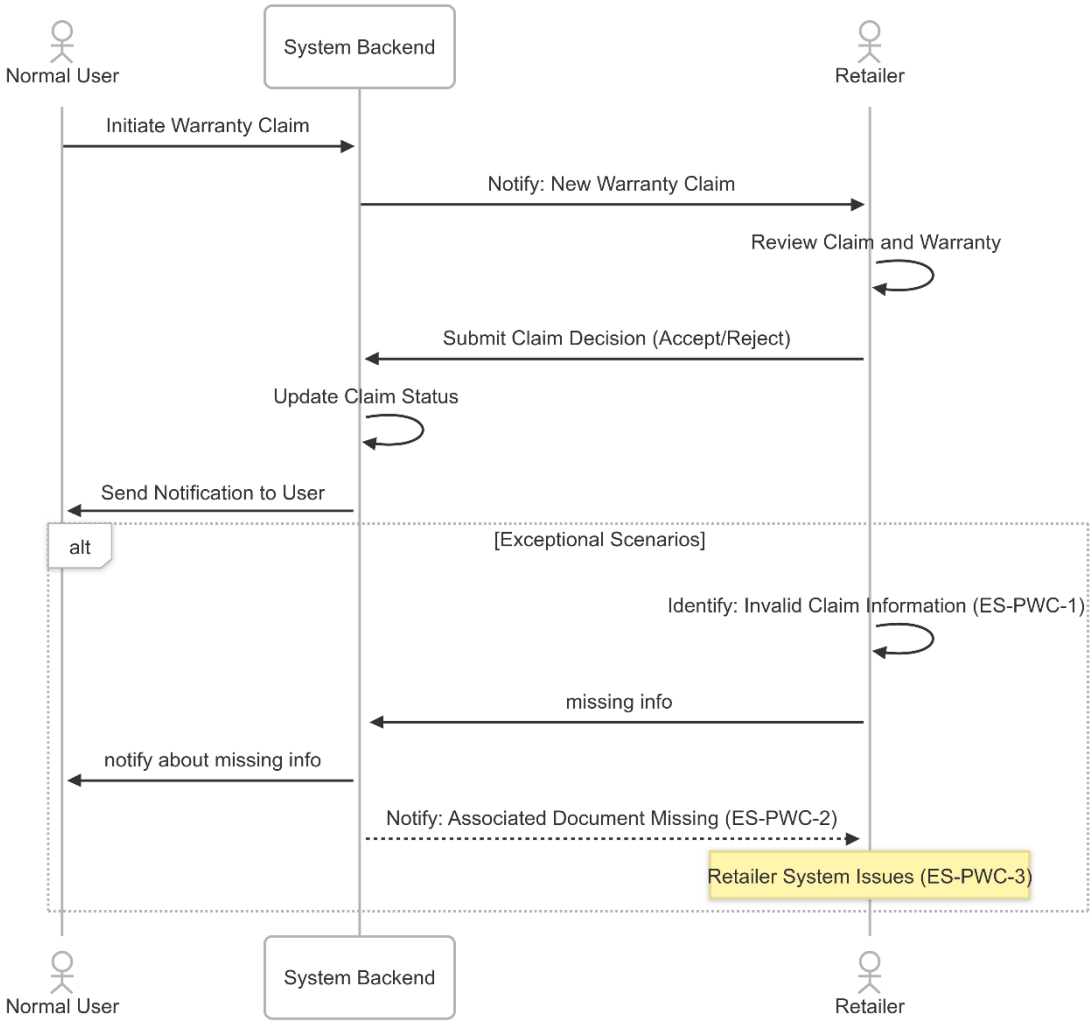

Figure 18: Warranty claim sequence diagram

Use Case: Manage Issued Documents

- **Goal:** The retailer wants to view or manage the digital documents they have issued.
- **Actors:** Retailer
- **Description:** This use case allows authorized Retailers to access a record of the digital receipts and warranties they have issued through the system. They may be able to view details or potentially perform limited management actions (e.g., voiding a receipt under specific conditions).
- **Scenario (Viewing Issued Receipts):**
    1. The Retailer logs into their account on the system.
    2. The Retailer navigates to the "Issued Documents" section.
    3. The system displays a list of digital receipts and warranties issued by the retailer.
    4. The Retailer can filter or search the list (e.g., by date, customer).

5. The Retailer can select a document to view its details.

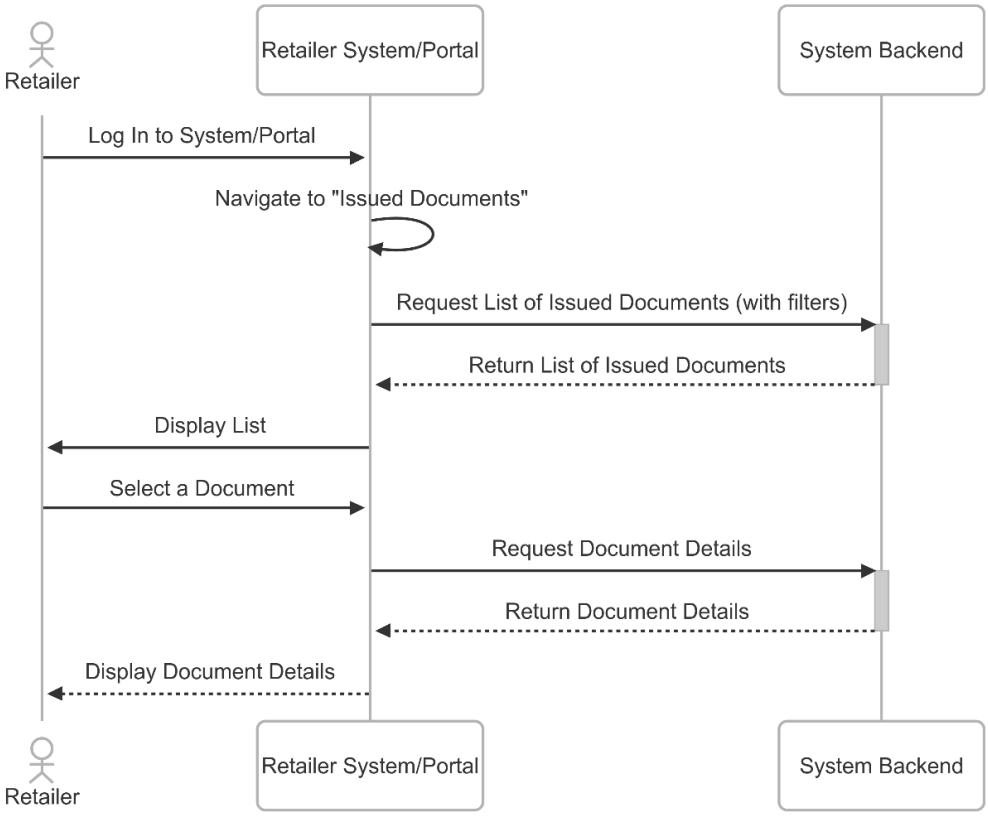

Figure 19: Manage issued document sequence diagram

4.4.3 UML System Class Diagram

Using a Unified Modeling Language (UML) Class Diagram, this part shows the fundamental structure of the system. The diagram shows the service components modeled as Classes responsible for managing the business logic and persistence of the main data entities modeled as Classes representing data structures. Class relationships show how various system components interact and rely on one another.

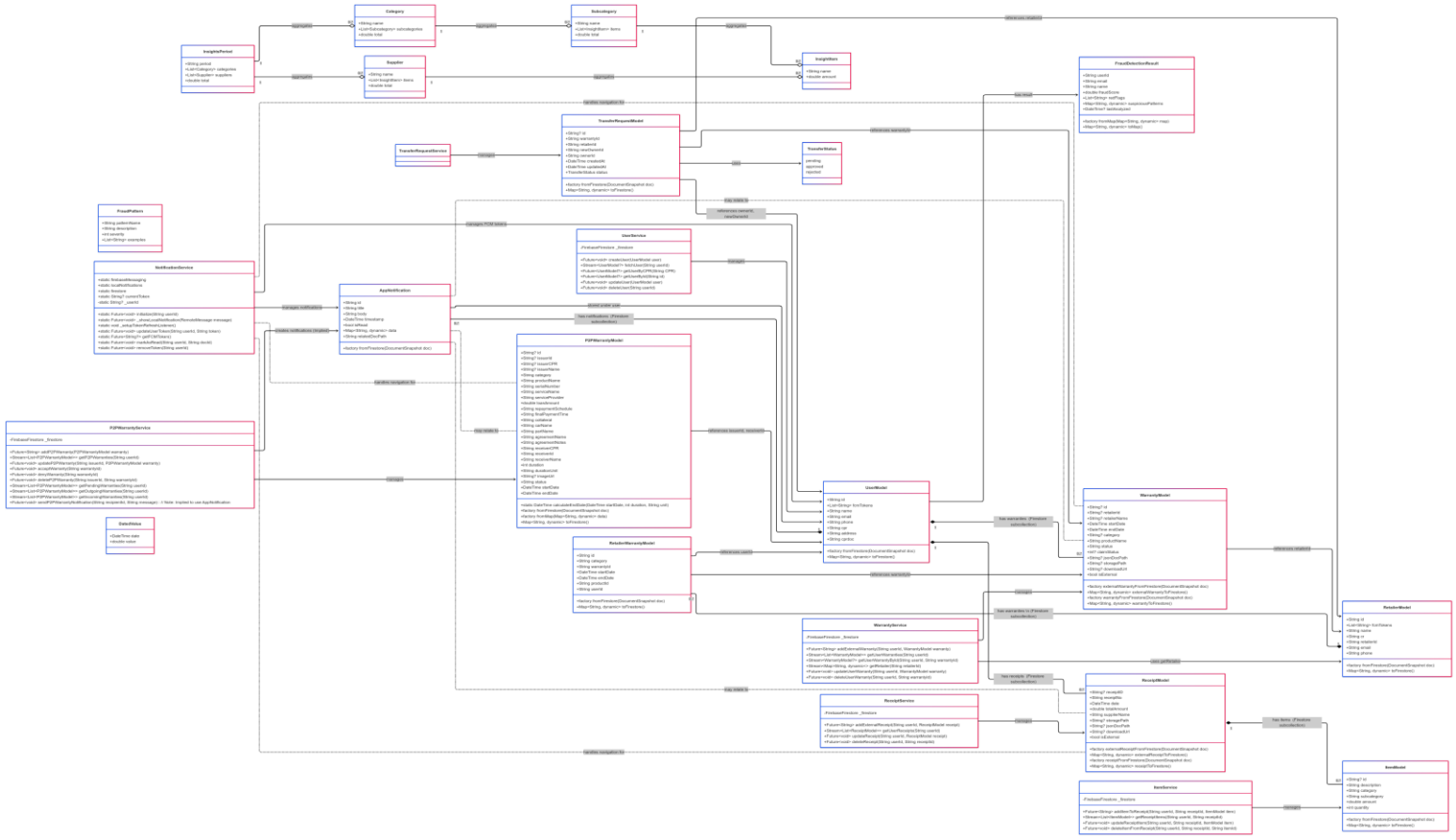

Figure 20: System Class diagram

.

To have a clear view if the diagram please refer to Appendix B.

4.4.3.1 Data Models (Classes)

The class diagram specifies various classes denoting the main data entities inside the system. Usually, these classes match the organization of data objects utilized for processing and communication or documents stored in the Cloud Firestore database.

UserModel: Represents a registered user or customer of the mobile application.

- Attributes: Consists of standard identification and contact information (id, name, email, phone, address), unique identifiers (cpr - Civil Personal Registration), a reference to a CPR document file (cprdoc), and a list of device tokens (fcmTokens) for push notifications.
- Methods: Contains factory constructors (fromFirestore) to generate UserModel instances from Firestore document snapshots and instance methods (toFirestore) to transform UserModel instances into a format appropriate for saving to Firestore.

RetailerModel: Represents a registered retailer interacting with the system.

- Attributes: Holds identification (retailerId), business information (name, cr - Commercial Registration), contact details (email, phone), and fcmTokens for notifications.
- Methods: For managing data transformation with Firestore, includes factory constructors (fromFirestore) and instance methods (toFirestore).

WarrantyModel: From a user's viewpoint, it denotes a product warranty record.

Key details are a unique id, references to the originating retailer (retailerId, retailerName), the warranty period (startDate, endDate), product information (productName, category), status tracking (status, claimStatus), and possible references to related files or documents (jsonDocPath, storagePath, downloadUrl) this will be filled if the user adds the warranty using the OCR service. The isExternal flag shows whether the warranty came from a retailer system.

- Methods: This class comprises several factory and instance methods, including externalWarrantyFromFirestore, externalWarrantyToFirestore, warrantyFromFirestore, and warrantyToFirestore. Depending on the warranty's source whether from a retailer system or user-uploaded via the OCR service these techniques are applied to manage data conversion.

RetailerWarrantyModel: Represents a warranty record mostly from the retailer's viewpoint.

- Attributes: Comprises its own id, a user reference (userId), core warranty information (startDate, endDate, category, productId), and a reference to the associated WarrantyModel using warrantyId.

- Methods: Data conversion using standard fromFirestore and toFirestore techniques. This model probably reflects a denormalized or simplified perspective of the warranty data kept particularly under the retailer's path in Firestore, maybe with retailer-specific fields or excluding user-centric ones. The connection indicates that it is connected to UserModel and WarrantyModel.

ReceiptModel: Usually submitted by a store, stands for a customer's purchase receipt.

- Attributes: Consists of a unique receiptID, the retailer's internal receiptNo, transaction information (date, totalAmount, supplierName), and possible references to related files or documents (storagePath, jsonDocPath, downloadUrl); this will be used only if the receipt is uploaded using the OCR service. The isExternal flag shows whether an external retailer system submitted the receipt via the API.
- Methods: For the various data representations depending on the source, includes several factory and instance methods (externalReceiptFromFirestore, externalReceiptToFirestore, receiptFromFirestore, receiptToFirestore), like WarrantyModel.

ItemModel: Represents one line item in a ReceiptModel.

- Attributes: Consists of its own distinct id, item description (description), automated classification results (category, subcategory), and transaction details for that item line (amount, quantity).
- Methods: Data conversion using standard fromFirestore and toFirestore techniques.

P2PWarrantyModel: Represents a warranty or agreement formed between two users (Peer-to-Peer).

- Attributes: Includes extensive fields capturing various details pertinent to different types of peer-to-peer agreements (loan amount, repayment schedule, collateral, car/part details, agreement notes), identifiers for the involved parties (issuerId, issuerCPR, issuerName, receiverId, receiverCPR, receiverName), warranty period details (startDate, duration, durationUnit, calculated endDate), product/service details (category, productName, serialNumber, serviceName, serviceProvider), optional image URL (imageUrl), and the P2P warranty's current status.
- Methods: Comprises factory constructors (fromFirestore, fromMap) for generating instances from Firestore or general map data, an instance method (toFirestore) for saving to Firestore, and a static utility method (calculateEndDate) for determining the end date depending on duration.

TransferRequestModel: Represents a request to transfer ownership of a warranty.

Using the TransferStatus enumeration, attributes include its own id, references to the entities involved (warrantyId for the warranty being transferred, retailerId, ownerId for the current owner, newOwnerId for the intended recipient), timestamps (createdAt, updatedAt), and the current status of the request.

- Methods: Data conversion using standard fromFirestore and toFirestore techniques.

AppNotification: Represents a notification object kept in the database meant for certain user.

- Attributes: Consists of a unique id, notification content (title, body), timestamp (timestamp), read status (isRead), a map for arbitrary data (data), and a path to the related document (relatedDocPath) the notification pertains to.
- methods comprises a fromFirestore factory constructor.

FraudDetectionResult: Represents the result of a user fraud investigation.

Links to the user (userId, email, name), a calculated fraudScore, a list of identified warning signs (redFlags), information on suspicious patterns (suspiciousPatterns), and the time of the last analysis (lastAnalyzed).

- Methods: Data conversion comprises fromMap and toMap methods.

FraudDetectionPattern: Represents a particular pattern or rule applied in fraud detection.

- Attributes: Specifies the pattern (patternName, description), its severity level (severity), and illustrative examples.

InsightItem: Represents a total combined for a particular item description inside the insights data structure.

- InsightItem: Indicates a total for a particular item description inside the insights data structure.

DatedValue: A number linked to a certain date. Applied to time-series data in insights.

- Attributes: The date and the corresponding value.

Subcategory: Represents an aggregation level within the insights data, subcategory groups items under a particular subcategory.

- Subcategory: Is an aggregation level in the insights data that groups items under a particular subcategory.

Category: An aggregation level in the insights data, it groups spending under a general category. The name of the category, a list of aggregated Subcategory totals and items inside it, and the total amount spent in this category.

Supplier: Indicates a level of aggregation in the insights data that groups expenditure by the supplier (retailer).

- Attributes: The name of the supplier, a list of aggregated InsightItems from this supplier, and the total amount spent at this supplier.

InsightsPeriod: Represents a particular time period e.g., a day, week, month for which spending data is compiled.

- Attributes: The period identifier (e.g., date string), lists of aggregated Category and Supplier data, and the total spending (total) for that time frame. AddReceiptv2 manages the user under which the aggregated data is stored using this framework.

TransferStatus: is an enumeration-like definition describing the possible states for a TransferRequestModel, such pending, approved, or rejected.

Service Classes 4.4.3.2

The diagram features classes denoting the service layer, which encapsulates the business logic and interacts with the data persistence layer (Firestore for this situation). These services oversee the lifecycle of the data models.

UserService: Manages UserModel data.

- Methods: Offers functions to build (createUser), retrieve (WorkspaceUser as a stream, getUserByCPR, getUserById), modify (updateUser), and remove (deleteUser) user profiles. It works with Firestore.

WarrantyService: From the user's point of view and interaction with stores, it handles WarrantyModel data.

Includes operations to add external warranties (addExternalWarranty), retrieve user-specific warranties (getUserWarranties as a stream, getUserWarrantyById as a stream), retrieve retailer information (getRetailer as a stream), update (updateUserWarranty), and delete (deleteUserWarranty) user warranties. It works with Firestore.

ReceiptService: Manages ReceiptModel data for users.

- Methods: Offers operations to add external receipts (addExternalReceipt), get user-specific receipts (getUserReceipts as a stream), update (updateReceipt), and delete (deleteReceipt) user receipts. It works alongside Firestore.

ItemService: Manages ItemModel data, particularly in relation to a receipt.

- Methods: Consists of operations to add (addItemToReceipt), get (getReceiptItems as a stream), update (updateReceiptItem), and delete (deleteItemFromReceipt) items linked to a particular user and receipt. It works with Firestore.

P2PWarrantyService: Focusing on peer-to-peer agreements, handles P2PWarrantyModel data.

- Methods: Offers a variety of operations for generating (addP2PWarranty), fetching (general getP2PWarranties and filtered streams for getPendingWarranties, getOutgoingWarranties, getIncomingWarranties), modifying (updateP2PWarranty), status changes (acceptWarranty, denyWarranty), and deleting (deleteP2PWarranty) P2P warranties. It also provides a way to send P2P warranty-related notifications (sendP2PWarrantyNotification). It works with Firestore.

NotificationService: In charge of handling and controlling application notifications.

- Methods: Consists of static methods for initialization (initialize), device token management (updateUserToken, getFCMToken, removeToken), local notification display handling (_showLocalNotification), token refresh listener setup (_setupTokenRefreshListener), and notification marking as read (markAsRead). It works with Firestore (firestore), local notification systems (localNotifications), and Firebase Messaging (firebaseMessaging). Stored with the UserModel, it controls FCM tokens.

TransferRequestService: Responsible for overseeing TransferRequestModel data.

- o Methods: Offers operations to generate (createTransferRequest), fetch user-specific requests (getUserTransferRequests as a stream), modify request status (updateTransferRequestStatus), and remove requests (deleteTransferRequest). It works with Firestore.

4.4.3.3 Relationship

The lines and arrows linking the classes in the diagram show the relationships inside the system, therefore showing how various entities and services are linked and interact. These relationships can be classified into several types: Associations, Composition, Dependencies, and Aggregation.

Associations: Indicate that one class is linked to or knows about another, usually done by keeping references (like document IDs) or by using method calls.

- **Directed Association (Service to Model)**: A directed association shows that a Service class is in charge of executing operations creating, reading, updating, deleting on instances of a particular Model class.
    - When handling warranties, the WarrantyService must retrieve data from the RetailerModel for example, the retailer's name.
    - When handling warranties, the WarrantyService must retrieve data from the RetailerModel for example, the retailer's name.

- **References between Models**: Relationships such as TransferRequestModel linking to WarrantyModel, UserModel, and RetailerModel, or WarrantyModel linking to RetailerModel, suggest that one data model contains an identifier (such as an ID) pointing to an instance of another data model. Likewise, P2PWarrantyModel refers to the two UserModels engaged (issuer and receiver), while RetailerWarrantyModel refers to UserModel and WarrantyModel.

Composition: Represents a strong "part-of" or "owns" relationship, where the "part" entity conceptually belongs only to the "whole" entity and usually cannot exist independently if the "whole" is gone. In this approach, these connections mostly reflect data being organized inside Firestore subcollections.

- A User can have several Receipts; each Receipt belongs solely to one User. ReceiptModel documents are kept in a receipts subcollection directly beneath the relevant UserModel document (/users/{userId}/receipts/{receiptId}), which implements this.
- A User can have several Warranties; each Warranty in this particular perspective belongs only to one User. WarrantyModel documents are kept in a warranties subcollection under the relevant UserModel document (/users/{userId}/warranties/{warrantyId}), which implements this.
- A User can hold several P2P Warranties, either as issuer or receiver. The diagram implies composition, mapping to a subcollection structure, or possibly via a shared collection with references by keeping P2PWarrantyModel documents in a subcollection under the User's document (/users/{userId}/p2p_warranties/{p2pWarrantyId}).
- A Retailer can have several RetailerWarrantyModel representations (reflecting their perspective of issued warranties). RetailerWarrantyModel papers are kept in a warranties subcollection under the relevant RetailerModel document (/retailers/{retailerId}/warranties/{warrantyId}), thus implementing this.
- A Receipt has several Items; each Item belongs only to one Receipt. Storing ItemModel documents in an items subcollection straight under the relevant ReceiptModel document e.g., /users/{userId}/receipts/{receiptId}/items/{itemId} allows this to be implemented.
- A User can have several application Notifications, and every notification is for a particular User. AppNotification documents are kept in a notifications subcollection under the relevant UserModel document (/users/{userId}/notifications/{notificationId}), which implements this.

- A User has a defined framework for totalled spending data for several time periods. The diagram suggests a composition to InsightsPeriod, suggesting either a related subcollection structure (like /users/{userId}/insights) or a single main insights document where InsightsPeriod data is stored. The relationship suggests that the user owns this insights structure.

Aggregation: A "has-a" relationship whereby the "part" entities can exist apart from the "whole" entity. The diagram mostly uses this to indicate the structure of the combined insights data inside the InsightsPeriod entity.

- An InsightsPeriod "has" or aggregates data pertaining to several Categories and Suppliers. Rather than keeping Categories or Suppliers as distinct top-level documents or subcollections, this is carried out by organizing the InsightsPeriod document to include nested fields such as maps or arrays holding the combined Category and Supplier data straight inside that document.
- A Category aggregation "has" or aggregates Subcategory data and possibly InsightItems straight. A Supplier aggregation or Subcategory "has" InsightItem data. Nested maps or arrays under the parent Category or Supplier framework inside the InsightsPeriod document hold this data.

Dependencies: Indicate one class depends on another, usually via method calls or by means of instances passed as parameters, without maintaining a direct, persistent reference.

- The NotificationService relies on entities like WarrantyModel, ReceiptModel, and P2PWarrantyModel to manage navigation when a user interacts with a notification that pertains to one of these items. It doesn't control these models directly but requires their structure or data to carry out its function.
- The diagram indicates that the TransferRequestModel's status attribute uses the TransferStatus enumeration. Ensuring data consistency for request states, this implies the status field on a Transfer Request document can only hold one of the predefined values specified in the TransferStatus enumeration (pending, approved, or rejected).

4.4.3.3 Relationships

The lines and arrows linking the classes in the diagram show the relationships inside the system, therefore showing how various entities and services are linked and interacted. These relationships can be classified into several types: Associations, Composition, Dependencies, and Aggregation.

- **Associations**: Indicate that one class is linked to or knows about another, usually done by keeping references (like document IDs) or by using method calls.
    - A directed association shows that a Service class is in charge of executing operations creating, reading, updating, and deleting instances of a particular Model class.
        - When handling warranties, the WarrantyService must retrieve data from the RetailerModel for example, the retailer's name.
        - When handling warranties, the WarrantyService must retrieve data from the RetailerModel for example, the retailer's name.
    - References between Models: Relationships such as TransferRequestModel linking to WarrantyModel, UserModel, and RetailerModel, or WarrantyModel linking to RetailerModel, suggest that one data model contains an identifier (such as an ID) pointing to an instance of another data model. Likewise, P2PWarrantyModel refers to the two UserModels engaged (issuer and receiver), while RetailerWarrantyModel refers to UserModel and WarrantyModel.
- **Composition**: Represents a strong "part-of" or "owns" relationship, where the "part" entity conceptually belongs only to the "whole" entity and usually cannot exist independently if the "whole" is gone. In this approach, these connections mostly reflect data being organized inside Firestore subcollections.
    - A User can have several Receipts; each Receipt belongs solely to one User. ReceiptModel documents are kept in a receipts subcollection directly beneath the relevant UserModel document (/users/{userId}/receipts/{receiptId}), so carrying this out.
    - User and Warranties: In this particular perspective, a User may hold several Warranties, and every Warranty belongs solely to one User. Storing WarrantyModel documents in a warranties subcollection under the relevant UserModel document (/users/{userId}/warranties/{warrantyId}) implements this.

- A User can hold several P2P Warranties, either as issuer or receiver. The diagram implies composition, mapping to a subcollection structure, or possibly via a shared collection with references by keeping P2PWarrantyModel documents in a subcollection under the User's document (/users/{userId}/p2p_warranties/{p2pWarrantyId}).
- A Retailer can have several RetailerWarrantyModel representations (reflecting their perspective of issued warranties). RetailerWarrantyModel papers are kept in a warranties subcollection under the relevant RetailerModel document (/retailers/{retailerId}/warranties/{warrantyId}), thus implementing this.
- A Receipt has several Items; each Item belongs only to one Receipt. Storing ItemModel documents in an items subcollection straight under the relevant ReceiptModel document e.g., /users/{userId}/receipts/{receiptId}/items/{itemId} allows this to be implemented.
- A User can have several application Notifications, and every notification is for a particular User. AppNotification documents are kept in a notifications subcollection under the relevant UserModel document (/users/{userId}/notifications/{notificationId}), which implements this.
- A User has a defined framework for totalled spending data for several time periods. The diagram suggests a composition to InsightsPeriod, suggesting either a related subcollection structure (like /users/{userId}/insights) or a single main insights document where InsightsPeriod data is stored. The relationship suggests that the user owns this insights structure.

- **Aggregation**: A "has-a" relationship whereby the "part" entities can exist apart from the "whole" entity. The diagram mostly uses this to indicate the structure of the combined insights data inside the InsightsPeriod entity.
  - Insights Period and Categories/Suppliers: An InsightsPeriod "has" or compiles data about several Categories and Suppliers. Rather than keeping Categories or Suppliers as separate top-level documents or subcollections, this is carried out by organizing the InsightsPeriod document to include nested fields such as maps or arrays holding the combined Category and Supplier data straight inside that document.
  - A Category aggregation "has" or aggregates Subcategory data and possibly InsightItems straight. A Supplier aggregation or Subcategory "has" InsightItem

data. Nested maps or arrays under the parent Category or Supplier framework inside the InsightsPeriod document hold this data.

- **Dependencies**: Indicate one class depends on another, usually via method calls or by means of instances passed as parameters, without maintaining a direct, persistent reference.
    - The NotificationService relies on entities like WarrantyModel, ReceiptModel, and P2PWarrantyModel to manage navigation when a user interacts with a notification that pertains to one of these items. It doesn't control these models directly but requires their structure or data to carry out its function.
    - The diagram indicates that the TransferRequestModel's status attribute uses the TransferStatus enumeration. Ensuring data consistency for request states, this implies the status field on a Transfer Request document can only hold one of the predefined values specified in the TransferStatus enumeration (pending, approved, or rejected).

4.4.3.4 Interaction between Services and Models

Active elements in the business logic layer of the system are the Service classes (UserService, WarrantyService, ReceiptService, ItemService, P2PWarrantyService, NotificationService, TransferRequestService). They are in charge of working with the data persistence layer (Firestore) to carry out operations (including creating, reading, updating, and deleting) on the passive data structures represented by the Model classes (UserModel, WarrantyModel, ReceiptModel, ItemModel). Services usually run database commands using an instance of the database client, shown by the private _firestore property in several Service classes. Using the factory constructors such as fromFirestore, fromMap provided by the Model classes, they transform data obtained from the database into functional application objects. On the other hand, they transform application data into a format appropriate for saving to the database using the instance methods such as toFirestore, toMap on the Model objects. The directed links from Services to the Models marked "manages" clearly show this duty and flow of control; the Service class knows about and runs on the Model class instances.

Chapter 5
System Design

This chapter describes the process of turning the system requirements into a tangible design and highlights the main implementation choices taken during the working prototype's development. It justifies the tools, technologies, and algorithms used; it explains the selected software architecture; it discusses the choice and integration of several parts.

5.1 Why Microservices Architecture?

Microservices Architecture builds the system. In line with the non-functional criteria defined in Section 4.2.2, this architectural decision was made to guarantee that the system stays scalable, flexible, resilient, and maintainable. Decomposing the system into smaller, autonomous services produces several advantages:

The independent development and deployment of microservices allow teams to work concurrently on different services and deploy updates without disrupting the entire application, thereby accelerating the development cycle.

- Scalability: Individual microservices can be scaled up or down based on the specific demand they experience, optimizing resource utilization and performance under varying loads.
- Resilience: The system's design ensures that the failure of an individual microservice is contained, thereby preventing a ripple effect that could compromise the functionality of the entire system.
- Diversity in Technology: Although the primary technologies are standardized, microservices offer the flexibility to employ different programming languages or technologies for specific tasks, ensuring an optimal fit and enhanced functionality for each service.
- Enhanced Maintainability: Small, modular codebases are simpler to comprehend, debug, and manage compared to large, monolithic applications.

5.2 System Architecture and Component Integration

With the mobile app acting as the client and the backend services running as the server, the system's architecture follows a client-server model. Using Microservices, especially via Firebase services, the server-side architecture is built; Firebase Functions act as a server-less execution environment. The primary components and their integration are as follows:

- Developed with Flutter, a cross-platform framework selected for its ability to produce natively compiled apps for both iOS and Android from a single codebase, the mobile application is the client. This choice guarantees a quick development cycle and consistent, high-quality user interface. The app talks to the backend via APIs, shows data, and runs user interactions.
- Embedded inside both the mobile app and backend services, Firebase Authentication guarantees safe user registration, login, and session management. This element was selected to satisfy the security non-functional requirement (NFR-SEC-2) since it supports several authentication techniques, scalability, and simple implementation.
- User profiles, document metadata, warranty information, and retailer data are all stored using Cloud Firestore as the main NoSQL cloud database. Firestore capacity for real-time data synchronization, scalability, and flexibility to fit changing data structures drove this choice, therefore satisfying the data storage needs of the system. Data management is handled by microservices, especially Firebase Functions interfacing directly with Firestore.
- Digital files like receipts and warranties in many formats including photos and PDFs are stored using Cloud Storage for Firebase. Fulfilling the system's need for consistent document storage, this service provides a scalable and safe object storage solution that effortlessly integrates with other Firebase services.
- Digital files like receipts and warranties in many formats including images and PDFs are stored using Cloud Storage for Firebase. This service satisfies the system's need for reliable document storage by means of a scalable and safe object storage solution that is smoothly integrated with other Firebase services.
- The backend logic is hosted by Firebase Functions (Microservices), which run separate microservices activated by different events. These events might be internal Firebase events like the uploading of a new file to Cloud Storage, which triggers a processing function, or HTTP requests from the mobile app, such searches and insights. Its serverless benefits include:
    - Reduced Operational Overhead: This benefit lessens the need for manual server management, including duties like patching and scaling.
    - This function guarantees best performance under high load conditions by dynamically changing computing resources depending on demand (NFR-PERF-1, NFR-PERF-2).

    - Cost Efficiency: The billing model is based on the compute time used, so for tasks with fluctuating demand, it is a reasonable choice.
    - Faster deployment times come from the simplified backend logic deployment method.
    - Scheduled triggering/creation of crone jobs.
- Firebase Functions and Firebase Cloud Messaging (FCM) work together to enable push notification delivery to the mobile app. Timely reminders about warranty expirations (FR-WM-5) and other important notifications depend on this integration, therefore fulfilling a major user need noted in our survey.
- Firebase Functions connects Google Cloud Document AI to enable the automated document processing. This service extracts data from scanned receipts and warranties using sophisticated artificial intelligence and optical character recognition (OCR) technologies. Meeting certain system criteria connected to receipt and warranty management depends on this feature.
- Mostly via APIs and event triggers, the microservices interact among themselves and with Firebase services. Built on scalable cloud services, this distributed architecture offers a strong basis for the system.

5.3 Data Design (High-Level Schema)

The system will employ Cloud Firestore as its principal database, which is a NoSQL, document-oriented solution. The data will be systematically arranged into collections. The following is a high-level overview of the primary collections:

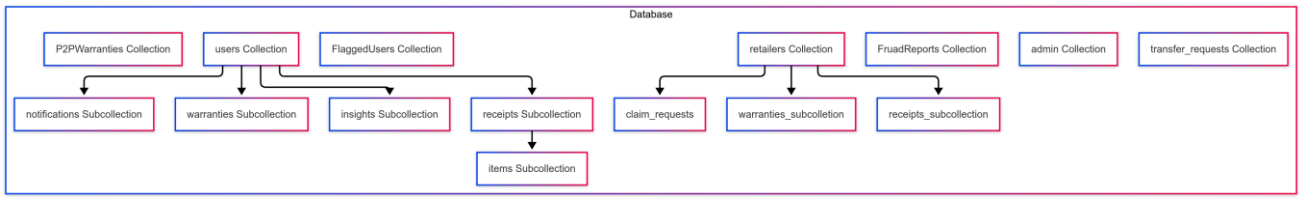

Figure 21: High level database schema

- users **Collection:**
    - Each document represents a user.
    - Sub Collections: warranties, receipts, insights, notifications.
    - Fields: userId (document ID), email, name, creationDate, etc.
- receipts **Collection:**
    - Each document represents a receipt.
    - Fields: receiptId (document ID), userId (reference to user), vendorName, purchaseDate, totalAmount, extractedItems (subcollection or array), documentUrl (reference to file in Cloud Storage), category, uploadDate, etc.
- warranties **Collection:**
    - Each document represents a warranty.
    - Fields: warrantyId (document ID), userId (reference to user), productName, purchaseDate, expiryDate, vendorName, termsAndConditions, documentUrl (reference to file in Cloud Storage), reminderSettings (subcollection or map), uploadDate, etc.
- retailers **Collection:**
    - Each document represents a registered retailer.
    - Sub Collections: claim_requests, warranties, receipts.
    - Fields: retailerId (document ID), name, contactInfo, etc.
- issued_documents **Collection:**
    - Documents representing receipts/warranties issued by retailers.
    - Fields: issuedDocId (document ID), retailerId (reference to retailer), userId (reference to user), documentType (receipt/warranty), documentData (relevant fields), issueDate, documentUrl, etc.

- claims **Collection:**
    - Documents representing warranty claims initiated by users.
    - Fields: claimId (document ID), userId (reference to user), warrantyId (reference to warranty), issueDescription, claimDate, status (e.g., pending, accepted, rejected), retailerId (reference to retailer), resolutionDetails, etc.
- p2p_agreements **Collection:**
    - Documents representing peer-to-peer agreements.
    - Fields: agreementId (document ID), creatorId (reference to user), otherPartyId (reference to user), itemDescription, terms, creationDate, acceptanceDate, status (e.g., pending, accepted), etc.

5.4 User Interface Design Approach

User interface design will take simplicity and clarity on smartphone screens. The main ideas are: A clean and simple design approach guarantees that only necessary information is highlighted, therefore lowering clutter and improving the user experience. Ensuring smooth access to important features like document capture, viewing lists, searches, and retrieval of insights and reminders. Emphasizing a clean and simple design, the user interface will give priority to necessary information and reduce clutter to improve user experience. Users will be clearly informed of visual feedback systems including those for upload progress, extraction results, and reminder settings. Ensuring all screens and interactions have a consistent and coherent look. Designing with accessibility rules in mind guarantees the app is usable by a larger audience. Building expressive user interfaces using Flutter's strong features will help to create an aesthetically pleasing and responsive app.

5.5 Software Algorithms

The primary software algorithm of the system involves the AI-driven extraction of data from scanned or uploaded documents. This process encompasses:

- Optical Character Recognition (OCR) use to convert images of text into machine-readable text within Document AI used in our implementation.
- Natural Language Processing (NLP) and Machine Learning (ML) will be utilized to analyze the structure and contents of receipts and warranties, allowing for the identification of key fields such as vendor, date, amount, items, and expiry dates, as well as categorizing expenditures. Google's Document AI service, specifically designed for such tasks, will be integrated into the backend processing of the system through Firebase Functions.

5.6 Architectural Drivers

The architectural design of the system has been profoundly influenced by several key drivers, which predominantly stem from the established non-functional requirements. These drivers directly led to the adoption of a Microservices Architecture, implemented on a serverless platform (Firebase Functions):

- **Scalability:** A scalable solution was required given the growing user base's rising demand and the expected increase in document volume. Adopting microservices architecture guarantees optimal resource allocation and performance optimization by allowing separate services, such document processing and search, to grow independently. The serverless architecture provided by Firebase Functions also enables automatic scaling depending on usage changes, therefore removing the need for manual server administration and improving operational efficiency.
- **Performance:** A very efficient backend processing system is needed to attain fast document retrieval and best search performance. Although frontend architecture and database structuring are important considerations, the ability of microservices to handle several requests at once greatly improves system responsiveness. Moreover, serverless functions' improved execution environment guarantees smooth scalability and effective resource use, therefore helping to achieve demanding performance goals.
- **Security:** Protecting private user data is top priority. Firebase provides built-in security features such data encryption and authentication systems to guarantee the integrity and confidentiality of user data. Adopting a microservices architecture allows security issues to be compartmentalized inside separate services, therefore enabling the application of exact and focused security measures as required and improving general system protection.
- **Reliability:** Key expectations are guaranteeing system availability and avoiding data loss. Firebase's managed services offer strong solutions by means of high availability and data redundancy, therefore protecting against possible disturbances. The independent architecture of microservices also improves general system resilience since the failure of one service is improbable to affect the operation of the whole system. In dynamic settings, this modular strategy helps to preserve operational continuity and dependability.
- **Maintainability:** Using microservices architecture is more efficient for managing the complexity of a system integrating various functions, including user features, retail operations, and artificial intelligence processing. Scalability and maintainability are greatly improved by breaking the system into smaller, specialized services. The serverless architecture of Firebase Functions also simplifies deployment and reduces the requirements of infrastructure management, therefore enabling more effective system updates and long-term sustainability.

Careful consideration of these architectural drivers during the design phase directly guided the choice of microservices architecture based on the Firebase serverless platform, therefore guaranteeing the system is well-equipped to satisfy both functional and non-functional needs.

5.7 UML System Package Diagram

Visualizing and recording the system design from several angles, UML diagrams are vital tools. By showing the interactions between external actors such as the Normal User and Retailer and the system's use cases, the UML Use Case Diagram (Section 4.4.1) provides a high-level view of the system's functionality. From the user's viewpoint, it specifies the limits of the system and describes what the system is meant to accomplish. By showing interactions between objects or components in a particular scenario over time, the Sequence Diagrams (Section 4.4.2) provide a detailed view of the dynamic behavior of the system. These diagrams depict the message sequence exchanged between the mobile app, backend microservices (Firebase Functions), and integrated services (Firestore, Cloud Storage, Document AI) in order to complete a particular task. Presented below, the UML Package Diagram offers a high-level, static perspective of the system's structure. It shows the dependencies and organizes related components into logical packages. Conveying the modular architecture of the system by means of this diagram is essential since it divides it into areas including the Mobile Application, Backend Services (Microservices), Firebase Services, and External Services. It clarifies how various system components interact and support the general functionality and interaction models represented in the Use Case and Sequence Diagrams. These UML diagrams taken together provide a complete picture of the system design from external user interactions and runtime behaviors to its internal structural organization providing a whole knowledge of how the system is meant to satisfy its needs.

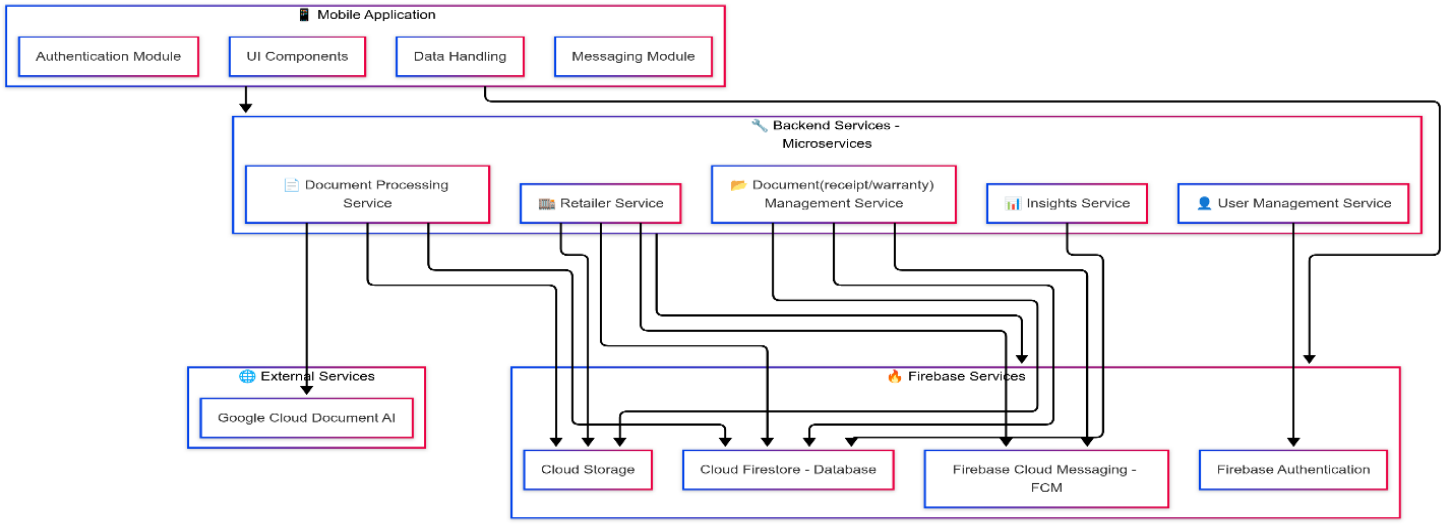

Figure 22: System Package diagram

Chapter 6
System Implementation and Testing

This chapter describes how to turn the Receipt and Warranty Management System design into a functional system. It explains how the functional application was created by selecting and integrating the several components specified during the design phase. Moreover, this chapter offers a critical analysis of the system's strengths and weaknesses as well as the findings from the testing techniques used to assess the performance, usability, and compliance to the stated criteria of the system.

6.1 System Implementation

The implementation stage stressed converting the architectural design and functional requirements into fully functioning system. This involved building the system's basic features and setting up the development environment including the selected technologies.

6.1.1 Technologies and Components Selection and Integration

Based on the system design (Chapter 5), the following key components were selected and integrated:

- **Mobile Application (Flutter):** Flutter was used to create the user interface. This meant designing the screens for document capture, list viewing, searching, insight display, reminder setting, and claim requests. Connecting the frontend with the backend services required integration with Firebase SDKs for Authentication, Firestore, and Cloud Storage.
- **Firebase Authentication:** User sign-up, login, and session management were handled by integrating the Firebase Authentication SDK into the Flutter app. This made it possible for safe access to user-specific data kept in Cloud Storage and Firestore.
- **Cloud Firestore:** Firestore SDK interacted with the database via Firebase Functions and Flutter app. To represent receipts, warranties, users, etc., data models were created in Dart (for Flutter) and Node.js (for Functions), therefore enabling the storage and retrieval of structured data.
- **Cloud Storage for Firebase:** Document file (images, PDFs) uploading was managed by including the Cloud Storage SDK into the Flutter app. To start the data extraction process, Firebase Functions were set to fire on file uploads.

- **Firebase Functions (Node.js):** Node.js was used to create the backend logic as serverless functions. Separate functions were created for particular activities including processing document uploads, calling the Document AI API, computing spending insights, sending reminder notifications (triggered by scheduled functions or database changes), and managing warranties and receipts. These functions interact with Firestore, Cloud Storage, and outside services including Document AI.
- **Firebase Cloud Messaging (FCM):** Push notifications were obtained by including the FCM SDK into the Flutter app. Notifications were sent using Firebase Functions via FCM
- **Google Cloud Document AI:** Integration with Document AI was accomplished by utilizing API calls from a dedicated Firebase Function. This function gets document files (or references) from Cloud Storage, forwards them to Document AI for processing, and gets the extracted data, a JSON file with all the relevant information extracted from the document; then, the required information is stored in the database and the JSON file is kept in the storage for future reference.

Configuring the Firebase project, establishing required permissions, developing the mobile app and Firebase Functions, and creating the communication between these components using APIs and event triggers constituted the integration process.

6.2 System Testing

Testing is a critical phase to ensure the implementation meets the specified requirements, is stable, and provides good user experience. The testing process involved several phases to evaluate different aspects of the system.

6.2.1 Testing Phases and Results

The testing phases include:

- **Unit Testing:** Individual Firebase Functions and key components of the Flutter application were tested in isolation to ensure they performed their specific tasks correctly.
    - Results*:* errors detection and correction during development, identified and fixed bugs in data processing logic, API calls, and UI components during development phase.
- **Integration Testing:** Tested the interactions between different components, such as the mobile app communicating with Firebase Functions, Functions interacting with Firestore and Cloud Storage, and Document AI integration.
    - *Results:* Verified the flow of data between components and identified issues with data serialization/deserialization and API endpoints.
- **Functional Testing:** Tested each functional requirement (as outlined in Section 4.2.1) to ensure the system performs the required actions correctly from the user's perspective.

This included testing document capture (scanning, upload), viewing, searching, setting reminders, etc.

- Results*:* Confirmed that core functionalities worked as expected, some exceptional cases in data extraction and search filtering were identified and addressed (tested & fixed).

- **Non-Functional Testing:** Evaluated aspects like performance, security, and basic usability.
    - Results*:* Initial performance tests showed acceptable response times for key operations under load. Security testing focused on authentication and data access controls, confirming that unauthorized access was prevented.
    - The following is the result of the load test and performance analysis

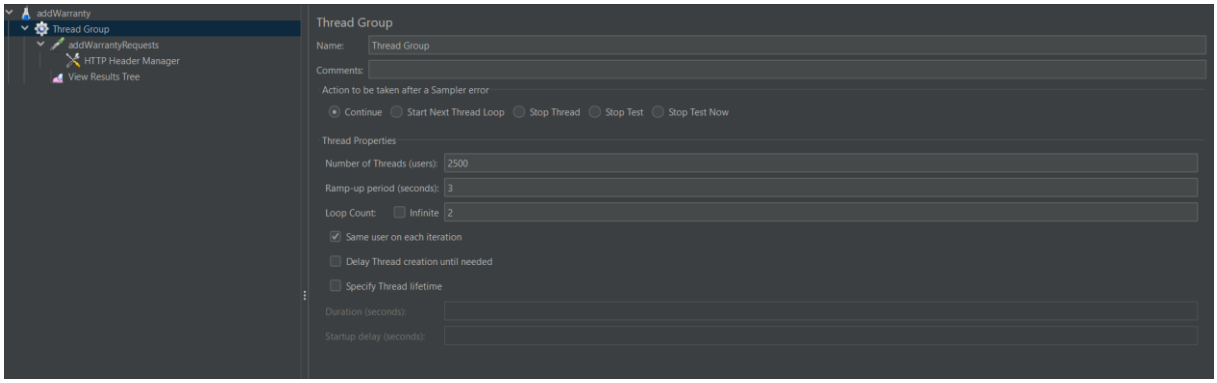

Figure 23: Setting up thread group

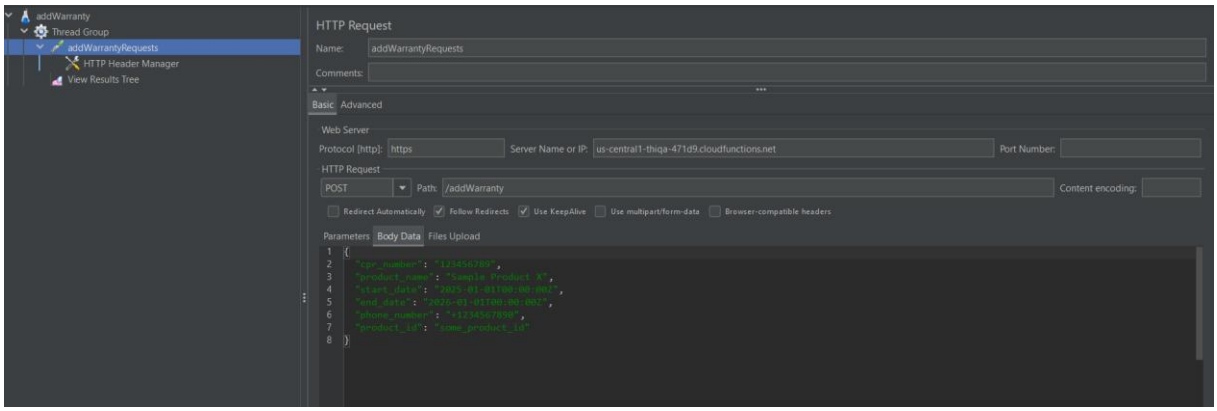

Figure 24: Http request add warranty test

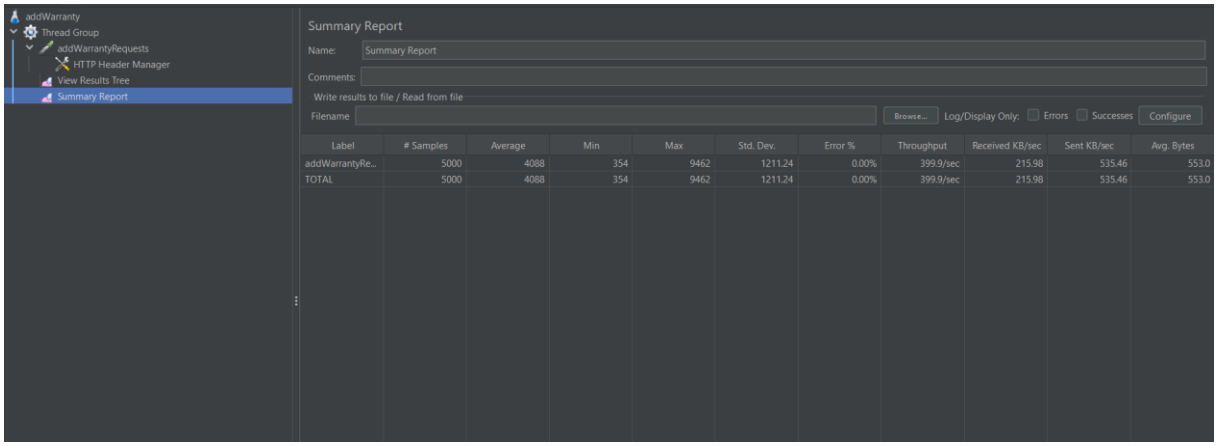

Figure 25: Test result report

Load Testing summary shows that apache JMeter ran 5000 requests to conduct a load test on the addWarranty Firebase Cloud Function. One notable result of this research was the discovery of a 0.00% error rate. Every request under the simulated load was successfully managed without errors, suggesting the function showed significant robustness. Moreover, the function showed a fair processing capacity, attaining a throughput of roughly 400 requests per second. This suggests the capacity to suitably handle a large number of simultaneous requests. Our study did, however, show significant variations in the time required to handle these requests. Though request times varied greatly from a minimum of 0.354 seconds to a maximum of 9.467 seconds, the average response time was about 4.088 seconds. This significant variation, which is supported by a fairly high standard deviation, implies that even if the function controls the load volume, the consistency and speed of individual request processing could vary which led to reconsider the function implementation and implement indexing to faster the documents retrieval process. Ultimately, the addWarranty Cloud Function was dependable and able to handle the number of requests produced during this test. The average response time and the observed variation, however, indicate that there are areas where performance optimization could enable more consistent and maybe faster performance for all users of this service. Every other retailer API service in our system was subjected to this same load testing technique to obtain a full understanding of their performance.

6.2.2 Discussion of Results and Comparison

The test findings show that the installed system effectively shows the basic features of the Receipt and Warranty Management System. Developing and integrating the several parts was made easier by Firebase's microservices architecture. The microservices approach enabled parallel development and simpler debugging of individual features when compared to conventional monolithic architectures. Using managed Firebase services cut the work needed for infrastructure management by much over configuring and running bespoke servers. Perfecting the AI-powered data extraction over a wide range of receipt and warranty formats, however, proved difficult. Although Document AI is strong, reaching 100% accuracy calls for strong error management and maybe user-guided correction systems.

6.2.3 Usability and User-Experience Testing

Usability and user-experience (UX) testing was conducted with a group of target users, based on the defined personas. This involved observing users interacting with the system, while performing typical tasks (e.g., scanning a receipt, searching for a warranty, claiming warranty) and gathering their feedback through interviews and questionnaires.

- **Methodology:** Participants were given a set of tasks to complete using the system. Their interactions were observed, and think-aloud protocols were used to understand their thought process. Post-task questionnaires and interviews focused on ease of use, satisfaction, and suggestions for improvement.
- **Results:**

- **Strengths:** Users found the concept highly valuable and appreciated the potential to reduce physical clutter and easily find documents. The process of capturing documents was generally well-received, particularly the scanning feature.
- **Weaknesses:** Some users found the initial setup and understanding of certain features (like P2P agreements) less intuitive. The accuracy of AI extraction varied depending on the document quality, sometimes requiring manual correction which users found tedious. Navigation between different sections could be improved based on user feedback.
- **User Experience:** Overall user experience was positive for core tasks, but the system highlighted areas where the user flow could be smoother and error handling more user-friendly. The point of having all documents in one place resonated strongly with users.

6.2.4 Strengths and Weaknesses of the Proposed System

Based on the implementation and testing of the system, the key strengths and weaknesses of the proposed system are:

- **Strengths:**
  - **Addresses a clear user need:** The system directly tackles the common problem of managing and finding receipts and warranties.
  - **Utilizing powerful cloud services:** Using Firebase and Google Cloud provides scalability, reliability, and advanced AI capabilities which all used in the developed system.
  - **Microservices Architecture:** Offers benefits in terms of scalability, resilience, and maintainability for future development.
  - **Cross-Platform Mobile Application:** Developed with Flutter, ensuring compatibility with different operating systems (android, IOS).
  - **Potential for Automation:** AI extraction and automated reminders significantly reduce manual tasks for users.
- **Weaknesses:**
  - **AI Extraction Accuracy:** Reliably extracting data from all possible receipt and warranty formats remains a challenge and requires robust handling of inaccuracies and may need user review of the extracted documents.
  - **Integration Complexity:** Integrating multiple Firebase services and external APIs adds complexity to the development and environment setup processes.
  - **Dependency on Cloud Provider:** The system is heavily reliant on the availability and pricing of Firebase and Google Cloud services.

- **User Adoption:** Encouraging users to consistently capture all their documents requires a seamless and rewarding user experience.
- **Retailer Integration:** Implementing and onboarding retailers for direct integration with Thiqa system presents a significant logistical challenge.

Overall, the system implementation and testing validated the core concept and architectural approach while also highlighting key areas for improvement and future development to enhance the system's robustness and user experience.

Chapter 7
Conclusion and Future Work

From initial design to implementation and testing, this chapter finishes the report by summarizing the thorough effort done all over the project for the Receipt and Warranty Management System. By proving the feasibility of a microservices architecture based on Firebase, the project successfully converted functional system into identified user needs. Presented are important findings and results from the development and testing stages, therefore stressing the system's capacity and the relevance of the results in solving the issue of handling personal receipts and warranties.

7.1 Project Summary

The project produced an operational working version of a Receipt and Warranty Management System by effectively navigating the important stages of software development. Primarily via a user survey, the Requirement Elicitation phase (Chapter 4) set the groundwork by pinpointing the fundamental issue of managing and retrieving personal receipts and warranties. The survey results gave important quantitative data confirming the frequency of the issue and underlining important user needs and priorities including the need for instant digital storage, efficient scanning with artificial intelligence extraction, fast search capabilities, and warranty expiration reminders. Defining the system's functional and non-functional needs was greatly aided by this stage. These needs were converted into a technical blueprint during the System Design stage (Chapter 5). Chosen was a Microservices Architecture mostly built on Firebase services Authentication, Firestore, Storage, Functions, FCM and combined with Google Cloud Document AI. Its capacity to satisfy the criteria for scalability, performance, dependability, and maintainability justified this architectural style combined with the serverless character of Firebase Functions. Along with the application of UML diagrams Use Case, Sequence, Package to visualize the system's structure and behavior, this phase also defined the data design, user interface strategy (mobile-first with Flutter), and core system processes. Focusing on constructing the working system depending on the design, the System Implementation and Testing phase (Chapter 6) This included using Flutter to create the mobile app, combining the chosen technologies, and using Firebase Functions to run the backend microservices. To confirm the system's function and performance, rigorous testing including

unit, integration, functional, and non-functional tests was carried out. Importantly, user-experience and usability testing gave insightful comments on the system's simplicity and pointed out areas for development from the viewpoint of the intended users.

7.2 Main Results and Findings

The work produced several notable outcomes and discoveries. The survey data strongly supported the first assumption regarding the challenges in handling receipts and warranties, so verifying a real need for a technological solution that moves toward digitalizing the whole process of post purchases documents. Guiding the development toward fundamental functions like capture, storage, search, and reminders, the survey clearly highlighted and prioritized the aspects most appreciated by possible users. Demonstrating the advantages of scalability and managed services during implementation, the selected Microservices Architecture on Firebase turned out to be a practical and efficient way to construct the system. Successfully implemented was a working system showing key features including securely storing warranties, seamlessly capturing receipts at purchase time, viewing, searching, and setting reminders. While stressing particular user interface and workflow areas that need improvement for better user experience, usability testing gave direct user feedback validating the intuitive character of fundamental interactions; the value proposition of the system struck a chord with users. Though issues in attaining universal accuracy were mentioned, the connection with Google Cloud Document AI showed the possibility for automated data extraction, a main feature for lowering manual effort. These results are important because they confirm that a system addressing the stated pain areas is technically possible to implement using current cloud-based and microservices architectures and wanted by users. The usability insights provide specific direction for improving the user interface and experience, and the prioritized features give a clear road map for next development.

7.3 Project Limitations:

1. Legal and Compliance Hurdles: Bahrain's data protection rules, might call for more compliance work on how customer data especially financial records and personal identifiers is stored, handled, and shared.
2. Minimal POS and Bank System Integration: Bahrain's absence of open banking infrastructure or retailer-side connections could restrict the ability to automatically retrieve receipts or confirm warranties without manual user involvement.
3. Time Limitations: Not all intended features and sophisticated security measures could be implemented given limited development time, which could compromise the system's stability, scalability, and user experience in early releases.
4. Restricted Domain Knowledge: The project lacks thorough operational expertise of the retail and banking sectors, which could affect the app's fit with real-world operations including POS integrations, return procedures, and financial reconciliation criteria.
5. Android Only Platform Restriction: Due to limited devices, testing conditions, and iOS development resources, the app has been launched just for Android.
6. Privacy-Sensitive User Identification: Thiqa requires users to provide sensitive identifiers like CPR numbers or payment card details for full functionality. This raises privacy concerns and may limit user trust and adoption.

7.4 Future Work:

While the current system of the Receipt and Warranty Management System demonstrates the core functionalities and validates the concept, several enhancements and advanced features are planned to elevate its utility, scalability, and intelligence. These potential improvements span both individual users and organizational stakeholders, and focus on personalization, automation, data-driven insights, and strategic integrations.

7.4.1 Enhanced User-Centric Features

To further empower users in managing their finances and documents, the following features are proposed:

- **Potential Savings Opportunities**: By analyzing users' historical spending patterns, the system can suggest personalized saving tips and identify categories where expenditures could be optimized.
- **Natural Language Queries & Conversational Insights**: Implementing a conversational interface, such as a chatbot or voice assistant, would allow users to interact with their financial data in a more intuitive manner. For example, users could

ask questions like “How much did I spend on electronics in March?” and receive contextual visual responses.

- **Smart Reporting & Narrative Insights**: Users will be able to generate custom reports using natural language commands, such as “Generate a report of my office supply expenses this year.” Additionally, the app will present AI-generated narrative summaries alongside charts, making financial insights more understandable and accessible to non-expert users.
- **Bar/Qr Code Product Warranty Search**: Implement a scanner feature in the app that allows users to scan product QR codes. Once scanned, the app will cross-reference the QR code with the stored warranty data and present the relevant warranty details.
- **Image Recognition Warranty Search:** Add image recognition functionality to identify products through pictures. You could leverage machine learning models to match the product image with stored records and retrieve warranty information.

7.4.2 Advanced Analytics for Organizations

Expanding the system’s capabilities to support businesses and warranty service providers introduces several powerful features:

- **Benchmarking Tools**: Organizations will be able to compare their warranty claim and renewal metrics against industry benchmarks. The system will highlight outliers and provide actionable insights (e.g., “Claim rate for small appliances is 15% above average”).
- **Demand Forecasting**: Using historical data and AI models, the system will forecast trends in warranty renewals and product demand. This will help organizations anticipate consumer needs and plan marketing strategies or inventory accordingly.
- **Customizable Reports**: Businesses will be able to generate tailored reports on receipt and warranty data, export them in professional formats (PDF, Excel), and use them for internal analysis, investor presentations, or compliance.
- **AI-Driven Warranty Prediction:** Use AI models to analyze product data, historical warranty claims, and industry trends to predict the optimal warranty period for new products. This will help businesses offer more accurate warranty durations, which could improve customer satisfaction and reduce costs associated with premature or extended warranty periods.

7.4.3 Technical and Infrastructure Improvements

To ensure the system remains scalable, secure, and user-friendly as it evolves, several technical enhancements are planned:

- **AI-Powered Error Correction**: Improve the accuracy of extracted data from documents by introducing user-assisted correction workflows and self-improving models based on correction feedback.
- **Multi-Language Support**: Extend accessibility by supporting multiple languages for both the interface and document recognition, allowing broader international adoption.
- **Customer Identification via QR Code/Card/Payment Integration:** For seamless receipt or warranty addition, allow users to link their payment methods or cards to their accounts. When users make a purchase at a POS, the QR code or payment card data can be used for automatic receipt and warranty linking. Alternatively, a unique

customer ID stored in the system could be used to auto-populate warranty and receipt data when identified at checkout.

7.4.4 Community and Ecosystem Development

- **P2P Reputation System**: Further develop the peer-to-peer warranty trust system by allowing users to rate each other, and build reputational scores based on successful P2P warranties and interactions.
- **User Education Hub**: Create a learning center within the app with guides and tips on financial literacy, warranty rights, and how to make the most of the system's features.
- **Adding Insurance Integration in the Application**: Integrating insurance options into the Receipt and Warranty Management System would allow users to manage not only their warranties but also the insurance coverage for their products. Users could easily add insurance details for their products, track insurance claims, and renew their policies through the app.

By implementing these future enhancements, the Receipt and Warranty Management System aims to evolve into a comprehensive digital assistant for personal finance and document management, while also offering strategic value to businesses in the warranty and after-sales service ecosystem.

References

1. Stephen, V.K., & Mathivanan, V. (2017). An Advancement in Paper Receipts the Electronic Receipt Administration Framework. *Indonesian Journal of Electrical Engineering and Computer Science*, *8*(3), 631-632.
2. Ministry of Health, 2024. Green IT. [online] Ministry of Health, Kingdom of Bahrain. Available at: https://www.moh.gov.bh/Services/GreenIT [Accessed 4 May 2025].
3. Lin, C.-J., Liu, Y.-C., & Lee, C.-L. (2022). Automatic Receipt Recognition System Based on Artificial Intelligence Technology. *Applied Sciences*, *12*(2), 853.
4. Gill, A. (2023). Developing a Centralized Receipt Service to Streamline Restaurant Operations and Reduce Costs. *Journal of Engineering and Applied Sciences Technology*, *5*(6), 1-12.
5. Green America (2020) Skip the Slip Report: Environmental Costs & Human Health Risks of Paper Receipts. [Online] Washington, DC: Green America. Available at: https://www.greenamerica.org/sites/default/files/2020-10/Skip%20The%20Slip%20Report%202020%20%28GA%29.pdf [Accessed 7 March 2022].
6. FasterCapital, n.d. *Receipts: The Importance of Organized Receipts in Petty Cash Management*. [online] Available at: https://fastercapital.com/content/Receipts--The-Importance-of-Organized-Receipts-in-Petty-Cash-Management.html#Which-is-Better-.html [Accessed 4 May 2025].
7. Stephen, V.K. & Mathivanan, V., 2017. An Advancement in Paper Receipts the Electronic Receipt Administration Framework. *Indonesian Journal of Electrical Engineering and Computer Science*, 8, 631-632.
8. DocuClipper (2025) *Receipt processing and data extraction software*. Available at: https://www.docuclipper.com (Accessed: [3 May 2025]).
9. DocuClipper (2025) *Receipt data extraction: Extract receipt data*. Available at: https://docuclipper.com/blog/receipt-data-extraction (Accessed: 4 May 2025).
10. Tabscanner (2024) *OCR receipt scanner API: Extract data from receipts*. Available at: https://tabscanner.com/ocr-receipt-scanner (Accessed: 4 May 2025)

11. Claimlane (2025) *Warranty Management Software: Automate claims & returns*. Available at: https://claimlane.com/warranty-management-software (Accessed: 4 May 2025)

Appendix A

Retailer Documentations

You can find the retailer API documentation in depth here:

Retailer API Documentation.pdf

Class Diagram Image

Here is the class diagram with full resolution:

class diagram.png