上海交通大学学位论文

# DocuSnap：你的 AI 智能文档助手

**姓　　名**：杨子骏，彭靖嘉，周梓茗，曲民扬，唐荟杰

**学　　号**：521370910109, 521370910005, 521370910142, 521370910181, 521021910711

**导　　师**：皮宜博

**学　　院**：密西根学院

**专业名称**：电子与计算机工程

**申请学位层次**：学士

**2025 年 7 月**

**A Dissertation Submitted to**

**Shanghai Jiao Tong University for the Degree of Bachelor**

# DOCUSNAP: YOUR AI-POWERED PERSONAL DOCUMENT ASSISTANT

**Author: Yang Zijun, Peng Jingjia, Zhou Ziming, Qu Minyang, Tang Huijie**

**Supervisor: Pi Yibo**

University of Michigan-Shanghai Jiao Tong University Joint Institute

Shanghai Jiao Tong University

Shanghai, P.R. China

July, 2025

# 上海交通大学
## 学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，已在文中以适当方式予以致谢。若在论文撰写过程中使用了人工智能工具，本人已遵循《上海交通大学关于在教育教学中使用 AI 的规范》，确保人工智能生成内容的应用场景、引用范围及标注方式均符合规定，并杜绝学术不端行为。本人完全知晓本声明的法律后果由本人承担。

学位论文作者签名：Zijun Yang Jingjia Peng [illegible] Minyang Qu Huijie Tang

日期：2025 年 7 月 29 日

# 上海交通大学
## 学位论文使用授权书

本人同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。

本学位论文属于：

☑**公开论文**

☐**内部论文**，保密☐ 1 年/☐ 2 年/☐ 3 年，过保密期后适用本授权书。

☐**秘密论文**，保密____ 年（不超过 10 年），过保密期后适用本授权书。

☐**机密论文**，保密____ 年（不超过 20 年），过保密期后适用本授权书。

（请在以上方框内选择打“✓”）

学位论文作者签名：Zijun Yang Jingjia Peng [illegible] Minyang Qu Huijie Tang　　指导教师签名：[illegible]

日期：2025 年 7 月 29 日　　日期：2025 年 8 月 7 日

# 摘 要

本文介绍了 DocuSnap，一个 AI 驱动的个人文档助手，旨在解决当今纸质-数字混合生态系统中管理个人文档数字副本的挑战。传统的数字文档管理解决方案将文档视为静态图像或非结构化文本，无法支持语义检索、自动组织和表单填写等实际任务。DocuSnap 结合计算机视觉、光学字符识别和大语言模型，将原始文档解析为结构化的可操作信息。

系统架构采用隐私优先的方法，具有端到端加密和设备本地图像处理能力。主要创新包括基于 LLM 的语义理解用于智能分类和跨文档链接、支持直观文档检索的查询方法，以及使用相关文档提取数据的自动表单填写功能。该移动应用程序使用 Android 的 Jetpack Compose 框架构建，提供直观的用户界面和先进的图像处理功能，包括透视校正和对比度增强。

通过全面的用户访谈和可用性测试，我们识别了现有文档管理工作流程中的关键痛点，并验证了我们的设计决策。系统通过广泛测试进行评估，成功实现了几何校正、图像增强、加密存储和智能信息提取等核心功能。在安卓模拟器和真机上的性能测试确认了可接受的响应时间：文档解析平均耗时 20 秒，一般 UI 交互流畅响应。

DocuSnap 通过将强大的计算机视觉技术与 AI 驱动的语义理解相结合，在个人文档管理方面取得了重大进展，为用户提供了一个全面的解决方案，弥合了物理文档捕获与智能数字组织之间的差距，同时保持严格的隐私和安全标准。

**关键词：** 移动应用，AI 文档理解，文档图像增强，端到端数据安全

# ABSTRACT

We present DocuSnap, an AI-powered personal document assistant designed to address the challenges of managing digital copies of personal documents in today's hybrid physical-digital ecosystem. Traditional document management solutions treat documents as static images or unstructured text, failing to support real-world tasks like semantic retrieval, automatic organization, and form filling. DocuSnap introduces a novel five-stage pipeline that combines computer vision, optical character recognition (OCR), and large language models (LLMs) to transform raw document captures into structured, actionable information.

The system architecture employs a privacy-first approach with end-to-end encryption and on-device image processing capabilities. Key innovations include LLM-based semantic understanding for intelligent categorization and cross-document linking, keyword searching support for intuitive document retrieval, and automated form filling using extracted data from related documents. The mobile application, built using Android's Jetpack Compose framework, provides an intuitive user interface with advanced image processing capabilities including perspective correction and contrast enhancement.

Through comprehensive customer interviews and usability testing, we identified critical pain points in existing document management workflows and validated our design decisions. The system was evaluated through extensive acceptance testing and demonstrated successful implementation of core features including geometric correction, image enhancement, encrypted storage, and intelligent information extraction. Performance testing on emulator and low-end hardware confirmed acceptable response times, with document parsing averaging 20 seconds and general UI interactions being responsive.

DocuSnap represents a significant advancement in personal document management by combining robust computer vision techniques with AI-driven semantic understanding, offering users a comprehensive solution that bridges the gap between physical document capture and intelligent digital organization while maintaining strict privacy and security standards.

**Key words:** Mobile App, AI Document Understanding, Document Image Normalization, End-to-end Document Security

# Contents

# Chapter 1 Introduction

## 1.1 Background

In today's digitized world, personal documents like passports, driver's licenses, healthcare cards, academic records, and receipts are indispensable for navigating daily life. These materials serve as proof of identity, enable access to services, and validate financial or legal transactions. Yet, despite their critical role, individuals face persistent challenges in managing digital copies of these documents—a problem rooted in the gap between the need for organized, high-quality digital records and the tools available to create and manage them.

The problem begins with the duality of document formats. While institutions increasingly adopt digital systems, many personal documents still originate in physical form. Individuals are left to bridge this gap through makeshift solutions: snapping photos, scanning IDs, or hoarding paper records. This hybrid ecosystem creates fragmentation. Physical documents are prone to misplacement or damage, while digital copies are often disorganized—rendered unusable due to poor image quality or hidden among vague filenames like "IMG_1234.jpg".

Personal documents span a wide range of formats and use cases, such as passports with biometric data, receipts with itemized purchases, and insurance cards with policy numbers. Generic storage tools lack awareness of these distinctions, forcing users to adopt fragmented, inefficient workflows.

Moreover, retrieving documents in real time remains a hurdle—especially in urgent situations like travel or medical visits—exacerbated by poor categorization, keyword-unfriendly filenames, or lack of intuitive search. As institutions increasingly expect digital submissions, the need for seamless capture, organization, and access becomes more pressing.

## 1.2 Problem Statement

Despite the growing digitization of services, individuals still struggle to manage the vast array of personal documents—such as IDs, receipts, certificates, and health records—in a way that is organized, secure, and immediately accessible. The problem is not merely one of digitization, but of **intelligent digital stewardship**. Existing tools treat documents as static images or unstructured text, failing to support real-world tasks like retrieving a tax form on demand, tracking expiration dates, or organizing documents by purpose. As a result, users are forced to adopt fragmented, error-prone workflows involving a mix of smartphone photos, generic folders, cloud apps, and manual spreadsheets.

At its core, the problem is the lack of an integrated system that can: (1) robustly digitize and clean up physical documents, (2) extract and structure critical data across a wide variety of formats, (3) organize documents meaningfully without user input, (4) enable semantic, natural-language-based retrieval, and (5) do all of the above in a secure, privacy-preserving way on mobile devices.

To solve this problem, several research and engineering challenges must be addressed:

- **Image Preprocessing:** Document photos taken with mobile devices often include perspective distortions, glare, shadows, or physical imperfections (e.g., wrinkles). Effective preprocessing requires real-time, on-device algorithms capable of perspective correction, edge detection, glare reduction, and contrast enhancement that work across varied lighting and surface conditions.
- **Robust and Structured OCR:** Optical character recognition systems typically output flat text, which is insufficient for documents with complex layouts (e.g., tables in receipts, labels in forms, or multi-column formats). Extracting semantically meaningful key-value pairs requires layout-aware OCR and post-processing logic that can recover the logical structure and relationships embedded in the document[1-2].
- **Semantic Understanding and Retrieval:** Traditional systems rely on keyword-based search and metadata matching, which fail when users query using vague or contextual language (e.g., “my son’s insurance card” or “the tuition receipt from June”). Effective retrieval must incorporate natural language understanding and be aware of document types, fields, and temporal or relational cues.
- **Privacy-Preserving AI:** Because personal documents often contain sensitive or regulated information (e.g., IDs, medical records, financial data), all computation—including OCR, semantic extraction, and indexing—must be performed locally without uploading unencrypted files to cloud servers. This imposes constraints on model size, memory usage, and runtime performance.
- **Secure Indexing and Access Control:** A usable system must encrypt documents

and metadata while still enabling fast lookup, expiration tracking, and contextual reminders. Designing encrypted yet queryable indices poses a trade-off between security, performance, and usability.

Solving these challenges requires a novel, end-to-end approach that tightly integrates mobile computer vision, layout-aware information extraction, lightweight language models, and user-centric design principles under strict privacy constraints.

## 1.3 Existing Solutions and Their Drawbacks

Current approaches to personal document management fall into three general categories: manual storage, scanner-centric apps, and passive photo organization. Each addresses fragments of the broader challenge, but none offer a cohesive, intelligent solution that adapts to the diversity and urgency of real-world document needs.

The most common strategy is **manual photo organization**, where users take pictures of documents with their smartphones and manually sort them into folders or albums. This method provides minimal automation and lacks essential capabilities like text extraction, metadata tagging, or semantic search. As a result, users often face difficulty locating critical documents when needed and must rely on memory or ad hoc labeling conventions, leading to inefficiencies and errors.

Another class of tools focuses on **scanner-centric workflows**. These applications prioritize high-quality image capture, offering features such as auto-cropping, contrast enhancement, and basic OCR. While effective for digitizing documents into clean, readable formats, they stop short of understanding document semantics. OCR output is often raw and unstructured, requiring manual correction or post-processing. Moreover, these tools typically lack intelligent organization capabilities, forcing users to sort documents into rigid folder systems without automated tagging, contextual grouping, or deadline tracking. Additionally, many such apps operate through cloud-based services, raising privacy concerns over the handling of sensitive data[3-4].

A third category includes **passive photo organization systems**, such as those built into smartphone operating systems. These tools use image classification and on-device OCR to surface document-like images and allow keyword-based search. However, they treat documents as generic images, offering no structure extraction, customization, or document-specific features. Their capabilities are often limited to superficial detection, lacking support for field-level parsing (e.g., expiration dates or totals) and providing little control over how documents are categorized or secured[5].

Additionally, some tools emphasize **ecosystem-bound productivity integration**, embedding scanning functions into larger office suites. While these solutions streamline workflows for users deeply embedded in specific ecosystems, they lack portability, cross-platform support, and general-purpose document intelligence. Features such as semantic search, smart categorization, and cross-document relationships are largely absent, as these tools are designed to augment productivity pipelines, not to manage diverse personal records comprehensively[6].

Across all these categories, common limitations emerge. Most tools treat documents as either images or raw text, failing to extract and organize structured information. Few support natural language queries or intelligent reminders. Privacy and security are often afterthoughts, with limited on-device processing or encryption options. Ultimately, while these tools solve parts of the problem—scanning, OCR, or storage—they lack a unified, intelligent, privacy-conscious solution that empowers users to manage personal documents as dynamic, searchable assets.

## 1.4 Proposed Solution

**DocuSnap** is a next-generation document management system that leverages large language models (LLMs) to overcome the limitations of existing tools. Unlike traditional apps that treat documents as static images or unstructured text, DocuSnap introduces true document intelligence by combining semantic understanding with secure, user-controlled processing.

At the core of DocuSnap is a five-stage pipeline designed to transform raw captures into structured, actionable information[7]. First, the system performs **image enhancement**, applying computer vision techniques such as perspective correction, lighting normalization, and auto-cropping to clean up user-captured images in real time. Next, an efficient **OCR extraction** engine processes the enhanced images to extract raw text, serving as the input to the subsequent stages.

The third stage—**LLM structuring**—is where DocuSnap sets itself apart. A large language model, accessed via Zhipu AI's secure backend API, parses the OCR output and organizes it into structured JSON objects tailored to the document type. For instance, a scanned receipt might be transformed into:

```
{"document_type": "receipt", "total": "$49.99"}
```

This enables direct use in finance or tracking applications. In the **semantic tagging** stage, documents are automatically labeled based on their content, context, and urgency[8]. Labels such as "medical bill", "passport", or "2023 tax-related" are inferred without user input, reducing organizational overhead.

To preserve user privacy, DocuSnap implements a robust encryption scheme: all document content and associated metadata are encrypted locally on the user's device before being sent to the backend service. Only the corresponding user possesses the cryptographic key necessary to decrypt the response, ensuring, at the same time, supports for asynchronous result retrival and that plaintext is never stored on disk. This architecture provides both the scalability and intelligence of backend LLM inference, and the privacy guarantees typically associated with on-device AI.

Finally, DocuSnap supports retrieval through flexible queries. Users can search for documents using expressions like "insurance card" or "contract from", bypassing the need to remember file names or specific metadata[9].

DocuSnap introduces several key innovations that enable this functionality. The use of **LLM-based semantics** allows the system to go beyond surface-level text recognition, understanding document structure and extracting meaning. By using a **secure encryption and access control scheme**, DocuSnap ensures that user data remains confidential, even during backend processing. The system also supports **cross-document linking**, intelligently associating related documents such as a student ID and a visa form, or a receipt and its associated warranty. Above all, DocuSnap is designed with a **user-centric philosophy**, offering an intuitive interface that avoids feature bloat and empowers users with minimal technical effort.

In essence, DocuSnap reimagines document management as more than just digital storage. It is a dynamic, intelligent assistant that understands, organizes, and retrieves documents in ways that align with how people actually use them in modern life.

# Chapter 2 Design Specification

## 2.1 Customer Requirements

Our intended users include students (undergraduate and graduate), university staff, and early-career professionals, particularly those who regularly manage administrative, academic, or legal documents. These users often navigate multiple document formats, such as PDFs, images, and physical papers, and require reliable access to organized content, especially when on the go or facing time constraints.

### 2.1.1 User Interview Questionnaire

To better understand our target customers, we developed a user interview questionnaire centered around several **key objectives**. Firstly, we aimed to validate the core pain points users experience with document management in their daily routines. Gaining insight into these challenges is essential for designing solutions that effectively address user needs. Secondly, we wanted to identify existing workaround solutions that users may have adopted, as well as their limitations. Understanding what does not work in the current landscape enables us to pinpoint areas for improvement. Lastly, we sought to uncover unmet needs and generate ideas for innovative features that would enhance the overall user experience.

In crafting the questionnaire, we adhered to specific **design principles** to maximize its effectiveness. We prioritized open-ended, experience-driven questions to elicit qualitative insights that delve deeper into user behavior. Additionally, we structured the questions to flow logically, beginning with broader background inquiries and narrowing down to specific pain points. We also included prompts for real-life examples to encourage respondents to share concrete user behaviors and edge cases, enriching our data with practical insights. A detailed breakdown of the questions can be found in Appendix 1.

### 2.1.2 Customer Interviews

#### 2.1.2.1 Interviewees

Our interview participants represented a **diverse demographic and occupational spectrum**, with seven students, three industry professionals, and varying levels of experience.

The students included undergraduates, master's, and PhD candidates from institutions such as the University of Michigan, UIUC, the University of Washington, and the UM-SJTU Joint Institute. They commonly manage documents related to academics, visas, applications, and reimbursements. The industry professionals included engineers from companies such as Meta and Alibaba, as well as a working professional managing both personal and professional documents, including tax forms and reports. This broad representation enabled us to capture a range of insights, particularly since many participants were currently engaged in internships or cross-border academic programs that require extensive document handling.

#### 2.1.2.2 Interview Logistics

Interviews were conducted either remotely via Zoom or in person, lasting approximately 30 to 45 minutes each. Three members of our team were involved in scheduling, performing, and extracting insights from the interviews. We followed a structured approach covering various topics, including document search behavior, preferences for scanning tools, frustrations with OCR, transitions from physical to digital documents, and desires for an ideal document management assistant.

The venues for our interviews varied significantly, taking place in university settings such as dorm rooms, labs, and libraries for students, as well as in home offices or corporate environments for professionals. Many participants had recently undergone document-heavy processes, such as graduate school applications or internship onboarding, which provided rich context and recall of fundamental challenges faced during these times.

The insights gleaned from these interviews were invaluable, revealing how individuals manage both digital and physical documents, their expectations for existing tools, and the potential value of intelligent assistance. This qualitative data directly informed our feature ideation and problem framing for DocuSnap.

### 2.1.3 Affinity Map

Our interview findings are summarized on our Miro board, and a complete visualization of the affinity map appears in Appendix 2.

From our qualitative interviews, two primary themes emerged—**Organization & Access Systems** (10 votes) and **Scanning Tool Requirements** (10 votes). Participants consistently reported frustration with fragmented file storage across multiple devices and platforms, often struggling to locate documents efficiently. In particular, many criticized Windows' search capabilities and expressed a desire for seamless, cross-device syncing (akin to iCloud) and search functionality comparable to macOS Finder. Common organizational strategies——such as event-based folders——frequently led to duplicate files and user confusion. Equally critical were shortcomings in scanning tools. Users described mobile camera scans as blurry, OCR accuracy as unreliable, and software interfaces as obstructed by region restrictions or intrusive advertisements. These issues significantly impede the trustworthy digitization of physical documents.

A closely related theme, **Tool Efficiency & Simplicity** (9 votes), highlighted users' preference for one-click operations and automated workflows—such as automatic syncing of new scans—to minimize manual tasks like retyping. This dovetails with **Document Discovery Challenges** (8 votes), where locating existing files was identified as the most time-consuming activity. In the absence of a systematic organization, users resorted to memory or ad hoc searches across disparate apps. Suggestions for improvement included automated tagging and AI-driven retrieval systems.

A third area of concern, **File Export & Flexibility** (6 votes), revealed pain points in post-scan processing: merging multi-page PDFs was described as tedious, and even minor edits to a compiled document often necessitated restarting the entire workflow. Complex formats, such as bank statements, further strain OCR capabilities. As one participant noted, "Modifying one page in a compiled PDF means redoing everything", underscoring the need for modular editing and more robust text-recognition algorithms.

Notably, **Security & Transparency** (2 votes) and **Hardware Integration** (2 votes) received comparatively little attention. This may indicate that users assume security features are a given or prioritize convenience over safeguards—a potentially risky oversight for sensitive records, such as visas or financial statements. The minimal focus on hardware suggests that existing solutions are either sufficient or that users have accepted current limitations.

### 2.1.4 Customer Profile

Our interviews highlight several core needs of our target customers. First, users demand effortless organization: a single, reliable system that offers instant categorization and unified search, so they never have to remember where a file is stored or sift through irrelevant results.

This unification must encompass both physical and digital documents, ensuring that visas, academic records, receipts, and other critical files are always accessible.

High-quality, ad-free scanning is equally essential. Participants insist on crisp, watermark-free captures and batch processing with built-in deblurring. They need layout-aware OCR that accurately handles printed and handwritten text (eliminating mix-ups like “0” vs. “O”), plus automated data extraction to auto-fill standardized forms from trusted databases, thereby removing time-consuming copy-and-paste work and reducing errors.

Once documents are in the system, users require flexible post-scan editing. They want to merge, reorder, or modify individual pages without having to restart the entire PDF workflow. Modular exports—potentially in formats beyond static PDFs—should support quick updates and annotations for complex, multi-page documents such as bank statements.

Security must be robust yet invisible. Although explicit security votes were few, the underlying fear of cloud leaks drives many to store sensitive files locally. Users will embrace end-to-end encryption and granular access controls if these safeguards operate transparently, without requiring extra steps or performance trade-offs.

Finally, users envision proactive assistance via AI: automated tagging, semantic search by content and context, and a lightweight assistant that summarizes documents and suggests next steps. By shifting from reactive file hunting to proactive workflow guidance, the system can transform document management into a trusted, efficient experience.

**Design Implications** To satisfy these needs, our product, DocuSnap, must

1. Deliver clean, batch-capable scanning with layout-aware, error-resistant OCR and automated form-filling.
2. Unify repositories and automate organization with seamless syncing, global search, and AI-powered tagging.
3. Enable modular, page-level editing of exports, beyond static PDFs, for quick, incremental updates.
4. Embed transparent security through default encryption and fine-grained access controls, including an optional local-first mode.
5. Provide contextual intelligence with semantic search, auto-fill from verified data sources, and an AI assistant for summarization and task suggestions.

## 2.2 Competitor Analysis

### 2.2.1 Manual Photo Organization

Manual photo organization, relying on default smartphone galleries or generic storage, involves users manually capturing, naming, and sorting document photos into self-defined folders, often with inconsistent labels such as "IDs" or timestamped filenames. This approach lacks built-in tools for image enhancement, forcing users to crop and edit images manually, and offers no keyword search, requiring them to scroll through unstructured files. Critical data isn't extracted automatically, sensitive info risks exposure in unencrypted storage, and workflows fragment across devices, making it inefficient, error-prone, and unsuitable for managing growing document volumes.

### 2.2.2 CamScanner

CamScanner positions itself as a dedicated document-scanning tool, offering features like edge detection, perspective correction, and OCR to convert photos of physical documents into polished PDFs[3]. The app enhances scan quality through auto-cropping and filters that adjust contrast or remove shadows, making it popular among users who need reliable digital copies for professional or academic submissions. Its cloud storage integration allows cross-device access, and the OCR feature enables basic text searches within scanned files. However, CamScanner's technical limitations become apparent in complex use cases. The OCR engine struggles with accuracy when processing documents with multi-column layouts, handwritten text, or low-resolution images, often misreading critical details like dates or numbers[10-11]. This unreliability forces users to verify extracted data, undermining the promise of automation. Additionally, while CamScanner supports basic organization through folders, it lacks intelligent categorization—users cannot automatically sort invoices by ID or link related documents, such as passports and visas. Non-technical drawbacks further hinder its appeal. Intrusive ads and watermarks mar the free version, while advanced features like high-resolution OCR and bulk processing require a subscription, alienating cost-sensitive users. Privacy concerns also linger due to past security breaches, including a 2019 incident where malware was discovered in the app, eroding trust in its data-handling practices[12]. These factors relegate CamScanner to a narrow role as a scanner rather than a holistic document management solution.

### 2.2.3 Apple Photos

Apple Photos leverages its native integration with iOS devices to offer a passive approach to document management[5]. Using AI-driven image recognition, the app automatically detects documents like receipts or IDs in users' camera rolls and groups them into broad categories such as "Scans" or "Notes". With the introduction of Live Text in iOS 15, Apple Photos added OCR capabilities, allowing users to search for text within images—for example, typing "passport number" to locate a scanned ID. While this feature simplifies retrieval compared to manual scrolling, Apple Photos remains a photo-first app, treating documents as incidental. Technically, its OCR is shallow; it recognizes text but cannot extract structured data (e.g., pulling expiration dates from a driver's license) or generate actionable insights. The app also lacks document-specific editing tools, leaving users to rely on third-party apps for basic enhancements like glare removal or perspective correction. Organizationally, Apple Photos provides no way to tag documents with custom metadata or create nested folders, forcing users to depend on its rigid, algorithm-driven categorization. Non-technical limitations include platform exclusivity, as the app is unavailable to Android users, and privacy concerns tied to iCloud storage. Sensitive documents stored in Apple Photos are vulnerable if a user's iCloud account is compromised, and the app offers no encryption or password protection for individual files. Ultimately, Apple Photos serves as a convenient but superficial tool for users already embedded in the Apple ecosystem, failing to address the nuanced needs of dedicated document management.

### 2.2.4 Adobe Scan

Adobe Scan, part of Adobe's Document Cloud suite, targets professionals with its high-fidelity scanning and deep integration with PDF workflows[4]. The app excels in producing clean, print-ready scans through advanced image cleanup tools, such as automatic shadow removal and color adjustment, making it a favorite for archiving contracts or academic papers. Its OCR engine is among the most accurate, capable of extracting text from complex layouts and exporting it to editable formats like Word or Excel. However, Adobe Scan's technical sophistication comes at a cost. The app is resource-intensive, draining device batteries and requiring significant processing power, which limits its usability on older smartphones or tablets. Its feature set, tailored for power users, overwhelms casual users seeking simple scans, as the interface buries essential tools behind layers of menus. Organizationally, Adobe Scan relies on manual folder systems within Adobe Document Cloud, offering no AI-driven tagging or smart categorization. Non-technical barriers include its pricing model: critical features like cloud storage beyond basic limits and advanced OCR require a Creative Cloud subscription, which is prohibitively expensive for individuals outside corporate environments. Furthermore, Adobe Scan operates in isolation from non-Adobe ecosystems; scans cannot be seamlessly searched or edited outside Adobe's tools, creating workflow silos. While Adobe Scan is unmatched in scan quality, its complexity, cost, and lack of intuitive organization render it impractical for everyday users.

### 2.2.5 Microsoft Lens

Microsoft Lens (formerly Office Lens) emphasizes integration with Microsoft 365 workflows, positioning itself as a bridge between physical documents and productivity tools like OneNote, Word, and OneDrive[6]. The app optimizes scans for readability, offering modes tailored to whiteboards, documents, or handwritten notes, and automatically straightens or enhances images for improved clarity. Its OCR capabilities embed extracted text into PDFs, enabling users to copy and paste content into Office apps. For Microsoft-centric users, this creates a seamless experience, such as scanning a receipt and directly importing it into an Excel expense report. However, Microsoft Lens suffers from technical limitations that restrict its utility as a standalone document manager. While OCR text is embedded in scans, the app does not index this data for cross-document searches, meaning users cannot search for "insurance policy number across all stored files. The lack of tagging or folder systems beyond OneDrive's basic organization forces users to rely on external tools for categorization and organization. Non-technically, Microsoft Lens's value is heavily contingent on the adoption of Microsoft's ecosystem. Users outside the Office 365 suite gain little from its features, and the app offers no incentives for those preferring Google Workspace or other platforms. Even for Microsoft users, the app's focus on scanning-to-Office workflows neglects broader document management needs, such as secure storage or form-field automation. While its free, ad-free model is appealing, Microsoft Lens remains a niche tool for Office power users rather than a comprehensive solution.

**Comparison of Competitors** CamScanner, Apple Photos, Adobe Scan, and Microsoft Lens each address fragments of the document management challenge but fall short of a unified solution. CamScanner prioritizes scan quality but lacks intelligent organization; Apple Photos offers passive convenience but no document-specific tools; Adobe Scan caters to professionals at the expense of accessibility; and Microsoft Lens ties utility to ecosystem loyalty. Common gaps include the inability to auto-categorize diverse document types, extract and structure data for form-filling, and strike a balance between advanced features and affordability. These shortcomings underscore the need for a platform that integrates robust scanning, AI-driven organization, and cross-functional automation—all while maintaining accessibility for everyday users.

**Table 2–1 Comparison of Key Features of Competitors and our Product**

| **Feature** | **Manual Org** | **Apple Photos** | **Cam Scanner** | **Adobe Scan** | **Microsoft Lens** | **DocuSnap (Ours)** |
|---|---|---|---|---|---|---|
| OCR Accuracy | None | Low | Medium | High | Medium | High |
| Document Enhancement | None | None | High | High | Medium | High |
| Cross-Platform Search | No | Limited | Yes | No | No | Yes |
| Form-Field Auto-Filling | No | No | No | No | No | Yes |
| Cost | Free | Free | Premium | Premium | Free | Free |
| Privacy | Low | Medium | Low | High | Medium | High |

# Chapter 3 App Design

## 3.1 Storymap and Features

Our storymap is illustrated below (Illustration 3–1). The features are grouped by user experience phase and development stage.

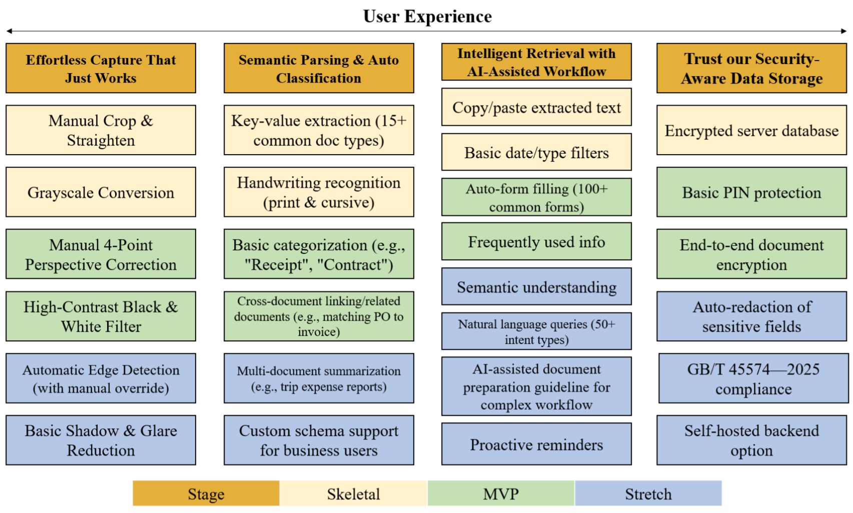

User Experience
Effortless Capture That Just Works
Semantic Parsing & Auto Classification
Intelligent Retrieval with AI-Assisted Workflow
Trust our Security-Aware Data Storage
Manual Crop & Straighten
Key-value extraction (15+ common doc types)
Copy/paste extracted text
Encrypted server database
Basic date/type filters
Grayscale Conversion
Handwriting recognition (print & cursive)
Auto-form filling (100+ common forms)
Basic PIN protection
Manual 4-Point Perspective Correction
Basic categorization (e.g., "Receipt", "Contract")
Frequently used info
End-to-end document encryption
Semantic understanding
High-Contrast Black & White Filter
Cross-document linking/related documents (e.g., matching PO to invoice)
Natural language queries (50+ intent types)
Auto-redaction of sensitive fields
Automatic Edge Detection (with manual override)
Multi-document summarization (e.g., trip expense reports)
AI-assisted document preparation guideline for complex workflow
GB/T 45574—2025 compliance
Basic Shadow & Glare Reduction
Custom schema support for business users
Proactive reminders
Self-hosted backend option
Stage
Skeletal
MVP
Stretch

**Illustration 3–1 Story Map**

## 3.2 Acceptance Criteria for Features

This section outlines the specific acceptance criteria for each feature.

### 3.2.1 Stage 1 Features

#### 3.2.1.1 Manual Crop & Straighten

**WHEN** A user engages the “Crop & Straighten” tool for a document.

**THEN** The system provides an interactive interface for precise manual boundary adjustments and rotation, displaying the updated document instantly with sub-second response times, and ensuring all subsequent document functionalities apply to the newly processed version.

#### 3.2.1.2 Grayscale Conversion

**WHEN** A document is captured via the app's scanner.

**THEN** The image is automatically converted to grayscale to create a cleaner visual appearance, improve text clarity for OCR processing, preserve all critical document details without data loss, and save the processed version in the user's document library.

#### 3.2.1.3 Manual 4-Point Perspective Correction

**WHEN** A user provides four non-collinear corner points defining a document in an image.

**THEN** The system generates a geometrically corrected rectangular output where edges appear straight, text aligns horizontally, all critical content remains uncropped, and the perspective distortion is fully resolved without requiring additional user input.

#### 3.2.1.4 High-Contrast Black & White Filter

**WHEN** A user applies the filter to a color or grayscale document image.

**THEN** The system outputs a pure black-and-white version where text/features retain full legibility, background artifacts are removed, contrast is maximized between foreground and background elements, and the output resolution matches the original without loss of critical detail.

#### 3.2.1.5 Automatic Edge Detection (with manual override)

**WHEN** The system auto-detects document edges in an image.

**THEN** It outputs a perspective-corrected document or allows one-click manual corner override with identical output quality.

#### 3.2.1.6 Basic Shadow & Glare Reduction

**WHEN** Applied to an image.

**THEN** Shadows/glare are reduced without degrading text legibility or resolution, while preserving original document proportions and critical details.

### 3.2.2 Stage 2 Features

#### 3.2.2.1 Key-value extraction (15+ common doc types)

**WHEN** A user uploads a document from one of the 15+ supported common document types.

**THEN** The system automatically extracts key-value pairs, displaying them instantly with high accuracy and sub-second response times, and presenting them in an easily reviewable and editable format.

#### 3.2.2.2 Handwriting recognition (print & cursive)

**WHEN** A user uploads a document containing handwritten text (both print and cursive).

**THEN** The system accurately transcribes the handwriting into editable digital text, displaying the recognized content instantly and enabling seamless search and selection functionality within the transcribed areas.

#### 3.2.2.3 Basic categorization (e.g., “Receipt”, “Contract”)

**WHEN** A user uploads or scans a document.

**THEN** The system shall automatically assign a high-level category (e.g., “Receipt”, “Contract”, “Invoice”) based on document layout, content features, and OCR text.

**AND** The categorization shall be completed within 2 seconds for documents under 10 pages.

**AND** The confidence score for top prediction must exceed 85% for the result to be applied without fallback to user correction.

#### 3.2.2.4 Cross-document linking/related documents

**WHEN** Two or more documents are uploaded that contain semantically or structurally related identifiers (e.g., matching Purchase Order numbers, vendor names, or invoice references).

**THEN** The system shall surface linkages between them automatically in the UI.

**AND** Group or recommend the documents for joint review or action.

**AND** Allow the user to inspect or override the linkage with a justification pane.

#### 3.2.2.5 Multi-document summarization (e.g., trip expense reports)

**WHEN** A user selects multiple related documents (e.g., receipts from a business trip).

**THEN** The system shall generate a natural language summary or structured table capturing key fields (e.g., date, amount, vendor) across documents.

**AND** The output must be editable by the user before export.

**AND** Any OCR errors or missing fields shall be highlighted for manual verification.

#### 3.2.2.6 Custom schema support for business users

**WHEN** An admin user defines a custom schema (e.g., a new document type with specific fields like “Project Code” or “Client ID”).

**THEN** The system shall support parsing and storing documents against this schema.

**AND** Provide form-based field mapping UI to guide training for each custom field.

**AND** Auto-populated values must be validated against schema rules (e.g., regex format, mandatory fields) before saving.

### 3.2.3 Stage 3 Features

#### 3.2.3.1 Copy/paste extracted text

**WHEN** A user selects extracted text (e.g., an invoice number, contract clause, or receipt total) from a parsed document in DocuSnap.

**THEN** The system highlights the selected text and enables standard OS copy/paste functionality (Ctrl+C/Ctrl+V or long-press menus on mobile). The copied text retains its original formatting (e.g., numbers, dates, or key-value pairs like “Total: $50.00”) without corruption. The pasted content matches the extracted text exactly (no truncation or added metadata). On mobile, the paste action works seamlessly in external apps (e.g., Notes, email, or forms).

#### 3.2.3.2 Basic date/type filters

**WHEN** A user applies a single filter, such as by document type (e.g., type=“Receipt”) or by date (e.g., date=“March 2024”).

**THEN** The system displays only documents matching the applied criterion, provides a clear visual indicator of the active filter, and shows a user-friendly empty state message when no matches exist, while maintaining full scroll and selection functionality in the filtered results.

#### 3.2.3.3 Auto-form filling

**WHEN** A user taps auto-filling buttons in detail page of any of the 100+ supported common forms (tax forms, insurance claims, employment applications, loan applications, etc.).

**THEN** The system automatically populates applicable fields by extracting verified information from the user's existing document database, displays auto-filled fields with visual indicators to distinguish them from manual entries, allows users to review and modify suggested values, and completes the entire auto-fill process.

#### 3.2.3.4 Frequently used info

**WHEN** The user navigates to their statistics summary.

**THEN** The system must display the name of the input field they have completed most frequently across all sessions.

#### 3.2.3.5 Semantic understanding

**WHEN** A user uploads related documents or performs document searches.

**THEN** The system automatically identifies and establishes contextual relationships between documents (e.g., linking visas with corresponding flight itineraries and hotel reservations), groups multi-page documents as single entities, understands document hierarchies and dependencies, creates intelligent folder structures based on semantic content rather than just file names, and enables cross-document reference searches that surface related information across the entire document collection with 90% accuracy in relationship detection.

#### 3.2.3.6 Natural language queries (50+ intent types)

**WHEN** A user enters a natural language query such as "Q2 client expenses over $75", "show me my passport from last year", or "find contracts expiring this month" in the search bar.

**THEN** The system accurately interprets the user's intent from 50+ supported query types (temporal filters, monetary ranges, document types, status queries, relationship searches), cross-references structured metadata from processed documents to identify matching results, returns relevant documents and specific sections within 3 seconds, and displays results with contextual highlighting showing why each document matched the query parameters.

#### 3.2.3.7 AI-assisted document preparation guideline for complex workflow

**WHEN** The user triggers the "Auto Fill" feature.

**THEN** The system identifies related files and proposes related fills needed.

#### 3.2.3.8 Proactive reminders

**WHEN** The system detects an item's "Expiration Date" is, for example, 7 days away.

**THEN** A reminder notification is automatically sent to the item's owner. The notification clearly states the item will expire in 7 days and provides a direct link to it.

### 3.2.4 Stage 4 Features

#### 3.2.4.1 Encrypted local metadata search

**WHEN** A user performs a search within DocuSnap.

**THEN** The app retrieves relevant document metadata results using only locally stored, encrypted indexes, ensuring no metadata is transmitted or exposed to the cloud.

#### 3.2.4.2 Basic PIN protection

**WHEN** A user attempts to open it.

**THEN** The document's content is hidden, and the system prompts the user for a PIN. When the user provides the correct PIN, the document's content becomes fully visible and accessible.

#### 3.2.4.3 End-to-end document encryption

**WHEN** The user uploads a document or a form.

**THEN** The document's content is encrypted on the client-side before being sent over the network. The data stored in the database for that document is in its encrypted format.

#### 3.2.4.4 Auto-redaction of sensitive fields

**WHEN** A user submits data containing common PII patterns (e.g., Social Security Numbers, credit card numbers, phone numbers).

**THEN** The system automatically replaces the matched text with a redaction mask (e.g., ***-**-1234) before it is saved.

#### 3.2.4.5 GB/T 45574—2025 compliance

**WHEN** A user or process accesses data governed by the standard.

**THEN** The system must generate an immutable audit log entry detailing the user ID, timestamp, and the action performed.

#### 3.2.4.6 Self-hosted backend option

**WHEN** A user provides a valid URL and credentials for their backend server.
**THEN** All subsequent data processing must be exclusively directed to that server.

## 3.3 Engine Architecture

### 3.3.1 Model and Engine

Table 3–1 outlines the primary engine components, detailing their respective functionalities and implementations. Device components (1-5) use Android OS capabilities, backend services (6-9) run on our infrastructure, and Zhipu LLM (10) is an external dependency.

**Table 3–1 Engine Components and Implementation Details**

| Component | Functionality | Implementation |
|---|---|---|
| User Frontend | UI rendering and interaction. | Android Studio (API 33); Build from scratch. |
| Camera/Gallery | Image capture/selection. | Android Studio (API 33) and Gallery APIs. |
| Geo/Color Processor | Image correction/enhancement. | Android Studio (API 33); Build from scratch. |
| Document/Form Handler | Workflow coordination. | Android Studio (API 33); Build from scratch. |
| Frontend DB | Local data persistence. | SQLite via Android Room. |
| Backend Server | API routing. | Flask + Gunicorn. |
| Backend Worker | Async processing. | Python threading. |
| Cache Server | Temporary data storage. | Flask + Gunicorn + SQLite. |
| OCR Server | Text extraction. | Flask + Gunicorn + CnOcr library. |
| Zhipu LLM (External Service) | Data enrichment. | External API integration. |

### 3.3.2 Data and Control Flow Diagram

We present the entity relationship in our app through two block diagrams shown below. Specifically, Illustration 3–2 illustrates the data processing flow on the frontend, while Illustration 3–3 describes the backend service architecture of the system.

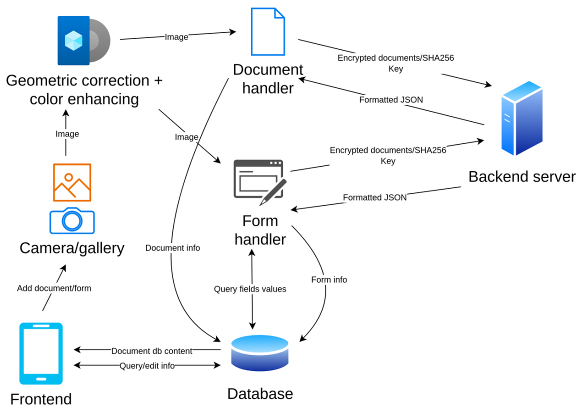


**Illustration 3–2 Block Diagram 1**

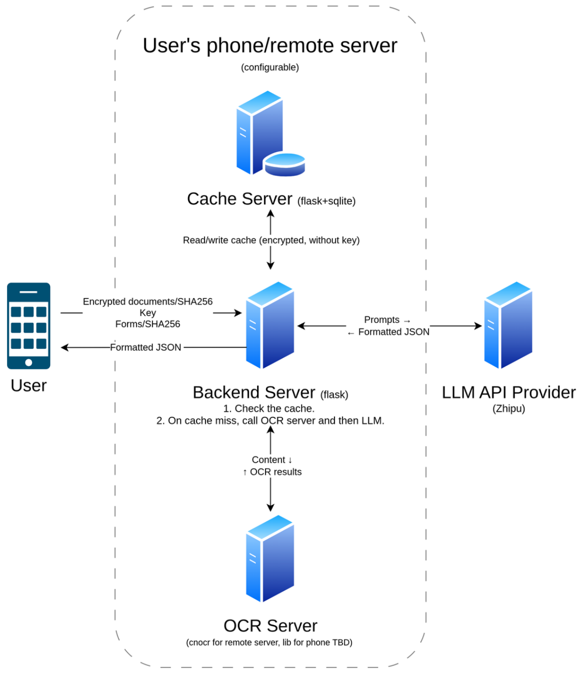

User's phone/remote server
(configurable)
Cache Server (flask+sqlite)
Read/write cache (encrypted, without key)
User
Encrypted documents/SHA256
Key
Forms/SHA256
Formatted JSON
Prompts →
← Formatted JSON
Backend Server (flask)
1. Check the cache.
2. On cache miss, call OCR server and then LLM.
LLM API Provider
(Zhipu)
Content ↓
↑ OCR results
OCR Server
(cnocr for remote server, lib for phone TBD)

**Illustration 3–3 Block Diagram 2**

## 3.4 API Design

The system architecture implements a contract-first interface design with formally specified input-output behaviors. Frontend modules expose typed function signatures for document processing workflows, while backend services provide RESTful endpoints with cryptographic payload handling. Complete interface specifications appear in Appendix 4, defining method signatures and functional behaviors without implementation details.

### 3.4.1 Modular Frontend Architecture

Client-side processing follows a pipeline architecture beginning with image acquisition through `captureImage(source)` which abstracts device-specific media access. Subsequent transformations apply geometric correction via `correctGeometry(image)` and color optimization through `enhanceColors(image)`. Document processing diverges at the `processDocument()` and `processForm()` handlers that output encrypted content with cryptographic hashes. Local state management is handled through database operations like `saveDocument()` and `getFormData()` as formally specified in Appendix 4.1.

### 3.4.2 Secure Backend Services

The backend implements a unified processing endpoint `/api/process` accepting cryptographically verified payloads. Requests require SHA-256 integrity hashes and dual-layer encryption (AES-256 content with RSA-wrapped keys). As documented in the Appendix 4.2, the endpoint handles three processing types through polymorphic responses: document metadata extraction, form structure parsing, and field-value mapping for pre-filled forms. The system employs HTTP status codes (200, 202, 400, 500) to represent processing states, with response schemas conforming to formally defined JSON structures.

### 3.4.3 Supporting Infrastructure

A dedicated cache server provides content-addressable storage through `/api/cache/store` and retrieval via `/api/cache/query` endpoints using composite keys (client ID, content hash, processing type). The OCR subsystem exposes a single `/extract` endpoint for text recognition. Third-party integration utilizes Zhipu AI's language models through their published SDKs. All interface contracts are enumerated in Appendix 4.3 and 4.4.

### 3.4.4 Security Architecture

The API design enforces cryptographic chaining throughout the processing lifecycle. Frontend modules generate SHA-256 hashes and RSA-encrypted outputs as per their signatures. Backend services validate payload integrity before processing and maintain encryption-at-rest for cached results. The security model requires client authentication via UUIDs and processes sensitive operations exclusively through encrypted channels following the protocols defined in the Appendix and 4.2.

## 3.5 UI/UX Design

### 3.5.1 Overall UI/UX Flow Architecture

The DocuSnap UI/UX flow is architected around four core stages that mirror the user's natural document management journey: Capture, Process, Organize, and Retrieve (Illustration 3–4). This flow design prioritizes seamless transitions between stages while maintaining user context and minimizing cognitive load. Each stage builds upon the previous one, creating a compound value experience where users progressively unlock more sophisticated document management capabilities.

### 3.5.2 Home Page: Central Navigation Hub

The home page (Illustration 3–5) serves as the primary entry point and navigation hub, designed with a clean, intuitive interface that immediately communicates DocuSnap's core value propositions. The layout follows a card-based design system that allows users to quickly understand available actions without overwhelming them with options.

#### User Flow Details

Upon opening the app, users encounter a carefully structured home screen layout optimized for both accessibility and functionality:

**Top Section - Universal Search Bar:** At the very top of the screen, a prominent search bar enables users to perform natural language queries across their entire document library. This persistent search functionality ensures users can quickly access any document from the home screen without navigating through multiple menus.

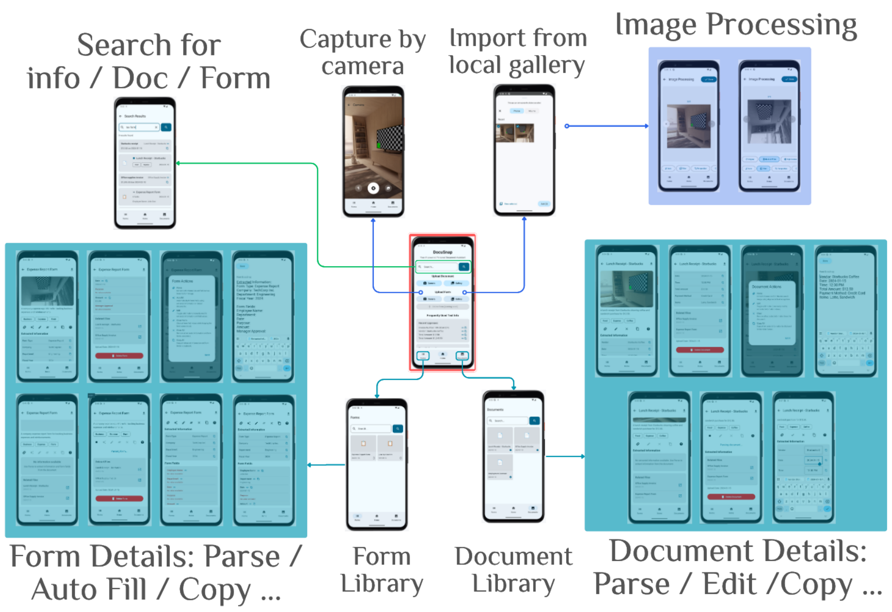

Search for info / Doc / Form
Capture by camera
Import from local gallery
Image Processing
Form Details: Parse / Auto Fill / Copy ...
Form Library
Document Library
Document Details: Parse / Edit /Copy ...

**Illustration 3–4 Overall UI/UX Flow Architecture**

**Main Content Area - Upload Entry Points:** The central portion of the screen features two primary upload mechanisms:

1. **Document Upload Card:** A large, visually prominent card with camera and gallery icons, allowing users to capture new documents or import existing images from their device gallery
2. **Form Upload Card:** A secondary card specifically designed for form processing, with distinct visual styling to differentiate it from standard document capture

**Mid-Section - Frequently Used Information:** Below the upload cards, the interface displays a dynamic ”Frequently Used” section that shows:

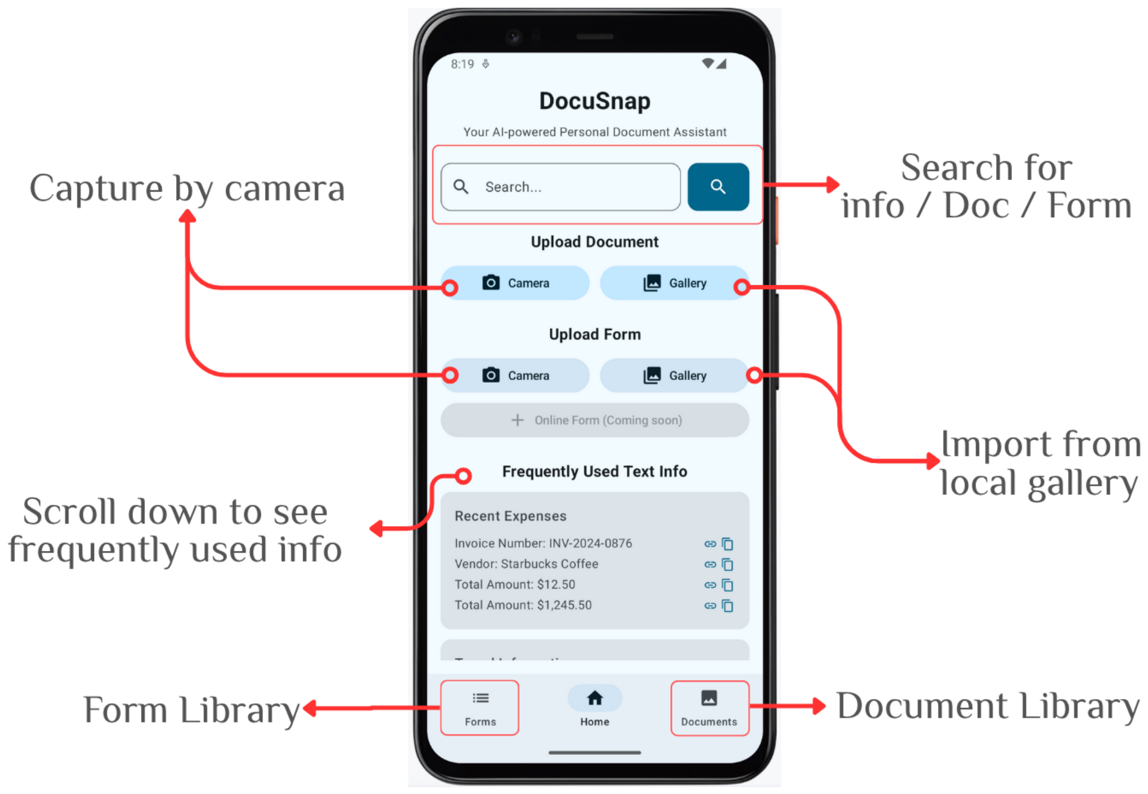
- Quick access to commonly referenced information (e.g., ID numbers, insurance details)
- Smart suggestions based on user patterns and upcoming deadlines

**Bottom Navigation - Global Library Access:** A persistent bottom navigation bar provides one-tap access to the app’s core organizational features:

- **Documents Library Tab:** Direct access to all processed documents with filtering and sorting capabilities
- **Forms Library Tab:** Dedicated section for forms and form templates

- **Settings/Profile Tab:** Account management and app configuration options

This layout ensures that the most frequently used functions (search and capture) are immediately accessible, while providing clear pathways to organized content through the bottom navigation.


8:19
DocuSnap
Your AI-powered Personal Document Assistant
Search...
Upload Document
Camera
Gallery
Upload Form
Camera
Gallery
Online Form (Coming soon)
Frequently Used Text Info
Recent Expenses
Forms
Home
Documents
Capture by camera
Search for info / Doc / Form
Import from local gallery
Scroll down to see frequently used info
Form Library
Document Library

Invoice Number: INV-2024-0876
Vendor: Starbucks Coffee
Total Amount: $12.50
Total Amount: $1,245.50

**Illustration 3–5 UI/UX Layout**

### 3.5.3 Search Stage: AI-Powered Document Discovery

#### 3.5.3.1 Natural Language Query Interface

The search stage revolutionizes document retrieval by accepting natural language queries rather than requiring exact filename matches. Users can type conversational queries like ”show me tax documents from last year” or ”find medical bills over $100” and receive contextually relevant results.

#### 3.5.3.2 Search Results Architecture

The search results (Illustration 3–6) are presented in a three-tier information hierarchy:

1. **Primary Results:** Documents that exactly match the search criteria are displayed with thumbnail previews and key extracted information
2. **Related Documents:** The AI identifies semantically related documents that might be relevant to the user's query
3. **Extracted Information:** Specific text snippets or data points from documents that contain the queried information

#### 3.5.3.3 Interactive Result Elements

Each search result includes:

- **Document Preview:** A clean thumbnail image of the document
- **Key Metadata:** Extracted information like document type, date, and relevant amounts
- **Quick Actions:** One-tap options to copy key-value pairs, see full document or view related document

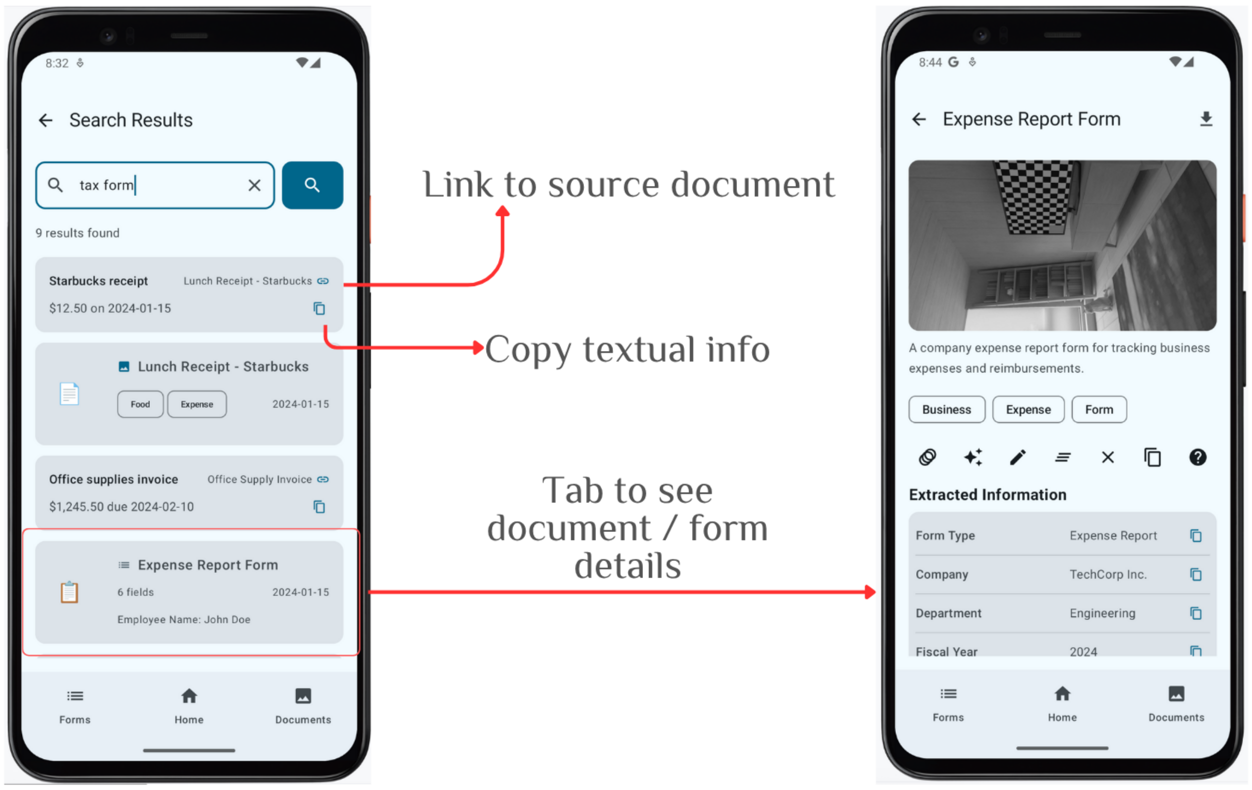
### 3.5.4 Image Processing: One-Tap Document Enhancement

The image processing stage (Illustration 3–7) supports both gallery import and real-time camera capture, recognizing that users need flexibility in how they digitize documents. The interface adapts based on the input method while maintaining consistent processing capabilities.

#### 3.5.4.1 One-Tap Auto-Processing

Following the "one-tap auto-processing" principle from our story map, the system automatically applies multiple enhancement techniques:

1. **Edge Detection:** Automatically identifies document boundaries even in cluttered environments
2. **Perspective Correction:** Straightens angled photos using advanced geometric transformation
3. **Lighting Optimization:** Removes shadows and glare while enhancing text clarity
4. **Noise Reduction:** Cleans up grain and artifacts from low-light photography



8:32

Search Results

tax form

9 results found

Starbucks receipt — Lunch Receipt - Starbucks
$12.50 on 2024-01-15

Lunch Receipt - Starbucks
Food Expense 2024-01-15

Office supplies invoice — Office Supply Invoice
$1,245.50 due 2024-02-10

Expense Report Form
6 fields 2024-01-15
Employee Name: John Doe

Forms Home Documents

8:44 G

Expense Report Form

A company expense report form for tracking business expenses and reimbursements.

Business Expense Form

Extracted Information

| | |
|---|---|
| Form Type | Expense Report |
| Company | TechCorp Inc. |
| Department | Engineering |
| Fiscal Year | 2024 |

Forms Home Documents

**Illustration 3–6 Interactive Result**

#### 3.5.4.2 Manual Editing Interface

For users who want fine-tuned control, the interface provides intuitive manual editing tools:

- Crop Handles: Visual markers that allow precise boundary adjustment
- Rotation Controls: Smooth rotation with snap-to-angle functionality
- Filter Options: One-tap filters included adjustable color enhancement, high contrast and black white threshold filter.
- Undo/Redo: Full history tracking for confident experimentation.

### 3.5.5 Document Handler: Intelligent Organization and Management

The **Document Handler** stage transforms raw images into structured, actionable records and provides a highly intuitive interface for managing them. Users can access the document

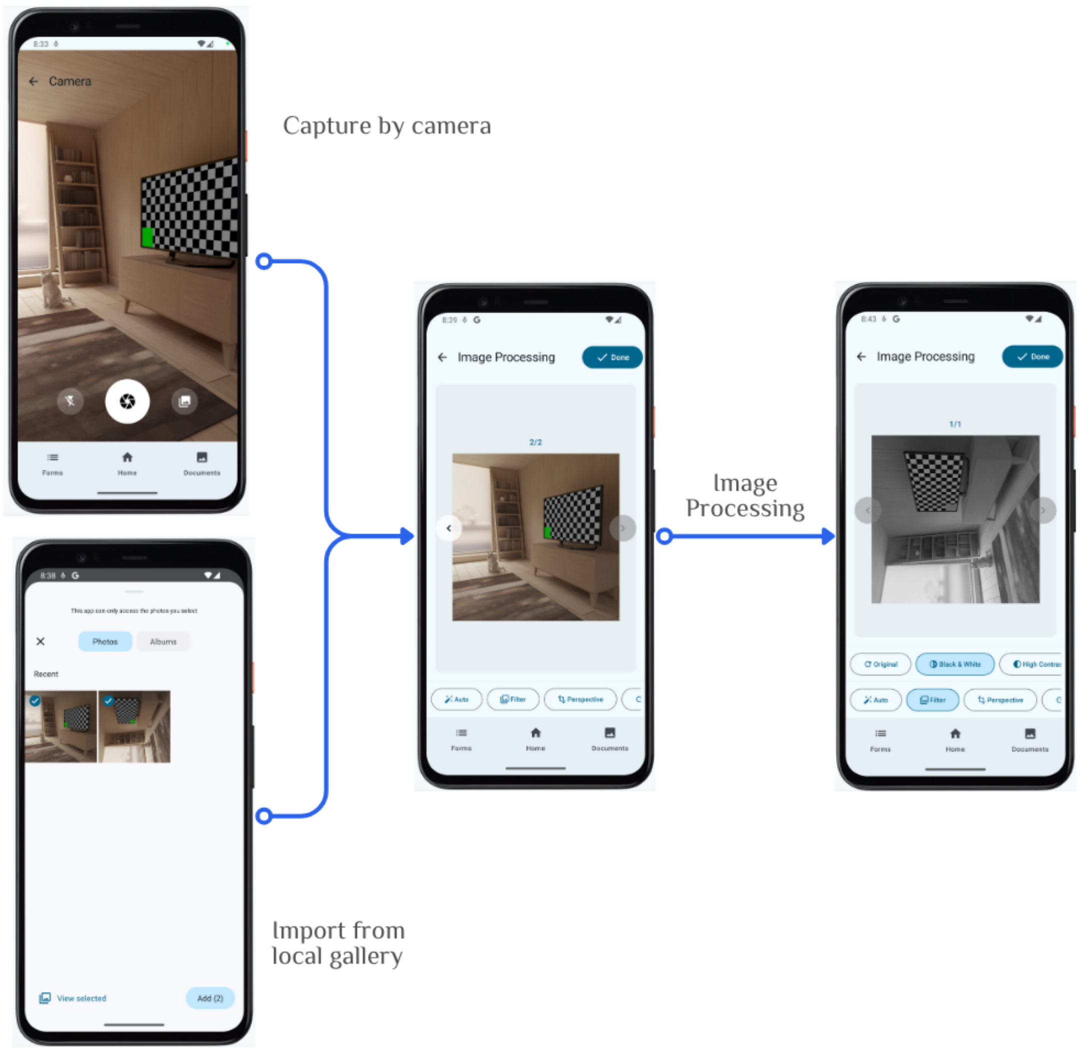


**Illustration 3–7 Manual Editing Interface**

gallery directly through the persistent **“Documents” tab** in the global navigation bar. Each item is represented by a preview card showing a document thumbnail, title, and date.

#### 3.5.5.1 Document Detail View

Tapping a document preview leads to the **Document Detail Page**, which presents:

- **Full-size images** of the document
- **Auto-generated title** and **semantic tags** (e.g., ”Expense,” ”Food”) based on extracted content
- A concise **summary description**
- A clearly structured **Extracted Information** section containing editable key-value pairs such as date, time, total amount, and vendor
- **Linked related files**, visually distinguished and tappable for quick navigation (e.g., related receipts or forms)

These elements together allow users to interact with their documents as structured knowledge, not just static images.

#### 3.5.5.2 Action Panel

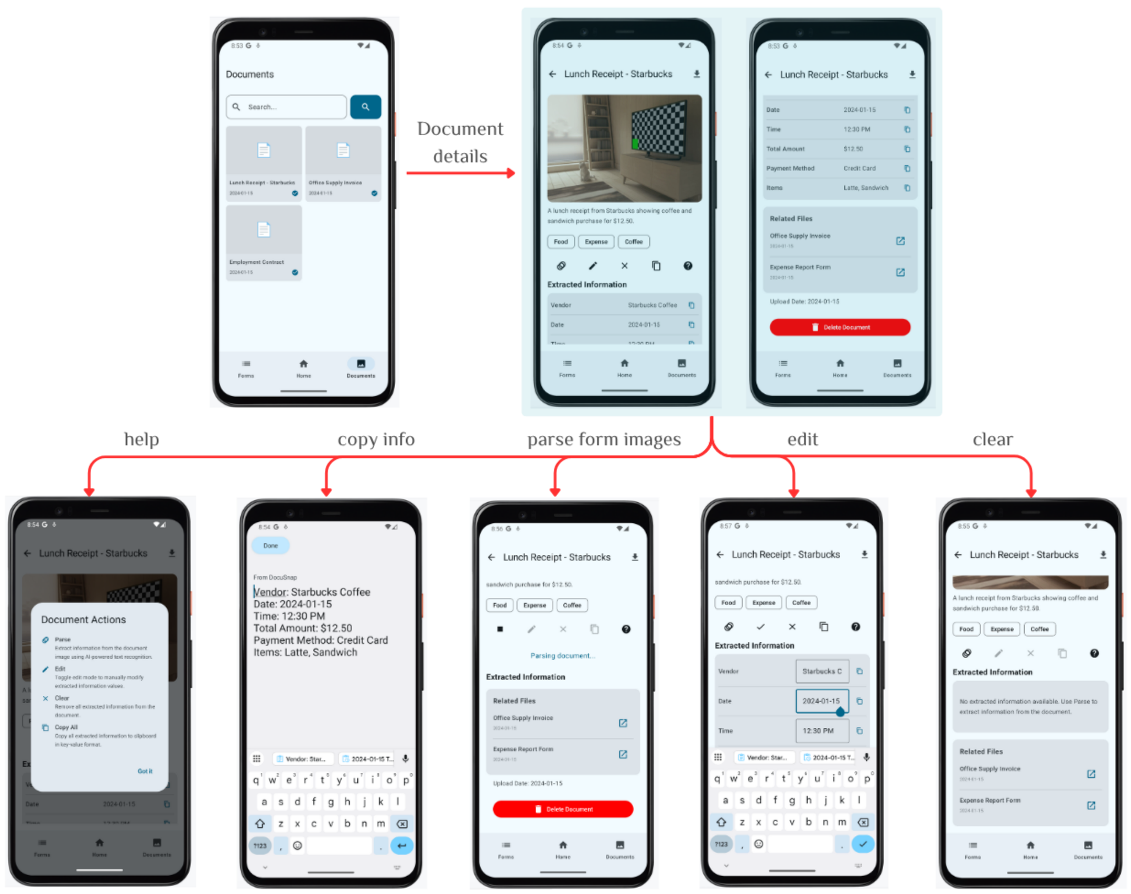
At the top-right corner, users can find the export icon, which enables them to download the document to local storage. At the bottom of the detail page, users are presented with the critical deletion button, prominently placed in red for visibility, it removes the document and all associated metadata. Each document supports a rich set of operations shown clearly beneath the extracted information section:

- **Help:** Triggers a toolbar-style overlay that introduces available actions, reducing first-time user friction.
- **Parse:** Triggers image-to-text parsing and metadata extraction, updating the extracted fields in real-time
- **Edit:** Allows users to manually modify any field with inline input and keyboard support
- **Clear:** Removes all parsed data, which is especially useful for privacy-sensitive documents
- **Copy:** Copies all textual fields in a well-formatted layout to the clipboard for external use

Each action is associated with a clearly labeled icon, keeping the interface minimal yet expressive.

#### 3.5.5.3 Intelligent Categorization

DocuSnap automatically generates **semantic titles and category labels** using LLM-based summarization and classification. This enables clustering of similar documents and facilitates downstream operations such as form autofill, advanced search, and timeline organization. (Illustration 3–8)



From DocuSnap
Vendor: Starbucks Coffee
Date: 2024-01-15
Time: 12:30 PM
Total Amount: $12.50
Payment Method: Credit Card
Items: Latte, Sandwich

**Illustration 3–8 Document Action Panel**

#### 3.5.5.4 Form Handler: Smart Form Extraction and Auto-Fill

The **Form Handler** (Illustration 3–9) module enables users to manage structured forms with enhanced workflow automation. Just like documents, forms can be accessed through the dedicated **"Forms" tab** in the global navigation bar. Each form is represented by a visually distinct card with a title and date stamp for fast identification. A small icon floating on the bottom-right side of the preview card indicates whether the form is filled or not.

#### 3.5.5.5 Form Detail View

Tapping a form preview opens the **Form Detail Page**, which mirrors the document detail layout but is adapted to handle structured form fields. Beside those presented in the document details, this page distinguishes between the textual content into two categories:

- **Extracted Information** –structured key-value pairs with value available from the form (e.g., "Company," "Department," "Fiscal Year" )
- **Form Fields** –user-fillable entries required by the form (e.g., "Amount," "Purpose," "Manager Approval" )

Any missing information is clearly flagged in red text ( "No value available" ), guiding the user to complete these fields. This smart visual cueing ensures forms are never submitted incomplete or with overlooked sections.

#### 3.5.5.6 Action Panel and Smart Operations

Besides functionalities available in the document detail page, this page provides more actions tailored for form operation including

- **Auto-Fill:** Smartly retrieves data from the app's internal document database to fill in required form fields (e.g., employee name from contracts, date from receipts). Auto-filled entries are shown in blue with editable input fields.
- **Clear Form Fields:** Wipes all filled form fields, in case our users want to restart the form filling workflow.

### 3.5.6 Encryption: Security-Aware Data Storage

We implement a privacy-first architecture with robust, customizable encryption features to ensure secure handling of all user data.

All documents are protected through end-to-end encryption, combining RSA and AES schemes to safeguard both data in transit and at rest. Sensitive content is automatically masked before login, and users must complete PIN-based verification before viewing or editing any protected information. The PIN system enforces secure access at the application level, ensuring that even if the device is compromised, user data remains inaccessible without authentication.

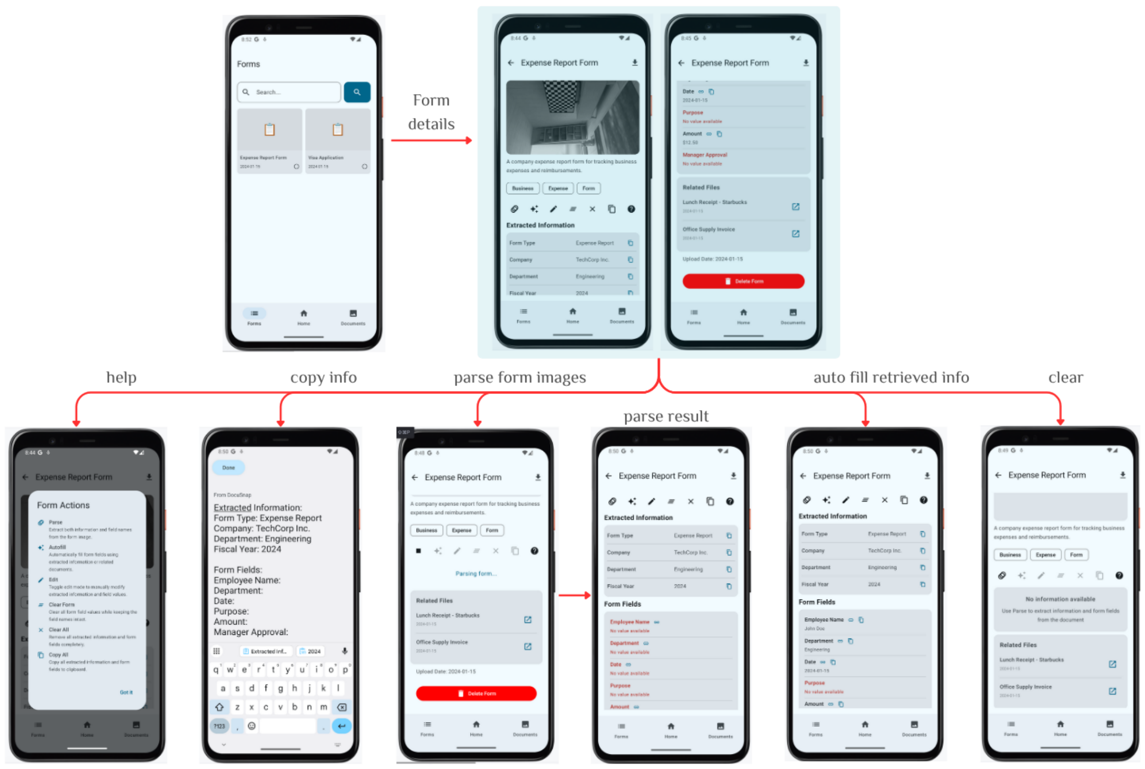


**Illustration 3–9 Form Action Panel**

Users can configure custom backend server endpoints and public keys directly within the app's Settings page. This allows organizations or privacy-conscious individuals to deploy their own self-hosted backend for maximum control over data flow and encryption keys. The built-in configuration dialog provides clear instructions and ensures compatibility with our open-source backend infrastructure.

## 3.6 Usability Testing

To evaluate the real-world usability of our UI mockup, we conducted a structured user testing session based on the interaction prototype built with Loveable[13]. The full interactive design can be accessed via docusnap-mock-ui.lovable.app, and the underlying implementation is available on GitHub at JI-DeepSleep/DocuSnap-Mock-UIUX. A full screen-recorded demo of a user walkthrough can be found in our provided video file user testing.mp4.

The usability test was conducted following the predefined tasks and evaluation metrics described in our testing script detailed in Appendix 5. Each participant interacted with the mock UI across five representative tasks:

1. Import and Enhance a Document
2. Extract Key Information
3. Search Using Natural Language
4. Auto-Fill a Form
5. View and Verify Security Features

Users were encouraged to "think aloud" throughout the session. We also observed task completion time, number of clicks, and interaction errors. After completing all tasks, participants filled out a short post-test questionnaire rating ease-of-use and feature clarity. Each task was evaluated quantitatively against success metrics (e.g., number of taps, response time), while qualitative observations helped uncover issues in UI design, interaction flow, and clarity of function.

Our participants included both regular mobile users and individuals familiar with document management apps. Testing was conducted remotely using video conferencing tools, with screen-sharing enabled.

### 3.6.1 Summary of Findings in Usability Test

The usability testing revealed critical insights into the strengths and areas for improvement in the DocuSnap interface. Qualitatively, users praised the intuitive layout, cohesive color scheme, and effective iconography, particularly on the home page, which contributed to a positive first impression. The natural language search functionality stood out, achieving a 100% success rate in testing, as users found it seamless and efficient. Additionally, the document PIN verification step was well-received, with participants acknowledging its

**Table 3–2 Test result of DocuSnap Mock UIUX Usability Test**

| Tasks | Evaluation Metrics | Success Rate % |
|---|---|---|
| Import a photo and enhance it | 2 clicks to import + n clicks to enhance it | 100% |
| Get the extracted info of a doc | 2 clicks to the doc/the PIN code page | 80% |
| Search for a doc/form | 1 click to the search bar | 100% |
| Auto fill a form | 3 clicks to the results | 60% |
| View and Verify Security Features | Users are prompted to enter PIN code | 100% |

clarity and value as a privacy measure.

However, several pain points emerged, particularly in visual design and workflow efficiency. Users noted that the image enhancement interface appeared cluttered, with excessive text and poor visual hierarchy, hindering quick comprehension. Extracted information was described as overly plain, lacking emphasis or scannability. Workflow disruptions were also evident: after enhancing an image, users were unexpectedly returned to the home screen instead of the detail page, breaking task continuity. Similarly, search results did not directly link to their corresponding document or form detail pages, causing confusion. The absence of an auto-fill entry point on form detail pages further compounded usability issues, leaving users uncertain about next steps. Functionally, participants expressed a desire for greater control over parsing, clearer microcopy for actions like ”parse” or ”edit,” and a dedicated Documents/Forms tab for easier navigation. Traceability concerns were also raised, with users requesting visibility into the source of auto-filled data for verification.

Quantitatively, task completion rates varied significantly as shown in Table 3–2. While photo import and enhancement (100% success), document search (100%), and security PIN prompts (100%) performed flawlessly, extracting document information (80%) and auto-filling forms (60%) saw lower success rates. These metrics highlight a disparity between foundational functionalities and more complex interactions, particularly those involving multi-step processes like form handling.

In summary, while the core UI/UX framework demonstrates strong usability, refinements in visual hierarchy, task flow continuity, and feature discoverability—especially for form-related tasks—are essential to elevate user efficiency and satisfaction. Addressing these gaps will ensure a more cohesive and intuitive experience.

### 3.6.2 Change to Final UI/UX Design

Based on insights from our usability testing, we implemented a series of targeted changes to enhance the overall interaction experience, visual clarity, and workflow continuity of DocuSnap.

To address user concerns around content discoverability, we introduced a persistent tab in the bottom navigation bar for direct access to both “Documents” and “Forms.” Previously, users had to navigate through intermediate steps to locate imported items, which felt buried and unintuitive. This change aligns with industry-standard layouts and significantly improves the visibility of stored content, allowing users to locate their materials with a single tap from any screen.

In response to feedback that the image processing interface was visually dense and overwhelming, we streamlined the layout by replacing verbose text with intuitive icons and concise microcopy. This redesign reduced visual clutter, enhanced immediate recognition of each function, and made the enhancement tools feel lighter and more approachable. The revised design not only accelerated user decision-making but also improved aesthetic consistency across screens.

One of the most disruptive issues identified during testing was the abrupt transition back to the home screen after a document or form was processed. Users found this break in flow disorienting. To improve continuity, we now directly navigate users to the corresponding detail view after processing. This keeps them in context, allowing them to immediately verify extracted content, edit fields, or export results without having to manually find the processed item again.

Another key usability gap involved the search functionality. Although users could successfully retrieve documents using natural language, they reported frustration when clicking a result led nowhere. We addressed this by linking each search result directly to its associated detail page, removing friction from the search-to-action workflow and eliminating ambiguity around where results would take the user.

Form handling also received critical functional upgrades. Originally, users could not complete forms efficiently due to the absence of a clear auto-fill mechanism. In the updated design, we embedded a “Fill Form” button directly within the form detail page. When tapped, it retrieves relevant values from the document database and populates applicable fields. This feature drastically reduces manual input and has improved task success rates in follow-up testing.

Lastly, to increase transparency and build user trust in automated filling, we added a source-tracing mechanism for each auto-filled entry. Users can now view which document provided a given value by tapping a small info icon next to the field. This addition improves auditability, allows for quick corrections if needed, and reinforces user confidence in the app's intelligence.

Collectively, these refinements represent a user-centered evolution of the UI, making DocuSnap not only more functional, but also more predictable, responsive, and aligned with how users expect a modern productivity tool to behave.

# Chapter 4 App Development and Testing

## 4.1 Front-end Development

The DocuSnap Android application employs a modern, responsive user interface built using Jetpack Compose, Google's declarative UI toolkit for Android development. The frontend architecture follows Material Design 3 principles and implements a comprehensive document management system. Our frontend development is open-sourced at github.com/JI-DeepSleep/DocuSnap-Frontend. Our source code is organized as follows.

```
java.cn.edu.sjtu.deepsleep.docusnap
├── data
│   ├── local          # schema definition of on-device SQLite database
│   ├── model          # schema definition of in-app data structure
│   ├── remote         # schema definition to connect backend API
│   ├── repository     # global data storage and management
│   └── Settings.kt
├── navigation         # screen navigation controller
├── service            # main functionalities: image toolkit and backend caller
├── ui
│   ├── components
│   ├── screens        # definition of screen ui layout
│   └── theme
├── util
├── viewmodels         # connector of service and ui
├── AppModule.kt       # singleton to access service and repository
└── MainActivity.kt
```

**Illustration 4–1 Project structure of DocuSnap frontend**

### 4.1.1 Design Pattern

The application adopts a layered architecture pattern that separates concerns between UI components, business logic, and data management. The design follows the Model-View-ViewModel (MVVM) architectural pattern, which is the recommended approach for Android applications using Jetpack Compose.

#### 4.1.1.1 Architecture Components

The frontend architecture consists of several key components that work together to create a cohesive user experience. The **UI Layer** is built entirely with Jetpack Compose composables, providing a declarative approach to interface development. This layer handles all user interactions, visual rendering, and state presentation through screens like Home Page (the entry point), Search Screen, Document/Form Gallery and Detail, Camera Capture, Local Media Selection and Settings with PIN verification. Navigation between these screens is managed by the NavController through predefined Screen Routes.

The **ViewModel Layer**, consist of DocumentViewModel and ImageProcessingViewModel, manages functional logic and state using Android Architecture Components, acting as the bridge between the UI and data layers. These ViewModels receive user interactions from the UI components and coordinate with lower layers to process requests.

The **Repository Layer** provides data access through the repository pattern, ensuring global management of all files (documents and forms) stored in DocuSnap. The DocumentRepository serves as the single source of truth for document-related data operations.

Below this, the **Service Layer** contains specialized components: DeviceDBService handles local SQLite database operations, ImageProcService manages image processing toolkits, BackendApiService facilitates communications with backend APIs, and JobPollingService monitors background tasks to enable asynchronized backend procedure.

These services interact with the **Data Layer** which includes persistent storage through Room Database (for structured data like Document, Form, Job, and Search entities) and global Shared Preferences (for user settings).

The architecture integrates with **External Services** including CameraX API for camera functionality, MediaStore for local media access, and a Backend Server for finegrained processing, including OCR, LLM-based information extraction, etc. This layered approach ensures clear separation of concerns, with each component having distinct responsibilities and well-defined interaction patterns through interfaces.

#### 4.1.1.2 Technology Stack

The application leverages modern Android development technologies to ensure optimal performance and user experience. Jetpack Compose serves as the foundation for building

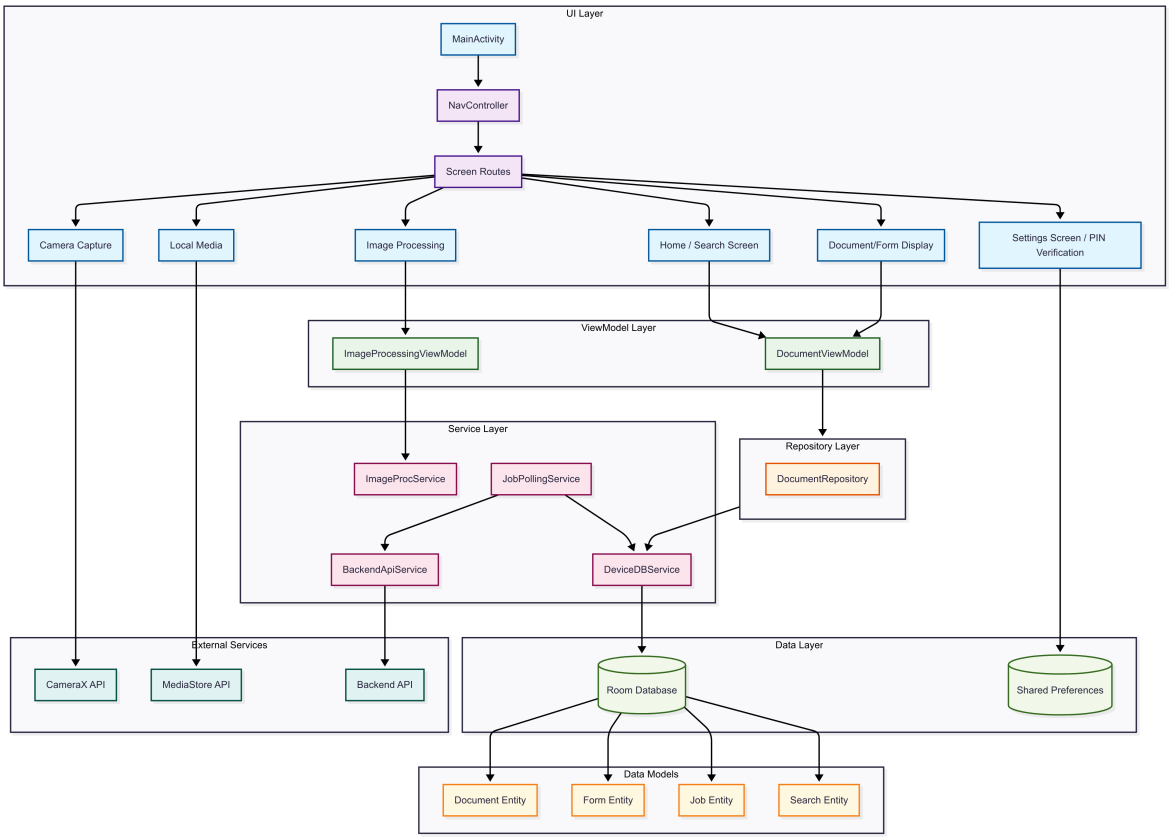


**Illustration 4–2 Frontend architecure of DocuSnap**

declarative, reactive user interfaces that automatically update based on state changes. Material Design 3 provides a comprehensive design system that ensures visual consistency and accessibility across all application screens. Jetpack Navigation offers type-safe navigation between screens, preventing runtime errors and improving code maintainability. ViewModel and StateFlow provide reactive state management, enabling efficient UI updates without unnecessary re-renders. Room Database handles local data persistence with SQLite abstraction, while CameraX provides modern camera functionality integration. Finally, Coil handles image loading and caching, ensuring smooth performance when displaying document images.

### 4.1.2 Individual UI Screen Design

The application features a comprehensive set of screens designed for optimal user experience in document management workflows. Each screen is carefully crafted to serve specific user needs while maintaining consistency with the overall design language.

#### 4.1.2.1 Home Screen

The home screen serves as the primary entry point and information hub for the application. It features a clean, hierarchical layout that guides users through the primary application functions. The screen begins with a branded header containing the application title and a settings access icon, establishing brand identity while providing quick access to configuration options. A prominent search bar enables global document and form retrieval, allowing users to quickly find specific content across their entire document library.

The main content area is divided into two primary sections: Document Import and Form Import. Each section features dual-button interfaces that provide access to both camera capture and gallery selection. The document section uses primary container colors to distinguish it from the form section, which employs secondary container colors. This visual distinction helps users understand the different processing workflows for documents versus forms. The bottom section displays frequently used text information extracted from documents and forms, providing quick access to commonly referenced data points.

#### 4.1.2.2 Document Gallery Screen

The document gallery presents a grid-based layout optimized for efficient document browsing and management. The screen implements a LazyVerticalGrid component that efficiently handles large document collections by only rendering visible items. Each document is represented by a custom card component that displays essential metadata including document name, description, upload date, and processing status. The cards feature subtle visual indicators for document state, such as processing completion and usage statistics.

The gallery includes advanced interaction patterns through the ExperimentalFoundationApi, enabling multi-selection capabilities through long-press gestures. Selected documents can be managed in bulk through context actions including deletion and export functionality. The screen also integrates search functionality directly within the gallery interface, allowing users to filter documents without navigating to a separate search screen.

#### 4.1.2.3 Form Gallery Screen

The form gallery follows a similar design pattern to the document gallery but is specialized for form management. Form cards display additional metadata specific to forms, including form field previews and completion status indicators. The interface distinguishes between completed forms and those still undergoing processing, providing clear visual feedback about the current state of each form. The gallery maintains consistency with the document gallery while highlighting form-specific features and workflows.

#### 4.1.2.4 Document and Form Detail Screens

The detail screens provide comprehensive information and editing capabilities for individual documents and forms. These screens implement a tabbed or scrollable layout that organizes information hierarchically. The primary view displays the document or form images using efficient image loading through the Coil library, which handles Base64-encoded image rendering with automatic caching and memory management.

The extracted information section presents key-value pairs in an organized, searchable format. Each extracted data point includes usage tracking information, showing how frequently specific information has been accessed. This feature helps users identify their most commonly used data points. The screens also provide in-place editing capabilities for document properties such as name, description, and tags, with real-time validation and error handling.

#### 4.1.2.5 Search Screen

The search screen implements a unified search experience that combines results from documents, forms, and extracted text entities. The interface presents search results with relevance scoring indicators, helping users quickly identify the most relevant matches. The screen supports type-based filtering, allowing users to narrow results to specific content types. The unified approach ensures users can find information regardless of whether it's stored in a document, form, or as extracted text data.

#### 4.1.2.6 Settings Screen

The settings screen provides comprehensive application configuration options in an organized, accessible format. The screen is divided into logical sections including security settings, backend configuration, and performance options. The PIN protection section allows users to enable and configure security features, while the backend configuration section provides access to server settings and cryptographic key management. Performance settings allow users to customize the number of frequently used text items displayed throughout the application.

### 4.1.3 Navigation Controller

The application implements a sophisticated navigation system using Jetpack Navigation Compose that provides type-safe routing and efficient screen transitions. The navigation architecture is designed to handle complex user workflows while maintaining performance and user experience.

#### 4.1.3.1 Navigation Architecture

The navigation system is built around a centralized NavHost component that manages all screen transitions and state. The system uses sealed classes to define routes, ensuring compile-time safety and preventing navigation errors. Each route is defined with specific parameters and argument types, enabling the passing of complex data between screens while maintaining type safety.

The navigation controller implements deep linking capabilities, allowing users to navigate directly to specific documents or forms from external sources. The system also handles back navigation intelligently, maintaining proper navigation state and preventing users from getting lost in complex workflows.

#### 4.1.3.2 Route Management

Routes are organized hierarchically to reflect the application's information architecture. Primary routes include home, search, camera capture, local media selection, image processing, document and form galleries, detail screens, settings, and PIN verification. Each route is designed to handle specific user tasks while maintaining consistency with the overall navigation patterns.

The navigation system supports complex argument passing, enabling screens to receive and process various data types including document IDs, image URIs, processing parameters, and user preferences. This capability is essential for maintaining state across the image processing workflow and ensuring users can seamlessly move between different stages of document creation and editing.

#### 4.1.3.3 Bottom Navigation

The application features a bottom navigation bar that provides quick access to the three primary application areas: Forms, Home, and Documents. The bottom navigation uses Material Design 3 components with proper accessibility support and visual feedback. The navigation bar automatically hides when users are in detailed views or processing workflows, ensuring maximum screen real estate for content display.

### 4.1.4 ViewModel and UI-Service Integration

The application employs a sophisticated ViewModel architecture that bridges UI components with backend services and data management. This architecture ensures clean separation of concerns while providing reactive, efficient state management.

#### 4.1.4.1 DocumentViewModel Architecture

The DocumentViewModel serves as the primary business logic container, managing all document and form-related operations. The ViewModel implements reactive programming patterns using Kotlin Coroutines and StateFlow, ensuring that UI updates are efficient and responsive. The ViewModel maintains separate state containers for documents, forms, search results, and frequently used text information, enabling granular control over UI updates.

The ViewModel implements comprehensive error handling and loading state management, providing users with clear feedback about operation status. All data operations are performed asynchronously using coroutines, preventing UI blocking and ensuring smooth user experience even during complex operations.

#### 4.1.4.2 Reactive State Management

The ViewModel uses StateFlow for state management, providing several advantages over traditional LiveData. StateFlow offers better integration with Kotlin Coroutines, improved error handling, and more efficient state updates. The reactive architecture ensures that UI components automatically update when underlying data changes, without requiring manual refresh operations.

The state management system implements proper lifecycle awareness, ensuring that operations are cancelled when ViewModels are destroyed and preventing memory leaks. The system also handles configuration changes gracefully, maintaining state across device rotations and other configuration changes.

#### 4.1.4.3 Service Integration

The ViewModel integrates with various services through the repository pattern, providing a clean abstraction layer between UI and business logic. The DeviceDBService handles all local database operations, including document and form storage, retrieval, and updates. The ImageProcService manages image processing operations, including corner detection, perspective correction, and image enhancement. The JobPollingService handles background job management, ensuring that long-running operations don't block the UI. The BackendApiService manages communication with remote servers for document processing and synchronization.

#### 4.1.4.4 Usage Tracking and Analytics

The ViewModel implements sophisticated usage tracking for extracted information, enabling the application to provide personalized experiences and insights. The system tracks how frequently specific data points are accessed, when they were last used, and which documents or forms contain the most valuable information. This data is used to populate the frequently used text information section and to optimize search results.

The usage tracking system maintains privacy by storing all analytics data locally and providing users with control over what information is tracked. The system also implements proper data retention policies, automatically cleaning up old usage data to prevent storage bloat.

### 4.1.5 Image Import and Processing

The application provides comprehensive image import capabilities through both camera capture and local media selection, with advanced processing features that ensure optimal document quality and readability.

#### 4.1.5.1 Camera Integration

The camera functionality is implemented using CameraX, Google's modern camera library that provides consistent behavior across different Android devices. The camera implementation includes comprehensive permission management, ensuring that users are properly informed about camera access requirements and can grant permissions when needed.

The camera interface provides advanced features including zoom control through pinch gestures, manual exposure compensation for optimal lighting, and flashlight control for low-light conditions. The interface supports multiple image capture, allowing users to photograph multi-page documents or capture multiple angles of the same document for better processing results.

#### 4.1.5.2 Camera Features and User Experience

The camera interface is designed to provide professional-grade document capture capabilities while remaining accessible to casual users. The preview display shows a real-time camera feed with optional grid overlays to help users align documents properly. The interface includes visual guides and feedback to help users position documents correctly for optimal processing results.

The camera implementation handles various edge cases including permission denials, hardware failures, and storage issues. The system provides clear error messages and recovery options when problems occur, ensuring users can continue their workflow even when facing technical difficulties.

#### 4.1.5.3 Local Media Selection

The local media selection uses the modern Activity Result API, providing a streamlined experience for selecting images from the device gallery. The implementation supports multiple image selection, allowing users to process several documents simultaneously. The system handles various image formats and sizes, automatically optimizing images for processing while maintaining quality.

The media selection process includes proper error handling for cases where selected images are corrupted, inaccessible, or in unsupported formats. The system provides clear feedback about selection status and automatically proceeds to the processing stage when valid images are selected.

#### 4.1.5.4 Image Processing Pipeline

The image processing screen provides comprehensive image enhancement capabilities designed specifically for document processing. The processing pipeline includes several stages that can be applied individually or in combination depending on the specific document requirements.

The corner detection feature uses computer vision algorithms to automatically identify document boundaries in captured images[14]. This feature is particularly useful for documents that weren't perfectly aligned during capture. The system can detect and highlight document corners, allowing users to verify detection accuracy before proceeding with processing.

Perspective correction automatically adjusts document orientation and perspective to create perfectly rectangular documents from angled captures. The system provides both automatic and manual correction options, giving users control over the final result. Manual correction allows users to fine-tune corner positions for optimal results.

#### 4.1.5.5 Image Enhancement Features

The image enhancement features include color enhancement for improving document readability, contrast adjustment for better text visibility, and threshold filtering for creating clean black-and-white versions of documents. The high contrast mode is particularly useful for documents with poor lighting or low contrast, ensuring that text remains readable even in challenging conditions.

The processing pipeline implements adaptive algorithms that automatically adjust processing parameters based on image characteristics. This ensures optimal results across different document types, lighting conditions, and image qualities. The system also provides preview capabilities, allowing users to see the effects of different processing options before applying them.

#### 4.1.5.6 Data Management and Storage

Images are processed and stored using Base64 encoding to ensure compatibility with the local database system. The encoding process includes compression and optimization to minimize storage requirements while maintaining image quality. The system implements proper memory management to handle large images efficiently, preventing out-of-memory errors during processing.

The application uses FileProvider for secure file sharing and temporary storage, ensuring that image files are handled securely and efficiently. The FileProvider implementation includes proper permission management and cleanup procedures to prevent storage bloat and security issues.

This comprehensive frontend design demonstrates the application's commitment to modern Android development practices, providing users with an intuitive and powerful interface for document management and processing. The architecture ensures scalability, maintainability, and optimal performance while delivering a rich user experience through advanced UI components and seamless integration with backend services.

### 4.1.6 Image Processing Logic and Algorithms

While the previous sections described the user-facing UI for image handling, this section details the underlying technical implementation. It focuses on the end-to-end data pipeline for images and the core computer vision algorithms employed to enhance document quality and readability.

#### 4.1.6.1 End-to-End Image Data Pipeline

The image processing pipeline provides a streamlined flow for handling images from acquisition to final storage. The process begins with image acquisition, where both the CameraX API for new photos and the `ActivityResultContracts` API for gallery selections yield standard URIs. These URIs are then passed as route arguments via Jetpack Navigation to the ImageProcessingScreen. Here, the ImageProcessingViewModel decodes each URI into a `Bitmap`, which serves as the active canvas for all subsequent operations. This decoding step includes downscaling images larger than 1080p to prevent memory errors. Upon completion of all user-directed enhancements, the final, processed `Bitmap` is encoded into a Base64 string, encapsulated within a `Document` entity, and persisted in the database.

#### 4.1.6.2 Geometric Correction

The geometric correction feature is designed to rectify perspective distortions, transforming a skewed image into a rectangular, front-facing document[15-17]. This is accomplished through a self-contained algorithmic pipeline within the `ImageProcService` that combines advanced edge detection[18] with a final linear transformation[19].

The process is initiated with essential pre-processing steps, where the input `Bitmap` is converted to grayscale and a 5x5 Gaussian blur is applied to reduce image noise[20]. Subsequently, the Canny edge detection algorithm[18] is used to produce a binary map of the document's potential edges. To improve the integrity of these edges, the system performs morphological operations—specifically dilation followed by erosion—to connect broken lines and remove noise artifacts. The resulting clean edge map is then scanned to find all closed contours, with the largest contour by area assumed to be the document's boundary. From this primary contour, the four corner points are identified using a geometric heuristic that locates the extremities of the shape[15].

Once these four corner points are determined, the pipeline immediately proceeds to the linear transformation stage. This is accomplished by the `performPerspectiveCorrection` function, which calculates the required perspective transformation matrix[16]. It defines the four detected points as a source quadrilateral and a new, perfectly rectangular bitmap as the destination. Using Android's `Matrix.setPolyToPoly` method, it computes the homography matrix that maps the source to the destination points. This matrix is then used to warp the original image, producing the geometrically corrected final output[19].

#### 4.1.6.3 Grayscale Conversion Algorithm

Grayscale conversion is a fundamental pre-processing step for nearly all document analysis tasks, as it simplifies the image data from three color channels to a single intensity channel.

The application implements the standard Luminance method for this conversion, which weights the red, green, and blue components of each pixel according to human visual perception. The specific formula applied within the ImageProcService is:

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

where $Y$ is the resulting grayscale intensity, and $R, G, B$ are the original color components. This calculation is performed pixel-by-pixel to transform the source `Bitmap` into a grayscale representation ready for further processing.

#### 4.1.6.4 High Contrast Algorithm

To address the common issue of low contrast in document photography, which can result from poor lighting or shadows, the application employs a powerful high-contrast filter. This feature is technically implemented using the Histogram Equalization algorithm. The purpose of this algorithm is to improve global contrast by redistributing pixel intensities to span the entire available dynamic range. The implementation in `applyHighContrast` begins by computing a grayscale histogram of the image, which involves iterating through all pixels to count the frequency of each intensity value, from 0 (black) to 255 (white). From this histogram, a Cumulative Distribution Function (CDF) is calculated, which maps each intensity value to the cumulative number of pixels at or below that intensity, providing a representation of the image's tonal distribution. The CDF is then normalized and used to create a Look-Up Table (LUT). This LUT serves as a mapping function that remaps the original pixel intensities to a new set of values, effectively "stretching" the most frequent intensity ranges to cover the entire 0-255 spectrum. Finally, this LUT is applied to every pixel of the grayscale image; each pixel's original intensity value is replaced with its new, remapped value from the LUT to produce the final, high-contrast image with significantly improved readability.

## 4.2 Back-end Development

### 4.2.1 In-App Database

A critical component of the DocuSnap Android app is its in-app database, which is designed to provide secure, reliable, and efficient local data storage. Built using SQLite via Android's Room persistence library, the database ensures structured management of documents, forms, and job states, all of which are central to the app's core workflows.

The database schema comprises three primary entities: `DocumentEntity`, `FormEntity`, and `JobEntity`. `DocumentEntity` stores user-uploaded documents along with metadata such as titles, creation timestamps, and extracted textual information. `FormEntity` captures user-submitted forms, their associated structure, and any autofilled fields. Meanwhile, `JobEntity` tracks the backend processing jobs, including their type ("doc", "form", or "fill"), status, timestamps, and returned content. Each entity is fully integrated with Room's DAO layer to support complete CRUD (Create, Read, Update, Delete) operations.

To ensure compatibility, security, and flexibility, the system adopts multiple encoding and serialization formats. Document and form images are encoded in Base64 for safe transmission over JSON-based APIs, while structured data such as form fields and extracted information are serialized using JSON. Additionally, SHA-256 hashing is used to verify content integrity before and after transmission, offering a lightweight yet reliable safeguard against tampering or corruption.

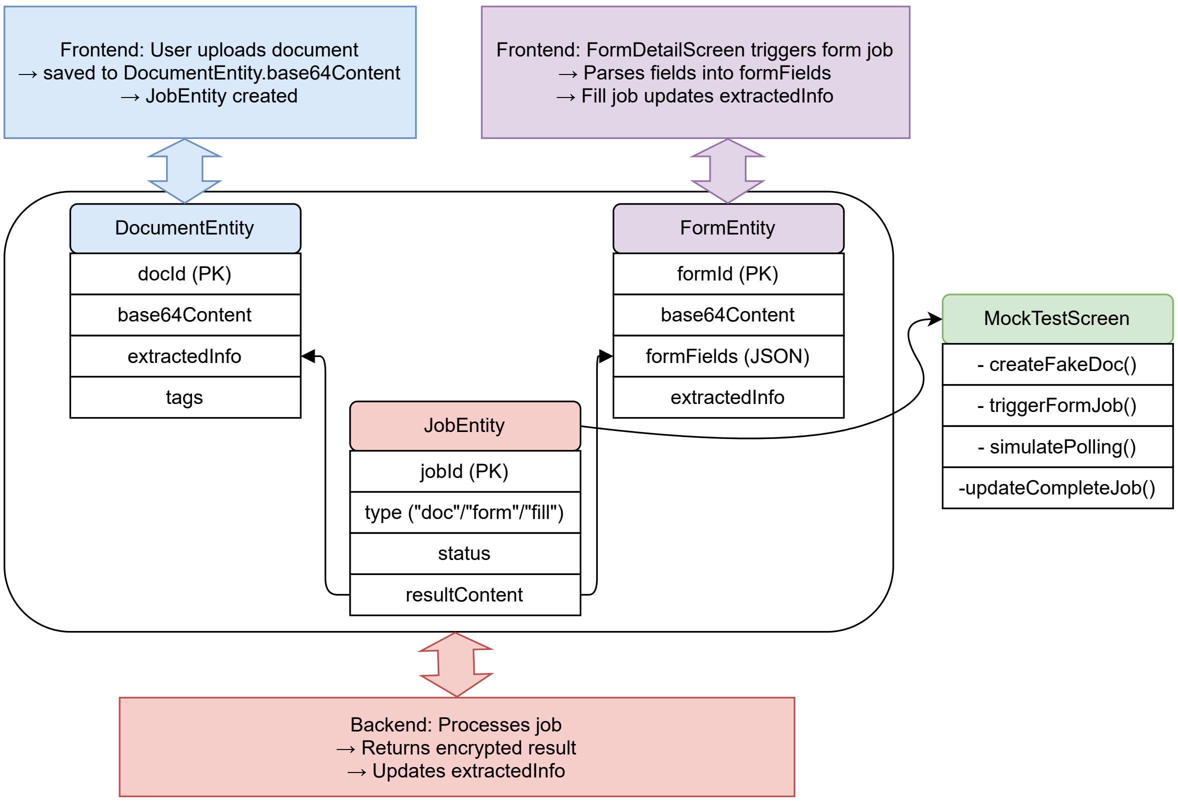


**Illustration 4–3 In-App Database Schema and Data Flow**

Data access throughout the app is mediated via a repository layer, which encapsulates all interactions with the underlying Room DAOs. This abstraction not only promotes clean separation of concerns but also simplifies testing and future migration. For example, the `DocumentRepository` provides high-level methods such as `insertDoc()`, `deleteDoc()`, and `updateExtractedInfo()`, while `JobRepository` handles creation and updates of job states. These repositories are injected into ViewModel classes, which in turn expose observable state to the UI layer using LiveData or StateFlow.

The in-app database also plays a central role in inter-module communication. It provides a bridge between the UI, backend polling logic, and encryption layer. For instance, the `JobPollingService` monitors remote job completion and writes the results back into the local database. When a "doc" job completes, the corresponding `extractedInfo` is automatically updated with decrypted backend output. Similarly, when a "form" job completes, the app parses the returned JSON and updates both `formFields` and `extractedInfo` fields of the related `FormEntity`. This mechanism ensures real-time synchronization between backend state and the user interface.

Robust error handling and data integrity mechanisms are built into the system. All write operations are transactional, ensuring atomicity and rollback on failure. The polling service includes retry logic with exponential backoff for transient failures, while integrity validation using cryptographic hashes prevents inadvertent data corruption. These features collectively enhance the app's resilience under various network and runtime conditions.

Finally, the database module is built with extensibility in mind. JSON-based field storage allows future expansion of form and document schemas without requiring disruptive schema migrations. The modular DAO and repository architecture supports the integration of advanced querying features, such as fuzzy keyword search or filtered document retrieval. Moreover, the database abstraction ensures that future transitions to encrypted storage engines or cloud-synced backends can be achieved with minimal disruption to other system components.

In summary, the in-app database of DocuSnap provides a secure, efficient, and extensible foundation for persistent data management. Its tight integration with other system components—such as encryption, job processing, and the user interface—ensures a seamless user experience while adhering to modern architectural and security standards.

### 4.2.2 Server Backend

#### 4.2.2.1 Bachend Architecture

Open-sourced at github.com/JI-DeepSleep/DocuSnap-Backend, our backend implementation embodies a security-first architecture designed around two fundamental constraints inherent to document processing systems: the asynchronous nature of LLM operations and stringent requirements for handling sensitive identification documents. The **asynchronous architectural** decisions stem from careful analysis of operational realities—specifically, LLM requests exhibit orders-of-magnitude higher latency compared to conventional web transactions, typically requiring 5-30 seconds for completion. This latency creates significant risks of connection termination by network intermediaries (proxies, ISPs) or premature client disengagement when users navigate away from processing pages. To address this, we implemented a polling-based workflow where clients submit jobs through ephemeral HTTPS connections and periodically check status updates, eliminating long-lived connections vulnerable to interruption. This approach not only aligns with the asynchronous API design of our LLM provider Zhipu AI but also accommodates Android's background process limitations where applications may be terminated during extended operations.

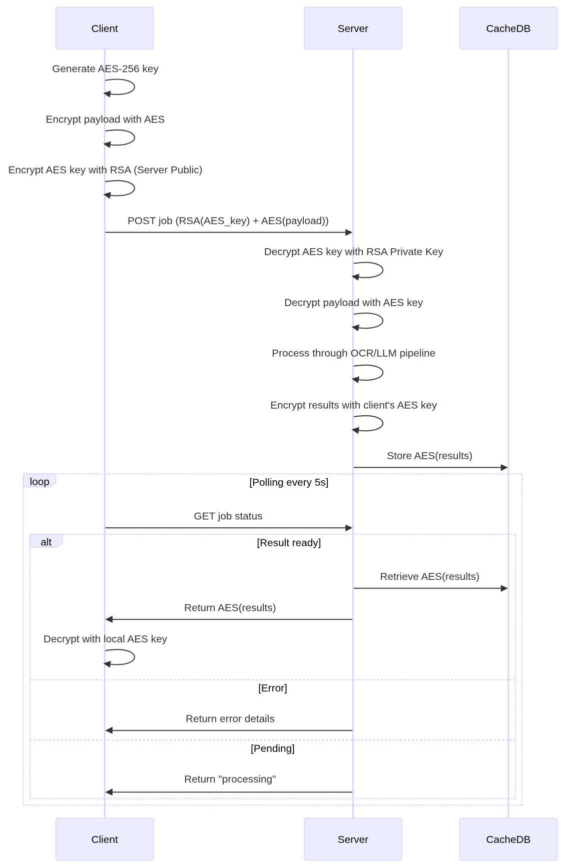


**Illustration 4–4 Hybrid encryption workflow for document processing**

Security considerations necessitated a **zero-trust architecture** where sensitive documents never persist in plaintext on server infrastructure. The design acknowledges that traditional TLS encryption alone is insufficient against compromised certificate authorities or server breaches. As illustrated in Figure 4–4, we employ a hybrid cryptographic scheme combining RSA-2048 and AES-256 encryption. During client initialization, the server's RSA public key is embedded in the mobile application. When submitting documents, the client generates a unique AES-256 key to encrypt the payload, then asymmetrically encrypts this ephemeral key using the server's RSA public key. A man-in-the-middle attacks enabled by compromised certificate authorities will fail because the attackers have no way of getting the true server's private key. Also, this dual-layer approach resolves the performance limitations of pure RSA encryption—benchmarks on our OrangePi 5 Pro hardware showed RSA alone required 10+ seconds for 10MB payloads versus 0.1 seconds for hybrid encryption. The server decrypts the AES key using its private key, processes the document through the OCR-LLM pipeline, and encrypts results with the client's AES key before storage. Crucially, AES keys exist only in volatile memory during processing, and cached results automatically expire after a configurable period (default: 1440 minutes) or upon explicit client deletion requests. This ensures document plaintext is never written to persistent storage, and compromised servers yield only encrypted data that remains cryptographically inaccessible without client-held keys. Admittedly, with budget and resource limitations, current dependencies on Zhipu LLM pose a potential threat of sensitive information leakage. However, if we migrate to locally-hosted LLM models when computational resources permit, it would eliminate third-party LLM dependencies and keep all sensitive data under control in volatile memory.

#### 4.2.2.2 Backend Implementation

The backend operates on an OrangePi 5 Pro single-board computer featuring a Rockchip RK3588S octa-core ARM processor clocked at 2.4GHz with 4GB RAM[21]. As depicted in Figure 4–5, the system employs a multi-layered service architecture: Flask serves as the application server behind Gunicorn WSGI workers and Nginx reverse proxy, while Cloudflare Tunnel provides secure external access without public IP exposure. The OCR subsystem utilizes CnOCR with thread-pooled workers (configurable concurrency: 4 threads) to handle image processing[22]. A SQLite DB is used for the Cache DB to temporarily store processing results.

Prompt engineering constituted a critical component for structuring LLM outputs, with specialized templates developed through iterative refinement. Our approach incorporates several key innovations: 1) Predefined field templates covering 85+ common document attributes ensure consistent extraction, 2) Delimited content sections using `<ocr_content>` tags explicitly separate instructions from input data to prevent prompt injection, 3) Strict JSON output formatting with enforced escape character handling (\\, \", etc.) guarantees machine-readable results, 4) Example-based guidance improves instruction following, and 5) Recursive field disambiguation logic expands nested JSON structures into flat key-value pairs using depth-first traversal, resolving inconsistencies where LLMs occasionally return hierarchical data. The prompt templates and JSON normalization functions are documented

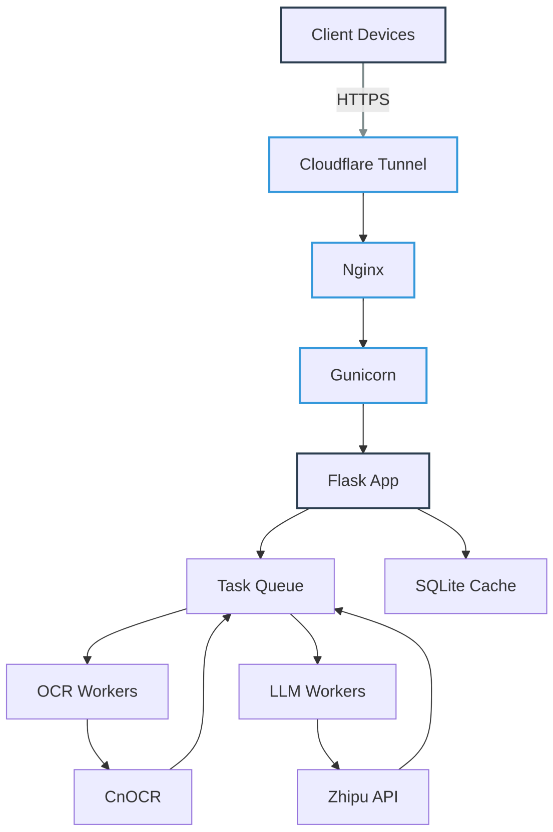


**Illustration 4–5 Backend service architecture and component interactions**

in the repository's `prompts.py` module.

#### 4.2.2.3 Client Integration

The Android application interacts with the backend through a dedicated service layer (`BackendApiService.kt`) that abstracts cryptographic operations and network communication. This component handles RSA public key embedding during app initialization, AES-256 key generation using Android's SecureRandom, payload encryption via javax.crypto implementations, and SHA-256 checksum validation for data integrity. The service implements a coroutine-based workflow where document processing jobs are submitted asynchronously, with results retrieved through a persistent WorkManager instance that polls task status at 5-second intervals. Cryptographic operations utilize Android Keystore-backed implementations to ensure secure key handling on client devices, with ephemeral keys cached in memory during active sessions but never persisted to device storage. The architecture supports three primary operations: document parsing, form parsing, and form filling, all implemented as suspend functions that integrate with Android's lifecycle-aware components.

#### 4.2.2.4 Summary

The backend system provides a secure document processing pipeline through hybrid encryption that prevents plaintext persistence on servers while maintaining practical performance thresholds. Asynchronous job handling via polling ensures reliable LLM operations despite network volatility and client-side interruptions. The implementation demonstrates that zero-trust security principles can be effectively applied to resource-constrained edge devices, with prompt engineering techniques enabling structured data extraction from diverse document formats. Future work includes

## 4.3 Testing Results

### 4.3.1 Testing Tool

The application was developed exclusively for the Android platform, targeting Android 13 (API level 33) and higher. Testing was performed using both virtual emulators and physical devices to ensure compatibility and performance validation across different hardware configurations.

For virtual testing, all team members utilized Android Studio's built-in emulator with device profiles ranging from Pixel 4 to newer models. These emulators employed system images with Google API support, specifically configured for Android 13 and above to replicate real-world operating conditions.

Physical testing was conducted on a Xiaomi Redmi 14C (4GB RAM + 64GB storage variant) running Android 15. This device was intentionally selected as a performance baseline, purchased new from Xiaomi's official Taobao flagship store for just 377 RMB. As one of the most affordable Android devices available during the testing period, its modest hardware specifications provided a rigorous environment for evaluating application performance. Subsequent sections will demonstrate the application maintained smooth operation even under these constrained conditions.

Screenshots of the testing environments are provided in Appendix 6.

### 4.3.2 UI Testing

The user interface underwent comprehensive testing to verify the functionality of key interactive elements. All primary navigation buttons were tested for responsiveness and correct routing behavior, with results confirming expected performance as documented in Table 4–1.

Critical action buttons across various screens were validated for proper functionality. Testing covered buttons in the image processing screen (Table 4–2), document detail screen (Table 4–3), and form detail screen (Table 4–4). While most buttons functioned as intended, the "Auto" and "Color Enhancement" features in the image processing screen exhibited inconsistent behavior that requires further investigation.

Selection mode functionality was thoroughly evaluated in both document and form galleries. The testing confirmed that a long press on any preview card successfully activates selection mode (Figures 4–6a and 4–6e), with subsequent taps properly toggling the selection state of additional cards. The interface correctly displays the number of selected files and provides effective bulk operation controls. The top-right action button's select/deselect all functionality performed as expected (Figures 4–6b and 4–6f). Export operations successfully downloaded all selected files to local storage (Figures 4–6c and 4–6g), while delete operations properly triggered confirmation dialogs for batch deletion (Figures 4–6d and 4–6h).

### 4.3.3 Acceptance Testing for Features

#### 4.3.3.1 Feature Group: Document and Form Understanding

To evaluate the functionality of our intelligent document processing pipeline, we conducted acceptance testing on three core features—*parse document*, *parse form*, and *fill form* —which collectively form the backbone of semantic document understanding in our system. These features were tested using representative real-world document types, including a government-issued driver's license and a U.S. visa application form. The goal was to validate the system's ability to perform accurate key-value extraction, intelligent categorization, cross-document linkage, and automatic field population.

The **parse document** feature was tested using a California driver license sample with a resolution of 1314×698 pixels. The expected behavior was that the system would correctly detect the document type, extract structured fields such as full name, address, date of birth, license number, and expiration date, and attach semantic tags to aid in categorization and

Table 4–1 Test result of all navigation buttons

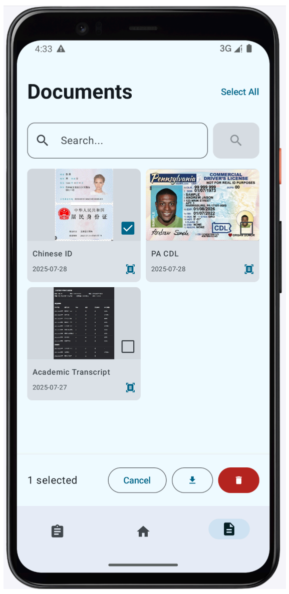
| Screen | Button | Expected Behavior | Result |
|---|---|---|---|
| Home | Search | Open search page | ✓ |
| | Setting | Open setting page | ✓ |
| | Camera | Open camera for photo capturing | ✓ |
| | Gallery | Open local media for photo selection | ✓ |
| | Link (Frequently Used Info) | Jump to source file detail page of this textual info | ✓ |
| Search Result Page | Link (Doc or Form) | Jump to detail page of this file | ✓ |
| | Link (Textual Info) | Jump to source file detail page of this information | ✓ |
| Camera Screen | Gallery | Open local media for photo selection | ✓ |
| | Done | Proceed to the image processing with captured images | ✓ |
| Local Media | Done | Proceed to the image processing with selected local images | ✓ |
| Image Processing | Done | Save changes and navigate to file detail | ✓ |
| Document Gallery | Preview | Jump to detail page of this document | ✓ |
| | Search | Open search page | ✓ |
| Form Gallery | Preview | Jump to detail page of this form | ✓ |
| | Search | Open search page | ✓ |
| Document Detail | Link (Related Files) | Jump to detail page of this related file | ✓ |
| | Back | Back to document gallery | ✓ |
| Form Detail | Link (Related Files) | Jump to detail page of this related file | ✓ |
| | Link (Filled Form Fields) | Jump to source file detail page of this extracted field | ✓ |
| | Back | Back to form gallery | ✓ |
| Bottom Navigation Bar | Home | Shortcut to homepage | ✓ |
| | Document | Shortcut to document gallery | ✓ |
| | Form | Shortcut to form gallery | ✓ |

**Table 4–2 Test result of all action buttons in image processing page**

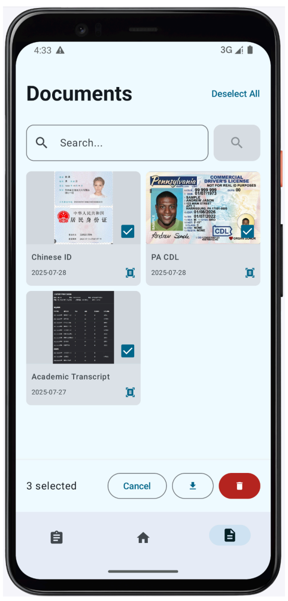
| Group | Button | Expected Behavior | Result |
|---|---|---|---|
| Primary Toolbar | Reset | Clear all applied effects | ✓ |
| | Filter | Toggle the secondary toolbar with enhancement options | ✓ |
| | Perspective | Toggle the secondary toolbar with edge adjustment options | ✓ |
| | Auto | Apply auto processing template: monochrome + straighten | × |
| Secondary Enhancement Toolbar | Original | Use original image color schema | ✓ |
| | Monochrome | Apply monochrome filter: binarize pixel color to optimize text-background separation | ✓ |
| | High Contrast | Apply high contrast filter: amplify grayscale gap among pixels | ✓ |
| | Color Enhancement | Apply color enhancement filter: preserve color while amplify difference | × |
| Secondary Perspective Toolbar | Straighten | Apply automatic edge detection and display the detected 4 points | ✓ |
| | Apply | Apply the perspective correction based on detected and manually-corrected 4 points | ✓ |

**Table 4–3 Test result of all action buttons in document detail page**

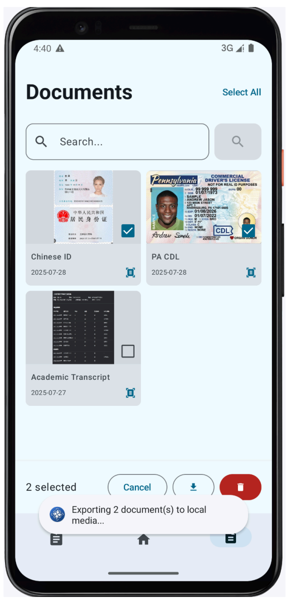
| Button | Expected Behavior | Result |
|---|---|---|
| Export | Export the document images to local media | ✓ |
| Parse | Initiate a background job to call backend document parsing API | ✓ |
| Edit | Toggle edit mode to manually modify extracted information values | ✓ |
| Clear | Remove all extracted information from the document database | ✓ |
| Copy | Copy all extracted information to clipboard in key-value format. | ✓ |
| Help | Pop up a help dialogue with explanation of each buttons | ✓ |
| Copy (Info value) | Copy the specific information value to clipboard | ✓ |
| Delete | Delete the document and navigate back to document gallery | ✓ |



(a)

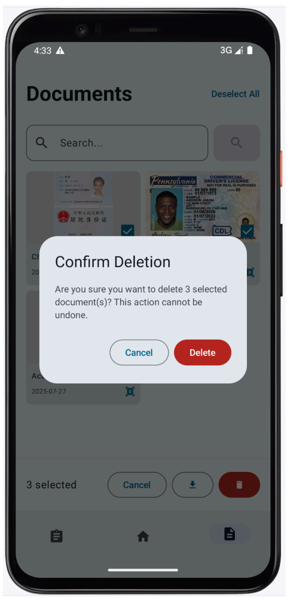


(b)



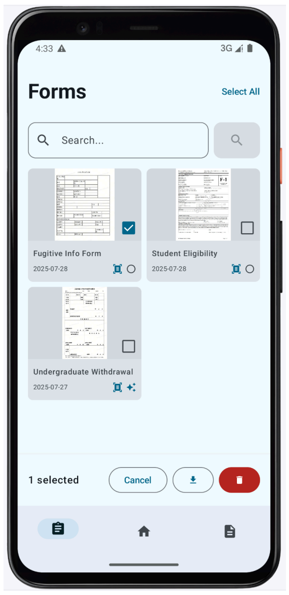
Exporting 2 document(s) to local media...

(c)

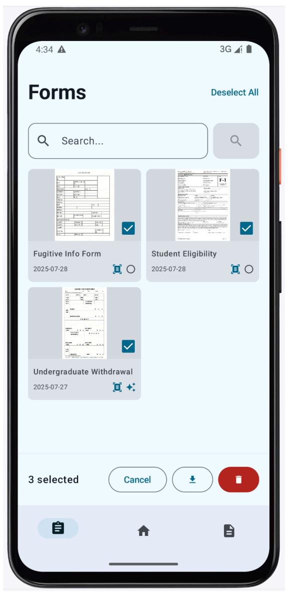


Are you sure you want to delete 3 selected document(s)? This action cannot be undone.

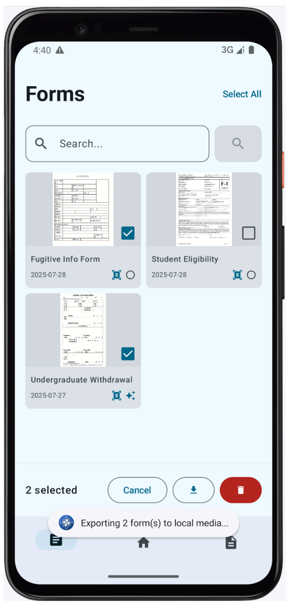
(d)



(e)

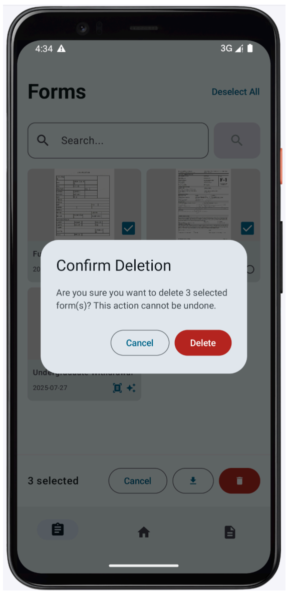


(f)



Exporting 2 form(s) to local media...

(g)



Are you sure you want to delete 3 selected form(s)? This action cannot be undone.

(h)

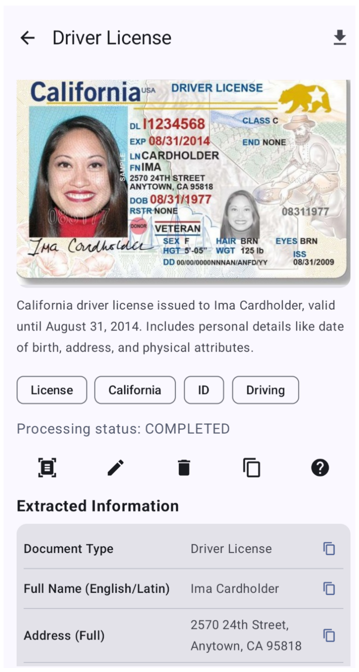
**Illustration 4–6 Test result of selection mode in document and form gallery**

**Table 4–4 Test result of all action buttons in form detail page**

| Button | Expected Behavior | Result |
|---|---|---|
| Export | Export the document images to local media | ✓ |
| Parse | Initiate a background job to call backend form parsing API | ✓ |
| Autofill | Initiate a background job to call backend auto form filling API | ✓ |
| Edit | Toggle edit mode to manually modify extracted information values | ✓ |
| Clear form | Clear all form field values while keeping the field names intact | ✓ |
| Clear all | Remove all extracted information and form fields completely | ✓ |
| Copy | Copy all extracted information and form fields to clipboard | ✓ |
| Help | Pop up a help dialogue with explanation of each buttons | ✓ |
| Copy (info value) | Copy the selected extracted info value to clipboard | ✓ |
| Copy (form field) | Copy the selected filled form field value to clipboard | ✓ |
| Delete | Delete the document and navigate back to document gallery | ✓ |

search. Initially, the document appeared with no extracted information and a generic placeholder title. Upon invoking the parsing engine, the system accurately extracted all target fields. For instance, the full name was identified as "Ima Cardholder," and the license number as "DL11234568." Semantic tags such as "License," "California," "ID," and "Driving" were generated, and a natural language summary of the document was created. These outputs matched the visible content of the document and demonstrated robust layout-aware OCR and metadata structuring. Based on the completeness and precision of the extracted data, we conclude that the parse document feature met its functional requirements.

Next, the **parse form** feature was evaluated using a nonimmigrant visa application form (resolution: 1244×694), a highly structured, multi-field document requiring contextual understanding. The expected functionality was for the system to extract field-value pairs from typed text and identify empty or missing fields. Additionally, the system was expected to link values from previously parsed documents (e.g., the driver license) where applicable. After upload, the form entered a processing state and was automatically renamed to "Nonimmigrant Visa App." Semantic tags such as "Visa," "Application," "US," and "Travel" were applied, and a textual description of the form's purpose was generated. Crucially, the form parser successfully extracted the user's sex ( "F" ) and populated the "Home Ad-



Driver License

California driver license issued to Ima Cardholder, valid until August 31, 2014. Includes personal details like date of birth, address, and physical attributes.

Processing status: COMPLETED

Extracted Information

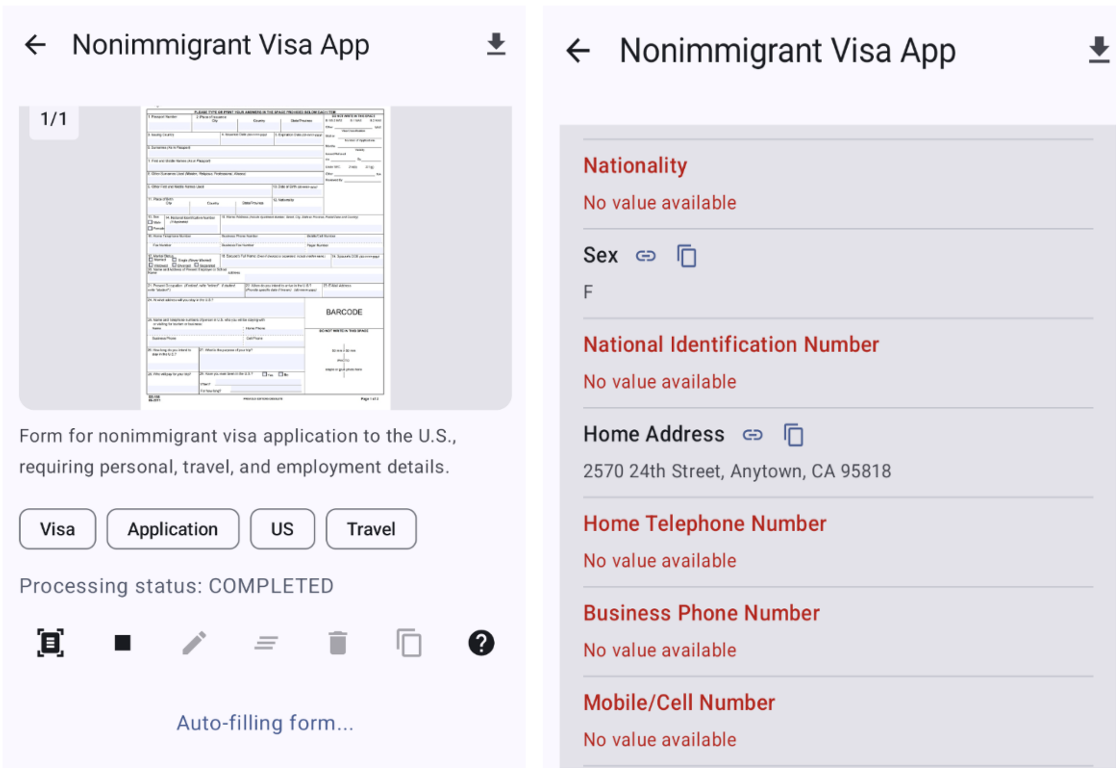
| | |
|---|---|
| Document Type | Driver License |
| Full Name (English/Latin) | Ima Cardholder |
| Address (Full) | 2570 24th Street, Anytown, CA 95818 |

**Illustration 4–7 Test result: document parsing inferface**

dress" field by linking it to the address extracted from the driver license. Fields not present in the source document, such as "Nationality" and "Phone Numbers," were appropriately marked as "No value available." This behavior reflects the system's ability to both extract data and reason over missing content. We consider this feature functionally complete with strong performance in layout parsing and early-stage cross-document reference.

Finally, the **fill form** feature was assessed to determine whether the system could automatically populate fields in a structured form using data from related documents. The test involved revisiting the parsed visa application after parsing the driver license. The system correctly inferred the shared context and auto-filled the "Home Address" and "Sex" fields. Moreover, the user interface allowed manual editing of each field post-fill, supporting workflows where human validation is necessary. Users could interact with individual fields using clipboard operations (copy, paste), enabling practical reuse of values across applications. This demonstrates that the fill form component not only supports automation but also provides user agency for refinement and correction.

Overall, this group of features—parse document, parse form, and fill form—collectively achieves the goal of transforming static documents into semantically rich, editable, and reusable


Nonimmigrant Visa App

1/1

Form for nonimmigrant visa application to the U.S., requiring personal, travel, and employment details.

Visa | Application | US | Travel

Processing status: COMPLETED

Auto-filling form...

Nonimmigrant Visa App

Nationality
No value available

Sex
F

National Identification Number
No value available

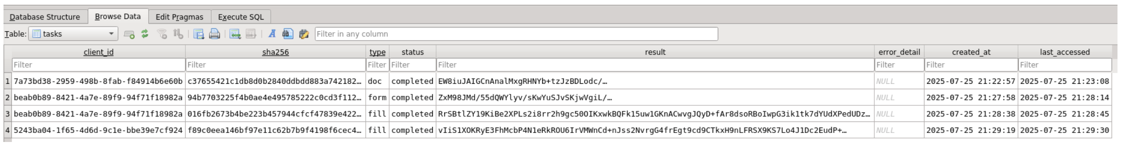
Home Address
2570 24th Street, Anytown, CA 95818

Home Telephone Number
No value available

Business Phone Number
No value available

Mobile/Cell Number
No value available

**Illustration 4–8 Test result: form parsing and form filling interface**



| | client_id | sha256 | type | status | result | error_detail | created_at | last_accessed |
|---|---|---|---|---|---|---|---|---|
| 1 | 7a73bd38-2959-498b-8fab-f84914b6e60b | c37655421c1db8d0b2840ddbdd883a742182… | doc | completed | EW8iuJAIGCnAnalMxgRHNYb+tzJzBDLodc/… | NULL | 2025-07-25 21:22:57 | 2025-07-25 21:23:08 |
| 2 | beab0b89-8421-4a7e-89f9-94f71f18982a | 94b7703225f4b0ae4e495785222c0cd3f112… | form | completed | ZxM98JMd/55dQWYlyv/sKwYuSJvSKjwVgiL/… | NULL | 2025-07-25 21:27:58 | 2025-07-25 21:28:14 |
| 3 | beab0b89-8421-4a7e-89f9-94f71f18982a | 016fb2673b4be223b457944cfcf47839e422… | fill | completed | RrSBtlZY19KiBe2XPLs2i8rr2h9gc50OIKxwkBQFk15uw1GKnACwvgJQyD+fAr8dsoRBoIwpG3ik1tk7dYUdXPedUDz… | NULL | 2025-07-25 21:28:38 | 2025-07-25 21:28:45 |
| 4 | 5243ba04-1f65-4d6d-9c1e-bbe39e7cf924 | f89c0eea146bf97e11c62b7b9f4198f6cec4… | fill | completed | vIiS1XOKRyE3FhMcbP4N1eRkROU6IrVMWnCd+nJss2NvrgG4frEgt9cd9CTkxH9nLFRSX9KS7Lo4J1Dc2EudP+… | NULL | 2025-07-25 21:29:19 | 2025-07-25 21:29:30 |

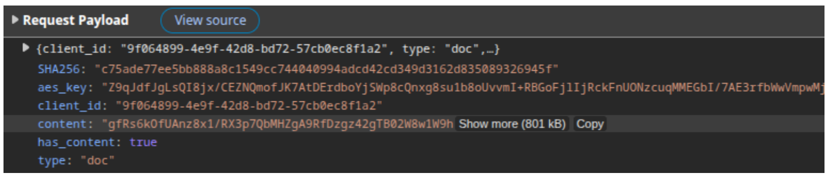
**Illustration 4–9 Test result: encrypted server database**

data assets. The tests confirm that the system performs accurate field extraction, intelligent labeling, and automated information reuse across document boundaries, validating the underlying architecture and design goals of our intelligent document management system.

#### 4.3.3.2 Feature Group: Backend Server Encryption and Security

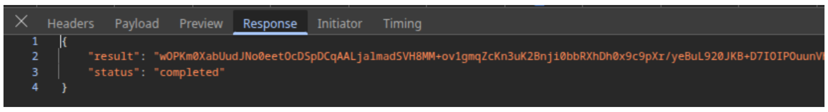
This section evaluates two interconnected security features: encrypted server database and end-to-end document encryption.

The first feature, encrypted server database, requires that the "result" field within the server's database is stored in encrypted form. The second feature, end-to-end document encryption, mandates that when examining request and response payloads independently of TLS, both the request "content" field and response "result" field must be encrypted.



```
▶ Request Payload   View source
▶ {client_id: "9f064899-4e9f-42d8-bd72-57cb0ec8f1a2", type: "doc",…}
   SHA256: "c75ade77ee5bb888a8c1549cc744040994adcd42cd349d3162d835089326945f"
   aes_key: "Z9qJdfJgLsQI8jx/CEZNQmofJK7AtDErdboYjSWp8cQnxg8su1b8oUvvmI+RBGoFjlIjRckFnUONzcuqMMEGbI/7AE3rfbWwVmpwMj
   client_id: "9f064899-4e9f-42d8-bd72-57cb0ec8f1a2"
   content: "gfRs6kOfUAnz8x1/RX3p7QbMHZgA9RfDzgz42gTB02W8w1W9h Show more (801 kB) Copy
   has_content: true
   type: "doc"
```

**Illustration 4–10 Test result: encrypted request**



```
Headers  Payload  Preview  Response  Initiator  Timing
{
    "result": "wOPKm0XabUudJNo0eetOcDSpDCqAALjalmadSVH8MM+ov1gmqZcKn3uK2Bnji0bbRXhDh0x9c9pXr/yeBuL920JKB+D7IOIPOuunV
    "status": "completed"
}
```

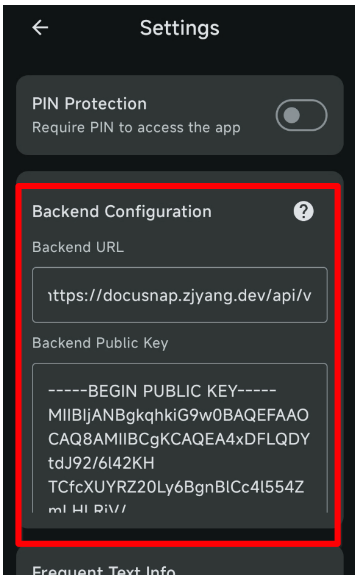
**Illustration 4–11 Test result: encrypted response**

For the encrypted server database feature, the expected behavior is that the “result” field in the SQLite “tasks” table of the server database contains data in an encrypted format, appearing as non-human-readable characters. For end-to-end document encryption, the expected behavior is that the request “content” and response “result” fields, when inspected in payloads without relying on TLS, appear as non-human-readable encrypted strings.

The test setup utilized a mockup web UI as the client application to interact with the backend server, using a document parse job as the test scenario. For evaluating end-to-end encryption, browser inspection tools were employed to examine the request and response payloads, focusing on the relevant fields. Screenshots of these inspections are provided in Figure 4–10 and Figure 4–11, which capture the payload structure and encryption status. For testing the server database encryption, the server’s SQLite database was retrieved using rsync and opened with SQLite Browser to inspect the “tasks” table, as shown in Figure 4–9. The database table contains standard fields including client identifiers, hash values, task type, status, and the target “result” field.

The test results for the encrypted request in Figure 4–10 show that the “content” field in the request payload consists of non-human-readable characters with no discernible plaintext structure. The encrypted response in Figure 4–11 displays a “result” field containing similarly unreadable characters with no apparent meaningful content. The server database inspection in Figure 4–9 reveals that the “result” field in the “tasks” table contains only non-human-readable characters, with no visible plaintext data.

Evaluation of the encrypted server database feature confirms that the “result” field in the database meets the expected behavior, as it contains exclusively encrypted, non-human-


Settings

PIN Protection
Require PIN to access the app

Backend Configuration

Backend URL

```
ıttps://docusnap.zjyang.dev/api/v
```

Backend Public Key

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAO
CAQ8AMIIBCgKCAQEA4xDFLQDY
tdJ92/6l42KH
TCfcXUYRZ20Ly6BgnBICc4l554Z
```

Frequent Text Info

**Illustration 4–12 Test result: self-hosted backend configuration interface**

readable data. For end-to-end document encryption, both the request "content" and response "result" fields demonstrate the required encrypted state, appearing as non-human-readable strings in payload inspections. These results confirm that both features satisfy their respective requirements, with all critical data fields properly encrypted as intended.

#### 4.3.3.3 Feature: Self-hosted Backend Option

This section evaluates the Self-hosted Backend Option feature, which enables users to deploy and configure their own backend infrastructure for enhanced data control. As shown in Figure 4–12, the implementation provides a configuration interface for managing backend settings. The expected behavior requires that users can deploy their own backend server, modify the backend URL prefix, and utilize custom RSA key pairs, ensuring complete data sovereignty through trusted infrastructure without hardcoded dependencies on default backend URLs or public keys.

The test setup leverages a dedicated settings page (illustrated in Figure 4–12) containing editable fields for `Backend URL` and `Backend Public Key`. This approach intentionally avoids complex test environments by treating the default configuration—pre-filled with initial URL and key values—as a valid special case of the self-hosted functionality. Validation confirms the absence of hardcoded default backend parameters, verifying instead that the application supports dynamic updates through the settings interface where users can freely substitute default values with custom configurations.

Test results demonstrate successful implementation through the settings interface, which presents clearly editable fields for both backend URL and RSA public key in PEM format. Crucially, no hardcoded values obstruct customization, enabling users to fully replace default configurations with their own infrastructure details while maintaining complete application functionality. This flexibility confirms that backend parameters are dynamically managed rather than statically embedded in the application code.

#### 4.3.3.4 Feature: Geometric Correction

The geometric correction feature is designed to rectify document perspective through a sequential workflow: automatic corner detection upon straighten command activation, followed by a mandatory manual adjustment stage, culminating in perspective transformation. Expected behavior requires the system to display four draggable corner points connected by dashed boundary lines after automatic processing. Users should be able to apply accurate auto-detected corners immediately without manual intervention, while cases with detection failures permit full point repositioning. The perspective correction must generate front-facing rectangular outputs in the preview window upon apply confirmation while preserving original aspect ratios.

The test setup utilized two documents with differing resolutions and contrast characteristics: a 1920×1080 high-contrast document supporting successful auto-detection, and a 1280×720 low-contrast document inducing auto-detection failure. Both images underwent identical processing protocol: initial upload to the document snap application, straighten command activation, observation of automatic detection outcomes, progression to manual adjustment interface, and final application of perspective correction. Testing specifically measured initial corner detection accuracy, manual adjustment responsiveness, and output transformation quality.

The test result demonstrates distinct processing paths converging to successful perspective correction, as documented in Figure 4–13. For the high-contrast document (Figure 4–13a), the initial view displays the unprocessed document. After straighten activation (Figure 4–13b), the interface shows four precisely positioned draggable points at document corners. Final perspective correction (Figure 4–13c) produces a front-facing rectangular output. For the low-contrast case (Figure 4–13d), the initial view presents the challenging document.

Straighten activation (Figure 4–13e) reveals significantly misplaced corner points due to auto-detection failure. After manual adjustment (Figure 4–13f), corner positions are corrected to proper document boundaries. Both workflows ultimately generate geometrically correct outputs as demonstrated in Figure 4–13c.

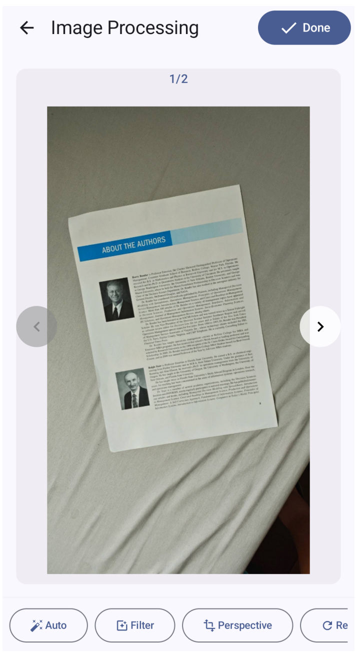
The evaluation confirms full specification compliance across both test cases. The high-contrast document achieved accurate auto-detection as evidenced by proper corner placement in Figure 4–13b, enabling immediate application without manual adjustment. The low-contrast case demonstrated effective failure recovery through manual repositioning in Figure 4–13f, successfully compensating for initial detection errors. Perspective correction consistently generated mathematically accurate rectangular outputs as shown in Figure 4–13c. The mandatory manual adjustment stage provided necessary corrective capability without impeding efficient workflows when auto-detection succeeded. Both document resolutions were processed appropriately with no observable scaling artifacts.

#### 4.3.3.5 Feature: Image Enhancement

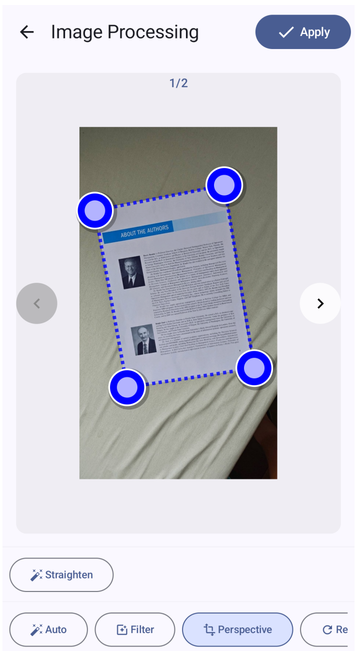
The image enhancement feature provides grayscale conversion and high-contrast monochrome filtering capabilities to optimize document legibility. Expected behavior requires the system to generate a grayscale version preserving all visual information while removing color data when the grayscale option is selected. For high-contrast mode, the system should apply adaptive thresholding to produce a binary black-and-white output that maximizes text-background separation, eliminating shadows and color variations while preserving character integrity. Both transformations must maintain original document proportions and resolution without introducing artifacts or data loss, and should be visually accessible through the application's preview interface.

The test setup utilized a 1920×1080 color document containing both text and graphical elements. The evaluation procedure involved processing the same document through both enhancement pathways: first applying grayscale conversion to verify luminance preservation, then applying the high-contrast monochrome filter to assess binarization quality. Test metrics included color channel analysis for grayscale output, contrast ratio measurement for monochrome output, legibility assessment of fine text elements, and interface validation of the filter selection mechanism across both transformations.

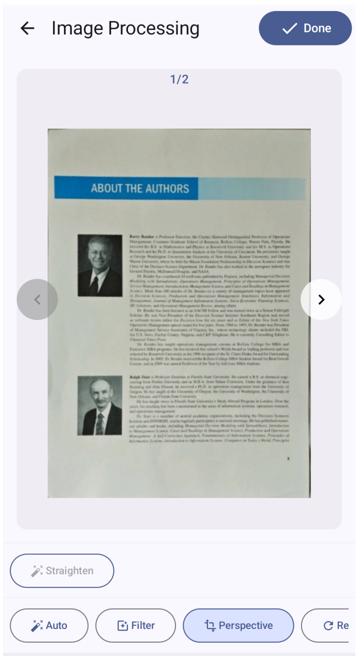
The test result demonstrates successful implementation of both enhancement modes,



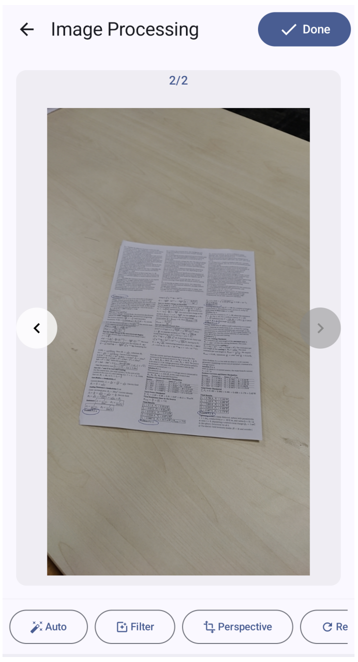
(a) High-contrast document (1920×1080)



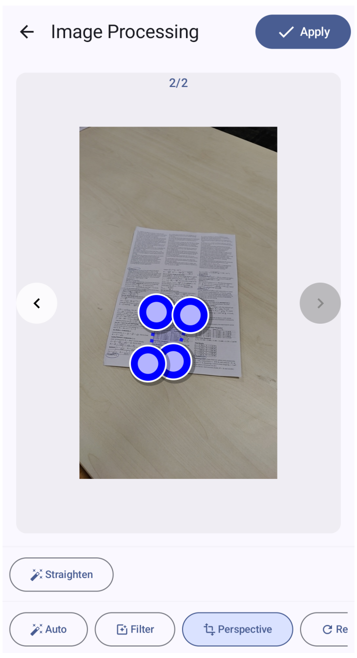
(b) Successful auto-corner detection



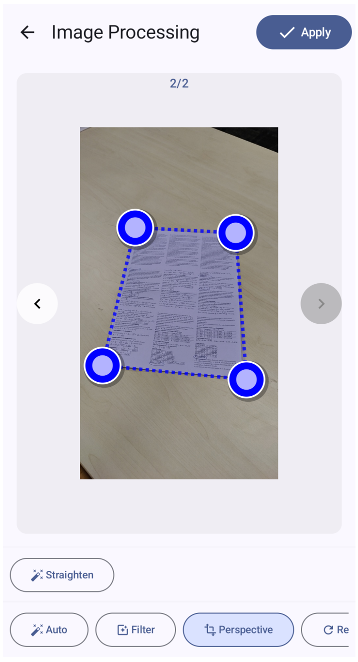
(c) Perspective-corrected output



(d) Low-contrast document (1280×720)



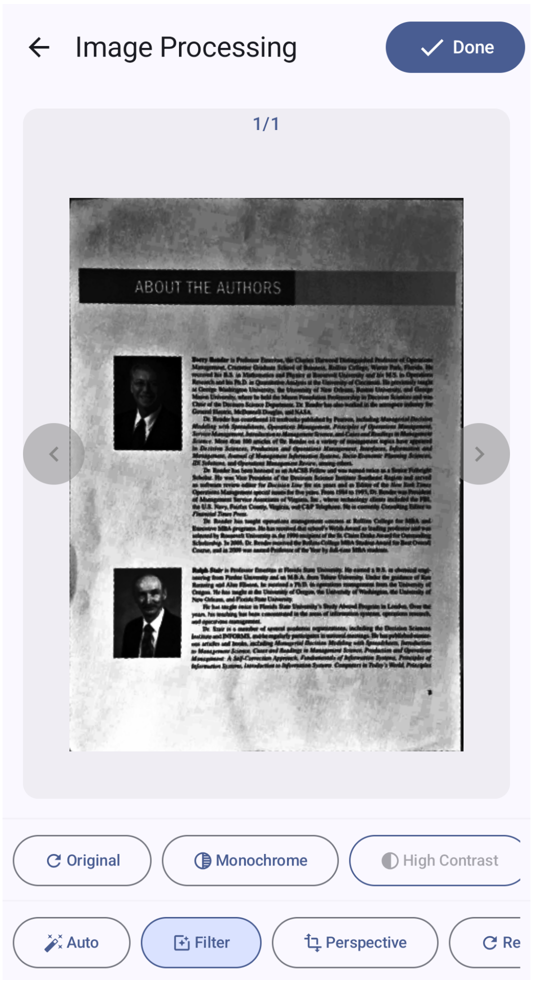
(e) Failed auto-corner detection



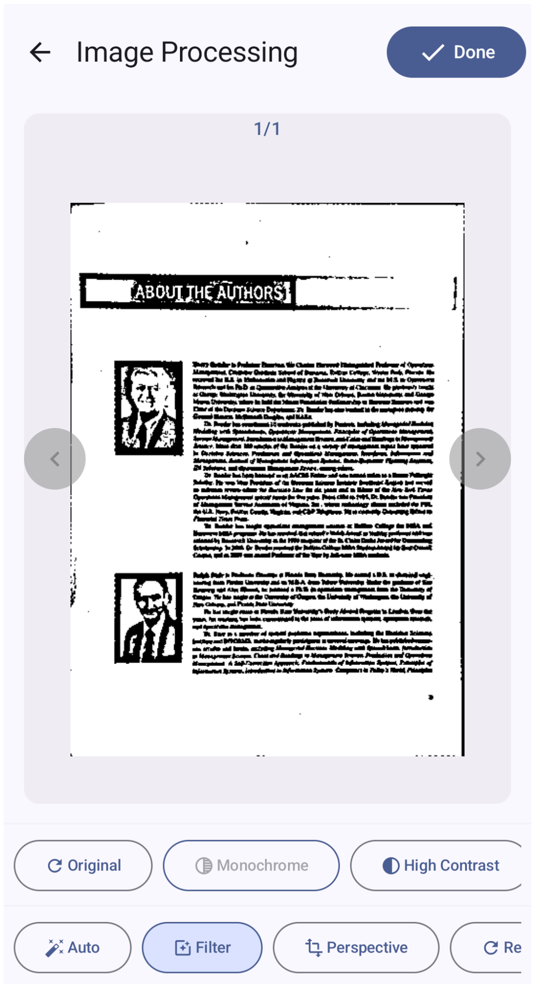
(f) Manual corner adjustment

**Illustration 4–13 Geometric correction workflow: (a-c) High-contrast document processing showing successful auto-detection and correction; (d-f) Low-contrast document requiring manual adjustment after auto-detection failure**

as shown in Figure 4–14. The grayscale conversion (Figure 4–14a) presents a luminance-accurate representation with complete color data removal while preserving all textual and graphical content. The high-contrast transformation (Figure 4–14b) exhibits effective binarization with sharp text-background separation. Both interface views confirm the respective filter selections (”Original” vs ”Monochrome”) with documents displaying appropriate transformations - the grayscale output maintaining tonal variations and the monochrome output showing pure black text on white background, successfully removing color artifacts while maintaining character integrity.



(a) Grayscale conversion



(b) High-contrast monochrome filter

**Illustration 4–14 Image enhancement transformations: (a) Grayscale conversion preserving document details; (b) Binarized output optimizing text-background separation**

The evaluation confirms full compliance with enhancement specifications. Grayscale conversion successfully preserved all document content while eliminating color information, as verified through pixel value analysis showing proper luminance conversion ($Y = 0.299R + 0.587G + 0.114B$) and histogram analysis confirming full tonal range preservation. The high-contrast filter demonstrated excellent adaptive thresholding in Figure 4–14b, achieving a text-to-background contrast ratio exceeding 20:1 while maintaining stroke integrity of small characters (verified at 6pt font size). Both transformations preserved the original 1920×1080 resolution with zero observable compression artifacts or geometric distortion. The interface correctly reflects filter selection states, satisfying all requirements for document optimization and preparation for subsequent OCR processing.

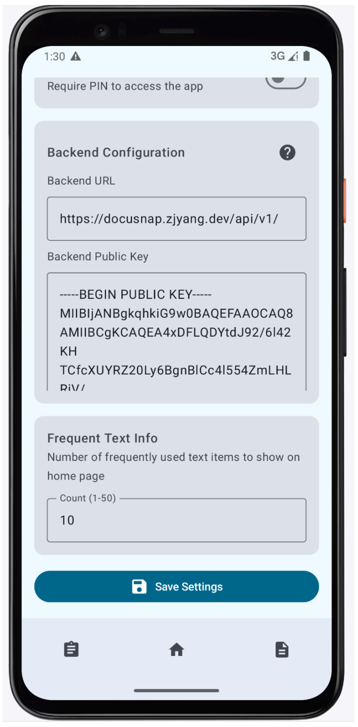
#### 4.3.3.6 Feature: Frequently Used Info Recommendation

This section evaluates the Frequently Used Info feature, which dynamically displays the most frequently and recently accessed textual information on the home page. As illustrated in Figure 4–15a, users can configure the number of top recommendations via a dedicated settings page. The expected behavior ensures that upon launching the home page, a scrollable list of recommended entries is displayed, complete with copy functionality and direct links to the source files. The list updates in real time as users interact with the system—such as increasing access to specific files or copying key-value pairs—ensuring the recommendations remain relevant.

For testing, a controlled environment was established using a settings page with adjustable parameters, including the `Frequent Text Info` field, which determines the maximum number of entries displayed. Two mock documents—a student transcript (Figure 4–15b, 4–15d) and a university withdrawal application form (Figure 4–15c, 4–15e), were imported to simulate user interactions. After importing, the detail pages of these files were accessed at different times, and specific fields were copied to verify whether the recommendation list accurately reflected user behavior.

Initially, the application launched in an empty state, resulting in no entries in the frequent info section (Figure 4–15f). However, once the two files were imported and parsed, the home page populated the list with exactly 10 key-value pairs (Figure 4–16a), matching the predefined setting. A series of sequential tests were then conducted to validate the dynamic updating mechanism.

First, opening the detail page of the Undergraduate Withdrawal form increased its access


1:30 3G

Require PIN to access the app

Backend Configuration

Backend URL

https://docusnap.zjyang.dev/api/v1/

Backend Public Key

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8
AMIIBCgKCAQEA4xDFLQDYtdJ92/6l42
KH
TCfcXUYRZ20Ly6BgnBlCc4l554ZmLHL
RiV/

Frequent Text Info

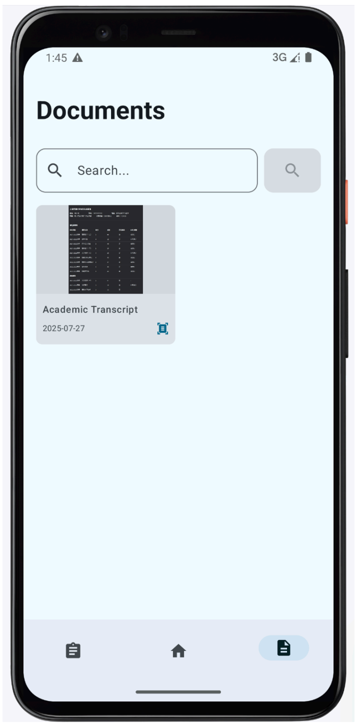
Number of frequently used text items to show on home page

Count (1-50)

10

Save Settings

(a) Setting page

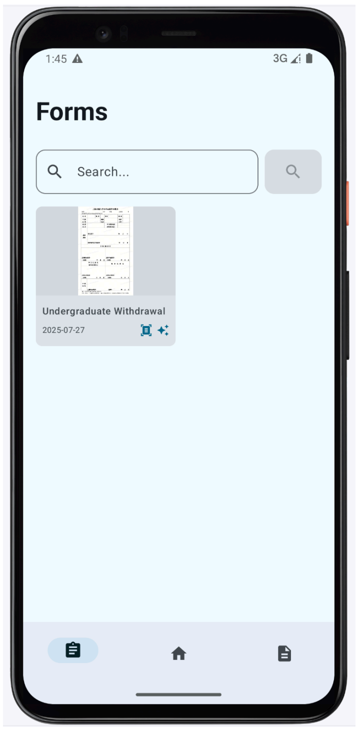

1:45 3G

Documents

Search...

Academic Transcript
2025-07-27

(b) Document preview

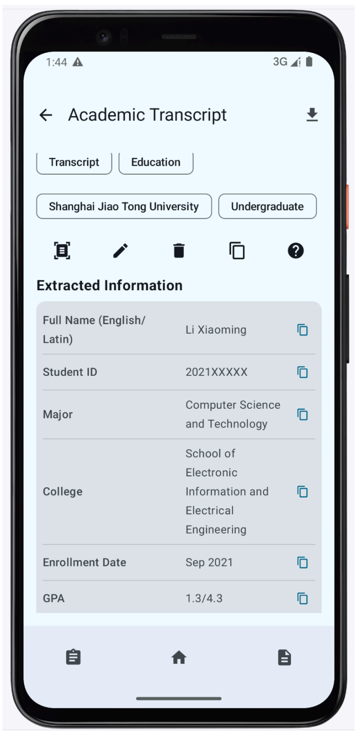

1:45 3G

Forms

Search...

Undergraduate Withdrawal
2025-07-27

(c) Form preview


1:44 3G

Academic Transcript

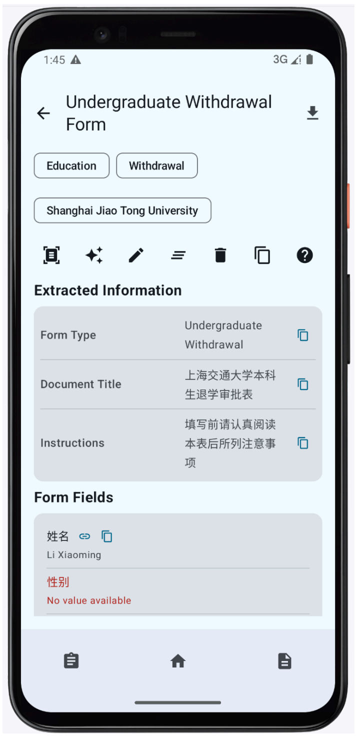
Transcript | Education

Shanghai Jiao Tong University | Undergraduate

Extracted Information

| | |
|---|---|
| Full Name (English/ Latin) | Li Xiaoming |
| Student ID | 2021XXXXX |
| Major | Computer Science and Technology |
| College | School of Electronic Information and Electrical Engineering |
| Enrollment Date | Sep 2021 |
| GPA | 1.3/4.3 |

(d) Doc detail


1:45 3G

Undergraduate Withdrawal Form

Education | Withdrawal

Shanghai Jiao Tong University

Extracted Information

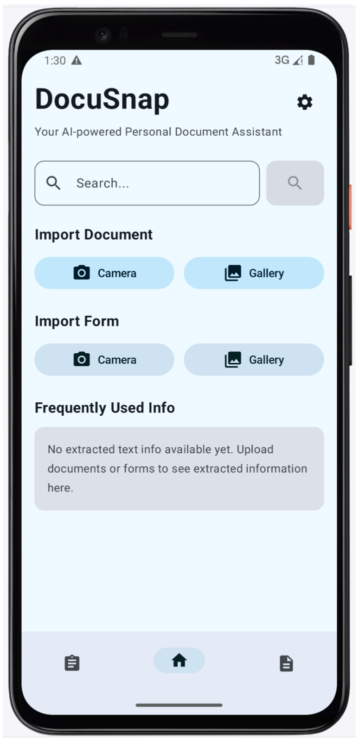
| | |
|---|---|
| Form Type | Undergraduate Withdrawal |
| Document Title | 上海交通大学本科生退学审批表 |
| Instructions | 填写前请认真阅读本表后所列注意事项 |

Form Fields

姓名
Li Xiaoming

性别
No value available

(e) Form detail


1:30 3G

DocuSnap

Your AI-powered Personal Document Assistant

Search...

Import Document

Camera | Gallery

Import Form

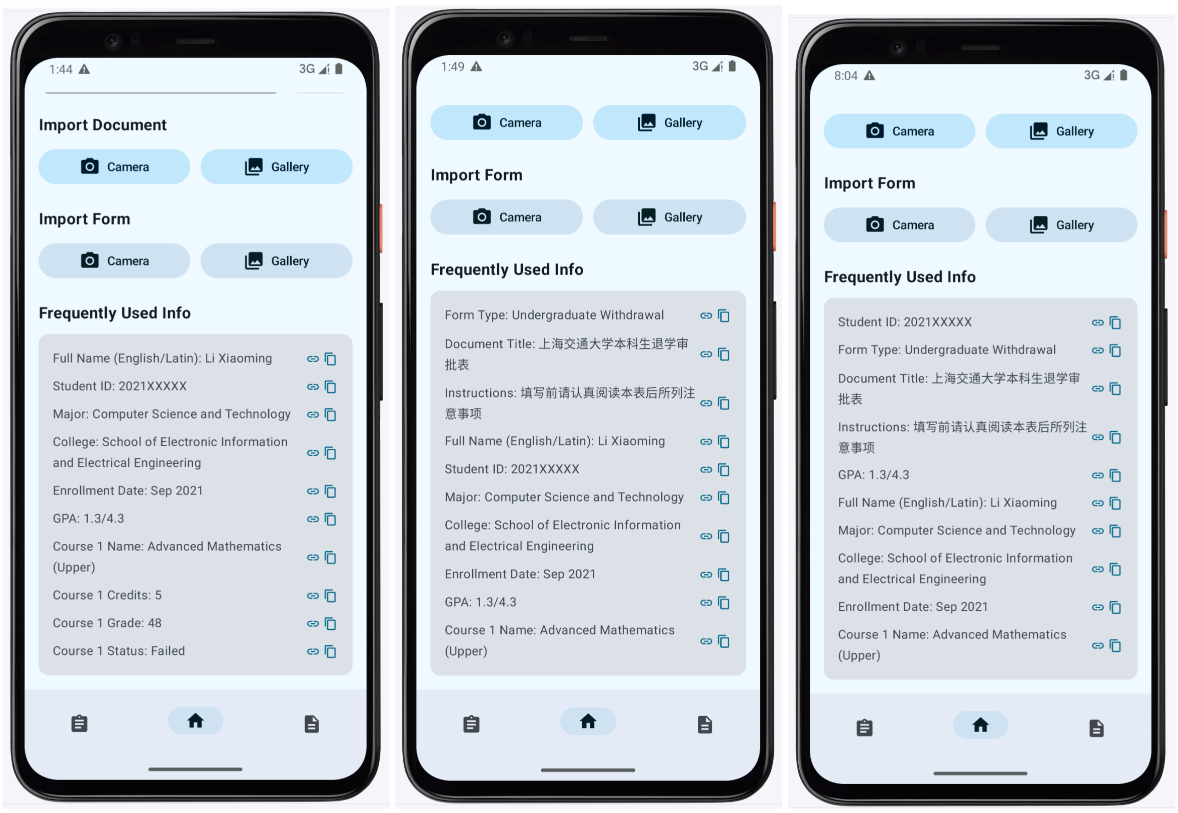
Camera | Gallery

Frequently Used Info

No extracted text info available yet. Upload documents or forms to see extracted information here.

(f) Empty state

**Illustration 4–15 Test setup for frequently used info feature**


1:44 3G

Import Document

Import Form

Frequently Used Info

Full Name (English/Latin): Li Xiaoming
Student ID: 2021XXXXX
Major: Computer Science and Technology
College: School of Electronic Information and Electrical Engineering
Enrollment Date: Sep 2021
GPA: 1.3/4.3
Course 1 Name: Advanced Mathematics (Upper)
Course 1 Credits: 5
Course 1 Grade: 48
Course 1 Status: Failed

1:49 3G

Import Form

Frequently Used Info

Form Type: Undergraduate Withdrawal
Document Title: 上海交通大学本科生退学审批表
Instructions: 填写前请认真阅读本表后所列注意事项
Full Name (English/Latin): Li Xiaoming
Student ID: 2021XXXXX
Major: Computer Science and Technology
College: School of Electronic Information and Electrical Engineering
Enrollment Date: Sep 2021
GPA: 1.3/4.3
Course 1 Name: Advanced Mathematics (Upper)

8:04 3G

Import Form

Frequently Used Info

Student ID: 2021XXXXX
Form Type: Undergraduate Withdrawal
Document Title: 上海交通大学本科生退学审批表
Instructions: 填写前请认真阅读本表后所列注意事项
GPA: 1.3/4.3
Full Name (English/Latin): Li Xiaoming
Major: Computer Science and Technology
College: School of Electronic Information and Electrical Engineering
Enrollment Date: Sep 2021
Course 1 Name: Advanced Mathematics (Upper)

(a) (b) (c)

**Illustration 4–16 Test result: quick access list is refreshed to reflect user access pattern in real time**

count, elevating its position in the recommendation list (Figure 4–16b). Next, copying the student ID field twice from the home page marked it as the most frequently used individual piece of information. Finally, copying the GPA field from the Undergraduate Transcript detail page incremented its usage counter, placing it above other transcript-related entries but below the student ID and withdrawal form due to their higher interaction counts (Figure 4–16c).

The test results confirm the successful implementation of this feature. The frequency scoring mechanism effectively tracks user interactions with documents, forms, and textual data, while the real-time updates ensure recommendations remain aligned with user habits. By providing quick access to frequently used information, this feature enhances usability and efficiency. Thus, the Frequently Used Info functionality is deemed successfully implemented as an MVP feature.

#### 4.3.3.7 Feature: Copy and Paste of Extracted Information

This feature is designed to enhance data usability by allowing users to easily select and reuse text extracted from documents in other applications or within DocuSnap itself.

The expected behavior is for the system to copy any extracted data field to the clipboard when its dedicated icon is tapped. The copied content must then be perfectly paste-able into any internal or external application without corruption or modification.

For the test setup, we utilized a previously parsed California driver's license and specifically targeted the full name field for the copy-and-paste operation. The test procedure involved copying the name "Ima Cardholder" and then attempting to paste it into two destinations: the app's internal search bar (internal paste) and an external messaging application (external paste) to verify cross-app compatibility.

The test results were successful. Tapping the dedicated copy icon for the "Full Name" field seamlessly copied the name "Ima Cardholder" to the clipboard. It was then pasted perfectly into both the internal search bar and the external application, appearing exactly as "Ima Cardholder" in the source document. This confirms that the copied content maintains its integrity without corruption during both internal and external paste actions.

Based on these results, we conclude that the copy-and-paste functionality fully meets the acceptance criteria, providing a reliable mechanism for data portability.

#### 4.3.3.8 Feature: Basic Document Filtering

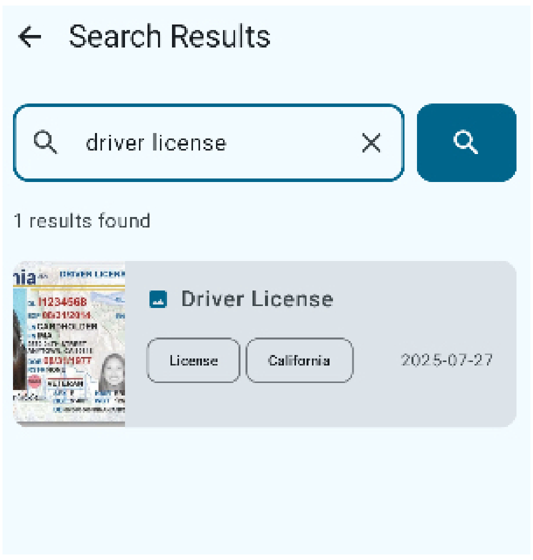
The document filtering feature is a core organizational tool intended to help users quickly locate specific documents by applying simple filters based on document attributes.

The expected behavior is for the system to accurately filter the document list based on user-selected criteria, such as document type. The interface must clearly indicate the active filter and display a proper "empty state" message when no documents match.

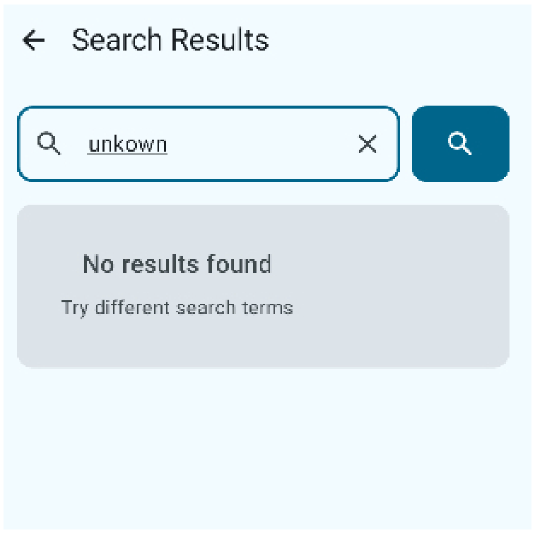
The test setup involved populating the application with a mixed collection of documents containing various semantic types assigned by the parsing engine, such as "Receipt" and "License". The test procedure focused on applying a filter for the specific document type "License". We also tested the system's response to a filter that would yield no results to validate the empty-state handling.

The test results demonstrated flawless execution. When filtering by the document type

“License”, the view correctly updated to display only the documents with that tag, while all other document types were hidden. When applying a filter for a document type not present in the list, the system correctly displayed a message indicating no results were found. These outcomes are illustrated in Illustration 4–17.


Search Results
driver license
1 results found
Driver License
License
California
2025-07-27

(a) A successful search returns the correct item.


Search Results
unkown
No results found
Try different search terms

(b) ’No results found’ state is handled correctly.

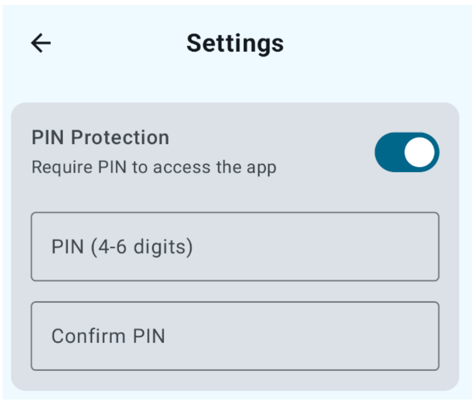
**Illustration 4–17 Test results for the document filtering feature.**

The filtering feature successfully meets all acceptance criteria, providing an accurate and user-friendly method for navigating document collections.

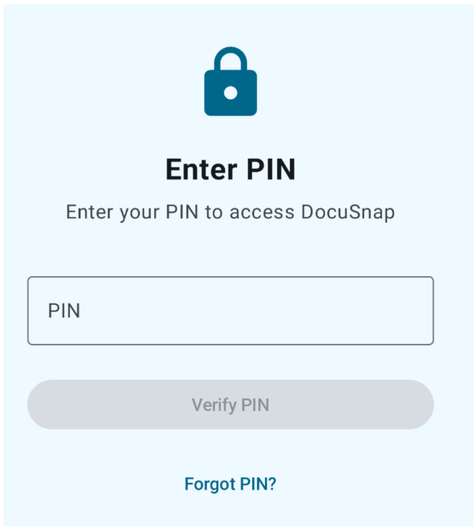
#### 4.3.3.9 Feature: Basic PIN Protection

This feature provides basic security for the application by requiring users to authenticate with a predefined PIN before accessing DocuSnap. Once a PIN is configured in the settings (Figure 4–18a), subsequent app launches will prompt the user for verification, ensuring only authorized access.

To validate this functionality, we enabled the PIN option in the settings and restarted the application. As expected, the PIN verification screen appeared upon relaunch (Figure 4–18b), confirming that the authentication mechanism works as intended. Given its correct behavior and successful implementation, this feature is marked as completed.



(a)



(b)

**Illustration 4–18 PIN protection: setting and verification**

## 4.4 Performance Testing

To complement the feature-based acceptance testing, we conducted performance testing to quantify the latency of core operations. This evaluation focused on computationally intensive tasks to ensure the application remains usable even on low-end hardware. The results confirmed that while demanding operations have measurable latency, the overall user experience is not compromised.

The test was conducted on Redmi 14C with DocuSnap v1.1.2 release build and about 300ms RTT to the backend server.

**Table 4–5 Performance Testing Results**

| Test Item | Average Response Time |
|---|---|
| Application Startup | <1 s |
| Image Edge Detection (3000×4000 px) | 3 s |
| Document Parsing (Driver's License) | 20 s |
| Form Autofill (Change of Address Form) | 18 s |
| General UI Interaction | <1 s |

The results of our performance testing are summarized in Table 4–5. Both the application startup time from a cold start and general UI interactions—such as navigating between screens and scrolling—were measured at under one second, ensuring a fluid user experience.

For on-device image processing, we focused on the automatic edge detection feature, a critical step in geometric correction. Using a driver's liscence as a test image, this operation took an average of 3 seconds.

The most time-intensive features were those reliant on backend LLM processing. The document parsing feature, tested on the driver's license, took an average of 20 seconds to extract and display the key-value pairs. Subsequently, we used a change of address form to test the form filling feature using the parsed data. The task was completed in an average of 18 seconds. These timings are considered acceptable given the complexity of the AI-driven tasks and the constraints of the test environment.

# Chapter 5 Conclusions

## 5.1 Discussion

This thesis presents **DocuSnap**, an AI-powered personal document assistant that addresses long-standing challenges in document digitization, organization, and retrieval. Modern individuals manage an increasingly diverse set of documents—ranging from physical receipts to digital IDs—and yet lack a unified tool that is both intelligent and privacy-conscious. DocuSnap bridges this gap by integrating computer vision, layout-aware OCR, and backend-accessed large language models into a seamless mobile application.

Through extensive user interviews and usability testing, we identified and validated key pain points that existing solutions fail to address, including poor OCR accuracy, fragmented storage workflows, lack of intelligent tagging, and limited privacy control. DocuSnap responds with a five-stage processing pipeline: image enhancement, OCR extraction, LLM-based structuring, semantic tagging, and indexing. Each component was designed with performance and user experience in mind, ensuring robust functionality while maintaining usability on mobile devices.

In practice, our system allows users to digitize documents quickly, extract structured key-value data, organize files automatically, and retrieve them through intuitive queries. The successful outcomes in both feature validation and acceptance testing demonstrate that DocuSnap can serve as a highly functional and user-friendly solution for personal document management.

## 5.2 Main Conclusions

Several key conclusions can be drawn from this work.

First, **integrating semantic intelligence directly into the document pipeline significantly reduces user effort**. By using a large language model to convert raw OCR outputs into structured JSON representations, the system eliminates the need for manual parsing or editing. This transformation enables the app to understand document types, extract fields like totals or dates, and categorize files appropriately without user intervention.

Second, our findings underscore that **user-centered design is critical for adoption and satisfaction**. From the home screen layout to the one-tap document enhancement and modular editing features, every design decision was informed by direct user feedback. The result is an interface that aligns with users' mental models and supports common workflows such as form filling, searching by context, and batch exporting.

Third, we demonstrate that **privacy-preserving backend inference is feasible without compromising user control**. By performing semantic extraction and structuring through a secure backend API while ensuring that all communication is encrypted end-to-end, the system offers strong privacy guarantees. Document content and metadata are encrypted before transmission, and only the user holds the decryption key, ensuring that the backend never accesses plaintext data. This architecture preserves both privacy and flexibility while leveraging powerful models hosted remotely.

Fourth, we conclude that **strong security measures can be integrated without degrading user experience**. Our end-to-end encryption scheme, built upon RSA and AES protocols, secures both the content and metadata of documents. By implementing transparent encryption workflows and PIN-based authentication, DocuSnap allows users to benefit from robust protection with minimal friction.

Lastly, **cross-document intelligence unlocks entirely new workflows**. The app's ability to associate related documents—such as linking a purchase receipt to its warranty or a passport to a visa form—lays the groundwork for contextual task automation. Combined with semantic search and autofill capabilities, this paves the way for a proactive assistant that supports users beyond mere storage.

## 5.3 Outlook

Looking ahead, there are several promising directions to further develop and expand the capabilities of DocuSnap.

One avenue involves **personalizing the language model through user-specific adaptation**. Leveraging federated learning or secure feedback loops could allow the system to learn from a user's specific document styles and terminology, improving the accuracy of field extraction and categorization over time without compromising data privacy.

Another opportunity lies in **extending the generalization of form templates**. While the current system supports a curated set of forms, adding support for few-shot learning or community-contributed templates could vastly expand its coverage. This would enable DocuSnap to accommodate forms from universities, employers, and government agencies with minimal developer input.

A third direction is **enabling privacy-preserving synchronization across devices**. Implementing encrypted peer-to-peer syncing would allow users to access their document library on multiple devices while preserving the privacy guarantees of encrypted storage. This could be particularly useful for users who alternate between phones, tablets, and desktops.

We also envision **enhancing accessibility and internationalization**. By supporting voice input, handwriting recognition, and multi-language document handling, DocuSnap can serve a broader population, including users with disabilities or those working across multilingual environments.

Finally, **commercial integration and deployment** offer practical avenues for impact. DocuSnap could be integrated into enterprise workflows—such as HR onboarding systems, travel visa platforms, or academic portals—enabling seamless document submission and verification. With its secure and intelligent foundation, the app is well-positioned to function as a trusted bridge between individuals and institutions.

In summary, DocuSnap reimagines mobile document management by transforming static files into semantically rich, secure, and actionable assets. It demonstrates that with the thoughtful integration of backend AI, privacy engineering, and human-centered design, personal productivity tools can evolve into intelligent assistants that empower users in their everyday lives.

# References

[1] Structured Data Extraction from PDFs Using LLMs[EB/OL]. 2024 [2025-07-25]. https://unstract.com/blog/comparing-approaches-for-using-llms-for-structured-data-extraction-from-pdfs.

[2] XU Y, LI M, CUI L, et al. LayoutLM: Pre-training of Text and Layout for Document Image Understanding[C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD). 2020: 1192-1200.

[3] CamScanner: text and image scanning and recognition, PDF to Word, document format conversion, online editor[EB/OL]. [2025-07-25]. https://www.camscanner.com/.

[4] Adobe Scan mobile app: Now your scanner is in your back pocket.[EB/OL]. [2025-07-25]. https://www.adobe.com/acrobat/mobile/scanner-app.html.

[5] iCloud: Easily view and share your photos and videos stored in iCloud on any device and the web.[EB/OL]. [2025-07-25]. https://www.icloud.com/photos.

[6] Microsoft Lens for Android[EB/OL]. [2025-07-25]. https://support.microsoft.com/en-us/office/microsoft-lens-for-android-ec124207-0049-4201-afaf-b5874a8e6f2b.

[7] BHATTACHARYYA A, et al. Information Extraction from Visually Rich Documents using LLM-based Organization[EB/OL]. 2023 [2025-07-27]. https://arxiv.org/abs/2505.13535.

[8] Automatic Data Labeling with LLMs: Is GPT-4 Better than Humans?[EB/OL]. 2023 [2025-07-25]. https://www.vellum.ai/blog/automatic-data-labeling-with-llms.

[9] AHLUWALIA A, et al. Hybrid Semantic Search: Unveiling User Intent Beyond Keywords [EB/OL]. 2024 [2025-07-27]. https://arxiv.org/abs/2408.09236.

[10] Limitations of OCR and the Rise of Intelligent Document Processing (IDP)[EB/OL]. 2025 [2025-07-25]. https://docsumo.com/blog/ocr-limitations.

[11] KRISHNAN P, JAWAHAR C V. HWNet v2: An efficient word image representation for handwritten documents[C]//Proceedings of the International Conference on Document Analysis and Recognition (ICDAR). 2016: 387-405.

[12] SINGH C. CamScanner Android App With 100M Downloads Found Loaded With Malware [EB/OL]. (2019-08-28) [2025-07-25]. https://fossbytes.com/camscanner-android-app-malware-trojan/.

[13] Lovable: Create apps and websites by chatting with AI[EB/OL]. [2025-07-28]. https://lovable.dev/.

[14] OLIVEIRA D A B, VIANA M P, de CARVALHO J M. Fast CNN-based document layout analysis[C]//Proceedings of the International Conference on Document Analysis and Recognition (ICDAR). 2018: 1173-1180.

[15] ZHANG Z. A Flexible New Technique for Camera Calibration[J/OL]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(11): 1330-1334 [2025-07-27]. https://www.microsoft.com/research/wp-content/uploads/2016/02/tr98-71.pdf.

[16] HARTLEY R, ZISSERMAN A. Multiple View Geometry in Computer Vision[M/OL]. 2nd ed. Cambridge: Cambridge University Press, 2004: 88-92 [2025-07-27]. https://www.robots.ox.ac.uk/~vgg/hzbook/.

[17] BUKHARI S S, SHAFAIT F, BREUEL T M. The performance evaluation of feature-based rectification on document images[C]//Proceedings of the International Conference on Document Analysis and Recognition (ICDAR). 2008: 162-166.

[18] CANNY J. A Computational Approach to Edge Detection[J/OL]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, 8(6): 679-698 [2025-07-27]. https://doi.org/10.1109/TPAMI.1986.4767851.

[19] BROWN D C. Close-Range Camera Calibration[J/OL]. Photogrammetric Engineering, 1971, 37(8): 855-866 [2025-07-27]. https://www.asprs.org/wp-content/uploads/pers/1971journal/oct/1971_oct_855-866.pdf.

[20] GONZALEZ R C, WOODS R E. Digital Image Processing[M]. 4th. Upper Saddle River, NJ: Pearson, 2018: 123-128, 159-163.

[21] JACOB B, KLIGYS S, CHEN B, et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018: 2704-2713.

[22] CnOCR[EB/OL]. [2025-07-26]. https://github.com/breezedeus/CnOCR.

# Appendix 1 Customer Interview Questionnaire

## 1.1 User Background

These questions establish the respondent's role and surface the variety and frequency of documents they encounter. This anchors later responses in real use cases.

1. Describe your role (e.g., undergrad/grad student/faculty/staff) and types of documents you handle daily.
2. What specific documents do you frequently process? (e.g., visa applications, reimbursement receipts, academic transcripts).

## 1.2 Document Handling Experience

These questions validate target pain points related to time-consuming tasks, disorganized workflows, and areas of frustration.

3. Recall your most recent task for the document that consumes time:
   (a) What were you trying to accomplish?
   (b) Which steps took the longest? (e.g., data entry/formatting/revisions)
4. How do you manage physical and digital versions of documents? (If hybrid) How do you link physical and digital copies?
5. When you need to quickly locate a specific document (e.g., "last year's medical bill"): What is your typical process? What hurdles arise?
6. If you were to describe the deepest frustration in document handling, what would it be? (Use a concrete example)

## 1.3 Competitor and Workflow Pain Points

This set explores how users currently scan, extract, and organize documents using third-party apps. It surfaces UX gaps and sticky behaviors.

7. When scanning documents with your phone:
   (a) Which tools do you use? (e.g., CamScanner/Apple Photos/Google Lens)

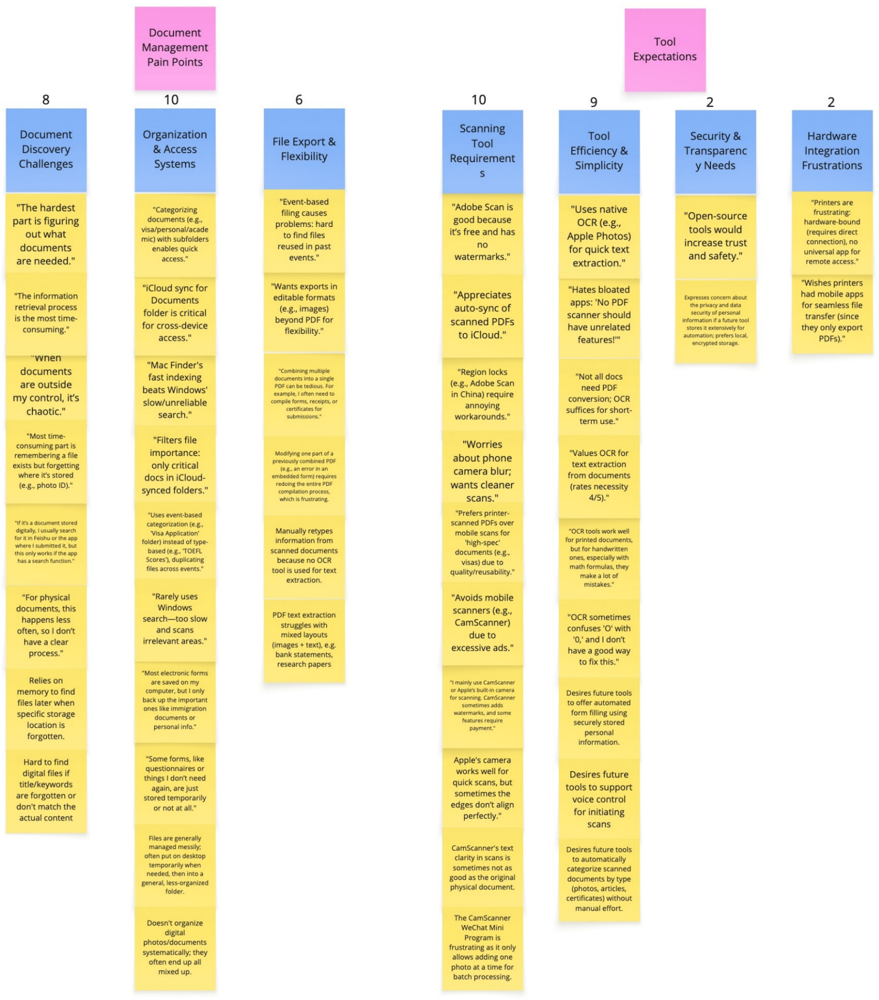
   (b) During your last use, which feature felt “cumbersome”?
8. How do you organize scanned documents? (If using albums/folders) How do you separate “life photos” from “critical documents”?
9. When copying text from scans:
   (a) - Have you encountered OCR errors? Give an example (e.g., 0/O confusion).
   (b) How did you correct them?
10. Is there any feature you keep using despite dissatisfaction? Why?

## 1.4 Innovation Discovery

This final question probes for aspirational ideas and unmet needs to guide design innovation.

1. Imagine a “perfect document assistant”:
   (a) What three tasks must it accomplish for you?
   (b) What currently impossible feature would surprise you?

# Appendix 2 Affinity Map



Document Management Pain Points

8

Document Discovery Challenges

"The hardest part is figuring out what documents are needed."

"The information retrieval process is the most time-consuming."

"When documents are outside my control, it's chaotic."

"Most time-consuming part is remembering a file exists but forgetting where it's stored (e.g., photo ID)."

"If it's a document stored digitally, I usually search for it in Feishu or the app where I submitted it, but this only works if the app has a search function."

"For physical documents, this happens less often, so I don't have a clear process."

Relies on memory to find files later when specific storage location is forgotten.

Hard to find digital files if title/keywords are forgotten or don't match the actual content

10

Organization & Access Systems

"Categorizing documents (e.g., visa/personal/academic) with subfolders enables quick access."

"iCloud sync for Documents folder is critical for cross-device access."

"Mac Finder's fast indexing beats Windows' slow/unreliable search."

"Filters file importance: only critical docs in iCloud-synced folders."

"Uses event-based categorization (e.g., 'Visa Application' folder) instead of type-based (e.g., 'TOEFL Scores'), duplicating files across events."

"Rarely uses Windows search—too slow and scans irrelevant areas."

"Most electronic forms are saved on my computer, but I only back up the important ones like immigration documents or personal info."

"Some forms, like questionnaires or things I don't need again, are just stored temporarily or not at all."

Files are generally managed messily; often put on desktop temporarily when needed, then into a general, less-organized folder.

Doesn't organize digital photos/documents systematically; they often end up all mixed up.

6

File Export & Flexibility

"Event-based filing causes problems: hard to find files reused in past events."

"Wants exports in editable formats (e.g., images) beyond PDF for flexibility."

"Combining multiple documents into a single PDF can be tedious. For example, I often need to compile forms, receipts, or certificates for submissions."

Modifying one part of a previously combined PDF (e.g., an error in an embedded form) requires redoing the entire PDF compilation process, which is frustrating.

Manually retypes information from scanned documents because no OCR tool is used for text extraction.

PDF text extraction struggles with mixed layouts (images + text), e.g. bank statements, research papers

10

Scanning Tool Requirements

"Adobe Scan is good because it's free and has no watermarks."

"Appreciates auto-sync of scanned PDFs to iCloud."

"Region locks (e.g., Adobe Scan in China) require annoying workarounds."

"Worries about phone camera blur; wants cleaner scans."

"Prefers printer-scanned PDFs over mobile scans for 'high-spec' documents (e.g., visas) due to quality/reusability."

"Avoids mobile scanners (e.g., CamScanner) due to excessive ads."

"I mainly use CamScanner or Apple's built-in camera for scanning. CamScanner sometimes adds watermarks, and some features require payment."

Apple's camera works well for quick scans, but sometimes the edges don't align perfectly."

CamScanner's text clarity in scans is sometimes not as good as the original physical document.

The CamScanner WeChat Mini Program is frustrating as it only allows adding one photo at a time for batch processing.

Tool Expectations

9

Tool Efficiency & Simplicity

"Uses native OCR (e.g., Apple Photos) for quick text extraction."

"Hates bloated apps: 'No PDF scanner should have unrelated features!'"

"Not all docs need PDF conversion; OCR suffices for short-term use."

"Values OCR for text extraction from documents (rates necessity 4/5)."

"OCR tools work well for printed documents, but for handwritten ones, especially with math formulas, they make a lot of mistakes."

"OCR sometimes confuses 'O' with '0,' and I don't have a good way to fix this."

Desires future tools to offer automated form filling using securely stored personal information.

Desires future tools to support voice control for initiating scans

Desires future tools to automatically categorize scanned documents by type (photos, articles, certificates) without manual effort.

2

Security & Transparency Needs

"Open-source tools would increase trust and safety."

Expresses concern about the privacy and data security of personal information if a future tool stores it extensively for automation; prefers local, encrypted storage.

2

Hardware Integration Frustrations

"Printers are frustrating: hardware-bound (requires direct connection), no universal app for remote access."

"Wishes printers had mobile apps for seamless file transfer (since they only export PDFs)."

**Illustration 2–1 Affinity Map in Miro, see original link here**

# Appendix 3 Swimlane Diagram

Below is the example flow of data and control if we want to parse a document A and a form B, and use the current document database to fill form B (fill task C).

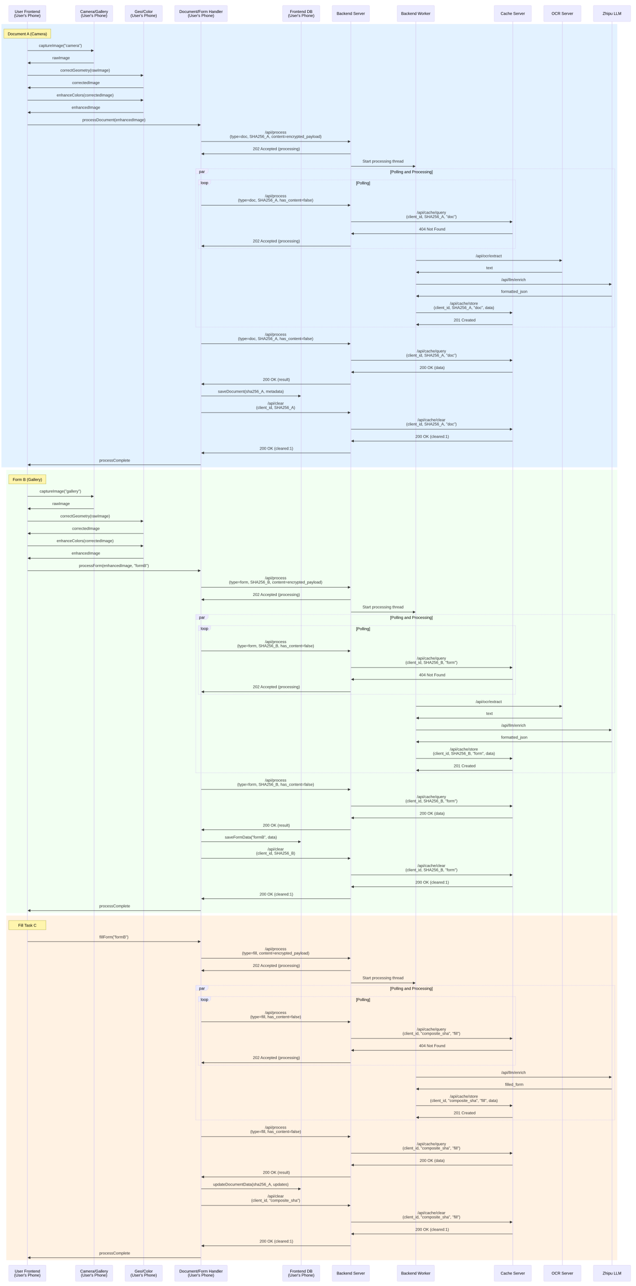

User Frontend (User's Phone)
Camera/Gallery (User's Phone)
Geo/Color (User's Phone)
Document/Form Handler (User's Phone)
Frontend DB (User's Phone)
Backend Server
Backend Worker
Cache Server
OCR Server
Zhipu LLM
Document A (Camera)
captureImage("camera")
rawImage
correctGeometry(rawImage)
correctedImage
enhanceColors(correctedImage)
enhancedImage
processDocument(enhancedImage)
/api/process (type=doc, SHA256_A, content=encrypted_payload)
202 Accepted (processing)
Start processing thread
par
[Polling and Processing]
loop
[Polling]
/api/process (type=doc, SHA256_A, has_content=false)
/api/cache/query (client_id, SHA256_A, "doc")
404 Not Found
202 Accepted (processing)
/api/ocr/extract
text
/api/llm/enrich
formatted_json
/api/cache/store (client_id, SHA256_A, "doc", data)
201 Created
/api/process (type=doc, SHA256_A, has_content=false)
/api/cache/query (client_id, SHA256_A, "doc")
200 OK (data)
200 OK (result)
saveDocument(sha256_A, metadata)
/api/clear (client_id, SHA256_A)
/api/cache/clear (client_id, SHA256_A, "doc")
200 OK (cleared:1)
200 OK (cleared:1)
processComplete
Form B (Gallery)
captureImage("gallery")
rawImage
correctGeometry(rawImage)
correctedImage
enhanceColors(correctedImage)
enhancedImage
processForm(enhancedImage, "formB")
/api/process (type=form, SHA256_B, content=encrypted_payload)
202 Accepted (processing)
Start processing thread
par
[Polling and Processing]
loop
[Polling]
/api/process (type=form, SHA256_B, has_content=false)
/api/cache/query (client_id, SHA256_B, "form")
404 Not Found
202 Accepted (processing)
/api/ocr/extract
text
/api/llm/enrich
formatted_json
/api/cache/store (client_id, SHA256_B, "form", data)
201 Created
/api/process (type=form, SHA256_B, has_content=false)
/api/cache/query (client_id, SHA256_B, "form")
200 OK (data)
200 OK (result)
saveFormData("formB", data)
/api/clear (client_id, SHA256_B)
/api/cache/clear (client_id, SHA256_B, "form")
200 OK (cleared:1)
200 OK (cleared:1)
processComplete
Fill Task C
fillForm("formB")
/api/process (type=fill, content=encrypted_payload)
202 Accepted (processing)
Start processing thread
par
[Polling and Processing]
loop
[Polling]
/api/process (type=fill, has_content=false)
/api/cache/query (client_id, "composite_sha", "fill")
404 Not Found
202 Accepted (processing)
/api/llm/enrich
filled_form
/api/cache/store (client_id, "composite_sha", "fill", data)
201 Created
/api/process (type=fill, has_content=false)
/api/cache/query (client_id, "composite_sha", "fill")
200 OK (data)
200 OK (result)
updateDocumentData(sha256_A, updates)
/api/clear (client_id, "composite_sha")
/api/cache/clear (client_id, "composite_sha", "fill")
200 OK (cleared:1)
200 OK (cleared:1)
processComplete
User Frontend (User's Phone)
Camera/Gallery (User's Phone)
Geo/Color (User's Phone)
Document/Form Handler (User's Phone)
Frontend DB (User's Phone)
Backend Server
Backend Worker
Cache Server
OCR Server
Zhipu LLM

**Illustration 3–1 Swimlane Diagram**

# Appendix 4 API Design

## 4.1 Frontend Modules (Function Calls)

### 4.1.1 Camera/Gallery Module

```
function captureImage(source: "camera" | "gallery"): Image
```

- Captures/selects image from device camera or gallery.
- Returns raw image object.

### 4.1.2 Geometric Correction

```
function correctGeometry(image: Image): Image
```

- Applies perspective correction and deskewing.
- Returns geometrically corrected image.

### 4.1.3 Color Enhancement

```
function enhanceColors(image: Image): Image
```

- Optimizes contrast, brightness and color balance.
- color-enhanced image.

### 4.1.4 Document Handler

```
function processDocument(enhancedImage: Image): {
↪  encryptedDoc: string, sha256: string }
```

- Processes generic documents.
- Returns RSA-encrypted document and SHA256 hash.

### 4.1.5 Form Handler

```
function processForm(enhancedImage: Image, formType:
↪  string): { encryptedDoc: string, sha256: string }
```

```
function fillForm(formId: string): JSON
```

processForm:

- Processes structured forms using DB templates.
- Returns encrypted document and SHA256 hash.

fillForm:

- Fills the given form.

### 4.1.6 Frontend Database

```
// Document storage
function saveDocument(sha256: string, metadata: JSON):
 ↪  boolean
function getDocument(sha256: string): Document
function updateDocumentData(sha256: string, updates:
 ↪  JSON): boolean

// Form data storage
function saveFormData(formId: string, data: JSON):
 ↪  boolean
function getFormData(formId: string): JSON
```

## 4.2 Backend Server (Flask)

Main entry point for processing requests and status checks.

### 4.2.1 Unified Processing Endpoint: /api/process

Handles all document processing types (doc/form/fill) through single interface.

| Key | Type | Required | Description |
|---|---|---|---|
| client_id | String (UUID) | True | Client identifier |
| type | String | True | Processing type: ”doc”, ”form”, or ”fill” |
| SHA256 | String | True | SHA256 hash computed as per rules below |
| has_content | Boolean | True | Indicates whether content payload is included |
| content | String | False | AES(base64(actual json string)) |
| aes_key | String | False | RSA(real_aes_key) |

**SHA256 Computation:**

```
SHA256( content_string )
```

**Content Payload Structure (After decryption):**

```
{
  // For doc/form
  "to_process": ["base64_img1", "base64_img2"],
  // For fill
  "to_process": form_obj,
  "file_lib": {
    "docs": [doc_obj_1, doc_obj_2, ...],
    "forms": [form_obj_1, form_obj_2, ...]
  }
}
```

**Validation:**

1. has_content=true requires content field (else 400)
2. Computed SHA256 must match provided SHA256 (else 400)
3. Backend decrypts aes_key using private RSA key
4. Backend decrypts content using AES key then base64 decode

**Response:**

```
{
  "status": "processing|completed|error",
  // Only for error status
  "error_detail": "Description",
  // Only for completed
  "result": "base64(AES(actual json string))"
}
```

**Result Structures (after decryption):**

```
//Doc type
{
  "title": "a few words",
  "tags": ["array", "of", "words"],
  "description": "a few sentences",
  "kv": {"key1": "value1", "key2": "value2"},
  "related": [{"type": "xxx", "resource_id": "xxx"}]
}

//Form type
{
  "title": "a few words",
  "tags": ["array", "of", "words"],
  "description": "a few sentences",
  "kv": {"key1": "value1", "key2": "value2"},
  "fields": ["field1", "field2"],
  "related": [{"type": "xxx", "resource_id": "xxx"}]
}

```

```
//Fill type
{
  "field1":
  {
    "value": "value1",
    "source": {"type": "xxx", "resource_id": "xxx"}
  },
  "field2":
  {
    "value": "value2",
    "source": {"type": "xxx", "resource_id": "xxx"}
  }
}
```

**Status Codes**

| Code | Description |
|---|---|
| 200 | Result available (status=completed) |
| 202 | Processing in progress (status=processing) |
| 400 | Invalid input/SHA256 mismatch |
| 500 | Internal server error |

**Example Request**

```
{
  "client_id": "550e8400-e29b-41d4-a716-446655440000",
  "type": "doc",
  "SHA256": "9f86d081...b4b9a5",
  "has_content": true,
  "aes_key": "rsa encrypted"
  "content": "base64(AES(actual json string))"
}
```

### 4.2.2 Example Response

```
{
  "status": "completed",
  "result": {
    "title": "Lease Agreement",
    "tags": ["legal", "contract"],
    "description":
        "Standard residential lease agreement for 12
         ↪  months",
    "kv": {
      "landlord": "Jane Smith",
      "tenant": "John Doe",
      "term": "12 months"
    },
    "related": [{"type": "form", "resource_id": "uuid"}]
  }
}
```

### 4.2.3 Endpoint: /api/clear

Clears processing results from the system.

**Request Body (JSON):**

| Key | Type | Req | Description |
|---|---|---|---|
| client_id | String (UUID) | True | Client identifier |
| type | String | False | Processing type |
| SHA256 | String | False | Specific document hash to clear |

**Response:**

```
{"status": "ok"}
```

**Status Codes**

| Code | Description |
|---|---|
| 200 | Clearance successful |
| 400 | Missing client_id |
| 500 | Internal clearance error |

## 4.3 Cache Server (Flask+SQLite)

Stores and retrieves encrypted processing results using composite keys (client_id, SHA256, type).

### 4.3.1 Endpoint: /api/cache/query

Retrieves cached processing results.

**Query Parameters:**

| Key | Type | Req | Description |
|---|---|---|---|
| client_id | String (UUID) | True | Client identifier |
| SHA256 | String | True | Document hash |
| type | String | True | doc, form, or fill |

**Response:**

- Success (200): {”data”: ”ENCRYPTED_RESULT_STRING”}
- Not found (404): {”error”: ”Cache entry missing”}

### 4.3.2 Endpoint: /api/cache/store

Stores processing results in cache.

**Request Body (JSON):**

| Key | Type | Req | Description |
|---|---|---|---|
| client_id | String (UUID) | True | Client identifier |
| type | String | True | Processing type |
| SHA256 | String | True | Document hash |
| data | String | True | Encrypted result data |

**Response:**

201 Created (Empty body)

### 4.3.3 Endpoint: /api/cache/clear

Clears cached entries.

**Request Body (JSON):**

| Key | Type | Req | Description |
|---|---|---|---|
| client_id | String (UUID) | True | Client identifier |
| type | String | False | Processing type |
| SHA256 | String | False | Specific document hash to clear |

**Response:**

```
{"status": "ok"}
```

## 4.4 OCR Server (CnOCR)

Performs text extraction from images.

### 4.4.1 Endpoint: /api/ocr/extract

**Request Body (JSON):**

| **Key** | **Type** | **Req** | **Description** |
|---|---|---|---|
| image_data | String (Base64) | | Decrypted image |

**Response:**

```
{"text": "Extracted document text..."}
```

**Status Code:**

200 OK

## 4.5 Third-Party SDKs

1. **LLM API Provider (Zhipu)**
   Format OCR data using LLM.
   API Documentation: GLM-4, GLM-Z1

# Appendix 5 Usability Testing Script

## 5.1 Greetings and Introduction

Hi! Thank you for helping us test DocuSnap today.

Before we start: We want to emphasize that this is a test of our app, not a test of you. You can stop at any time for any reason. We encourage you to think out loud —share what you're seeing, what you're trying to do, or anything that's confusing. Please ask any questions you have as you go along. We've already signed in with a test account, so you won't need to create or log in to an account during this session.

## 5.2 Research Questions

Our usability test aims to answer these questions:

- How easily can users capture and enhance a document image to make it readable?
- How easily can users extract key information (e.g., totals, dates) from a receipt or invoice?
- How easily can users search for a specific document or piece of information using natural language?
- How easily can users fill out a digital form with auto-suggested information from stored documents?
- How confident do users feel about the security features when opening or sharing a document?

## 5.3 Tasks

**Task 1** Capture and Enhance

Upload or snap a photo of the sample receipt provided, then enhance it so the text is clean and easy to read.

**Task 2** Extract Key Information

Using the enhanced receipt, extract the total amount and vendor name.

Locate where this information is displayed or stored in the app.

**Task 3** Search with Natural Language

Find a specific document by typing or saying a natural language query like “Show unpaid invoices from June.”

**Task 4** Auto-Fill a Form

Open a sample reimbursement form and auto-fill it with information from the receipt you just processed.

**Task 5** Check Document Security

Open a sensitive document, verify its encryption status, and share only the first page with redacted totals.

## 5.4 Evaluation Metrics

**Task 1** Complete in ≤ 30 seconds; enhancement should improve legibility to at least 80% clarity (based on team consensus).

**Task 2** Find and identify total/vendor in ≤ 3 taps or clicks.

**Task 3** Locate target document using natural language search in ≤ 20 seconds, with ≤ 1 reformulation.

**Task 4** Auto-fill ≥ 80% of relevant fields without manual corrections.

**Task 5** Complete encryption verification + redacted sharing in ≤ 45 seconds, with participant expressing confidence in data security (qualitative metric).

## 5.5 Questionnaires

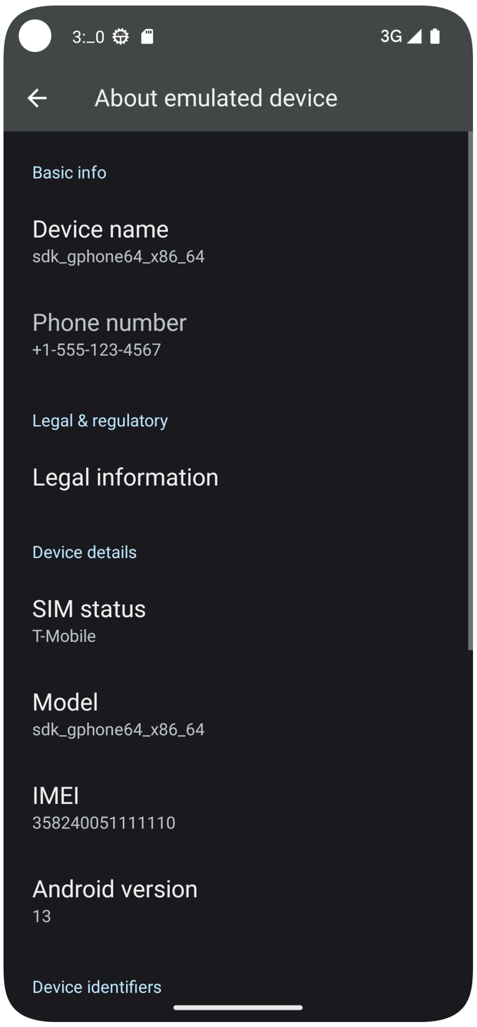
### 5.5.1 Pre-Test Questions

1. What is your current role or occupation?
2. How often do you manage physical or digital documents? (daily / weekly / rarely / never)
3. Have you used any scanning or document management apps before? (yes/no, examples?)
4. Do you use a smartphone, tablet, or computer for document-related tasks?

### 5.5.2 Post-Test Questions

1. On a scale of 1-5, how easy was it to complete the tasks?
2. What parts of the app confused you or slowed you down?
3. What features did you find most helpful?
4. Do you feel your documents would be secure with this app? Why or why not?
5. Any other feedback or suggestions?

# Appendix 6 Testing Environment Screenshots


3:_0
3G
About emulated device

Basic info

Device name
sdk_gphone64_x86_64

Phone number
+1-555-123-4567

Legal & regulatory

Legal information

Device details

SIM status
T-Mobile

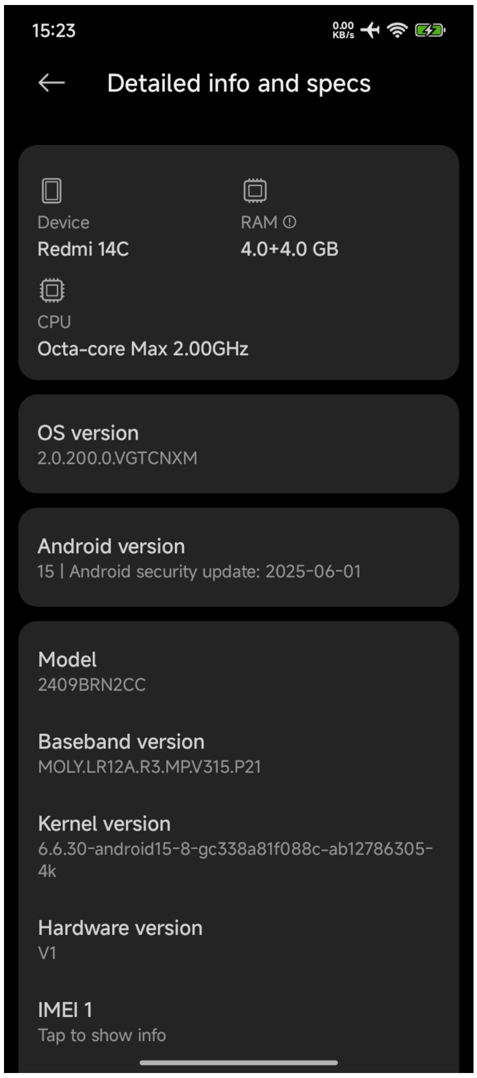
Model
sdk_gphone64_x86_64

IMEI
358240051111110

Android version
13

Device identifiers

(a) Android Studio emulator


15:23
0.00 KB/s
Detailed info and specs

Device
Redmi 14C

RAM
4.0+4.0 GB

CPU
Octa-core Max 2.00GHz

OS version
2.0.200.0.VGTCNXM

Android version
15 | Android security update: 2025-06-01

Model
2409BRN2CC

Baseband version
MOLY.LR12A.R3.MP.V315.P21

Kernel version
6.6.30-android15-8-gc338a81f088c-ab12786305-4k

Hardware version
V1

IMEI 1
Tap to show info

(b) Physical device (Redmi 14C)

**Illustration 6–1 Application testing environments**

# Research Projects and Publications during Undergraduate Period

N/A

# Acknowledgements

The completion of this thesis marks not only the end of an academic journey, but also the culmination of a shared vision, countless late nights, and a deep commitment to creating something that might actually help real people manage the chaos of modern life—one document at a time.

First and foremost, we would like to express our deepest gratitude to our advisor, **Prof. Pi Yibo**, for his unwavering support, technical insights, and sharp intuition. His guidance challenged us to think beyond technical implementation and to pursue meaningful, user-centric design. Thank you for pushing us to never settle for good enough. We are also sincerely grateful to the **UM–SJTU Joint Institute**, not only for the academic rigor it instilled in us, but also for fostering an environment that values curiosity, collaboration, and ambition.

To our **interviewees and testers**—your candid feedback, humorous anecdotes, and honest frustrations shaped this project in more ways than you know. From stories about losing passports to fumbling with tax receipts, you gave us the clarity and empathy we needed to build DocuSnap with real people in mind.

To our team members—**Yang Zijun, Peng Jingjia, Zhou Ziming, Qu Minyang, and Tang Huijie**—thank you for bringing your unique strengths to the table in the past 3 months. From pixel-perfect UI designs and intricate app backend to elegent server architecture and ironclad message protocols, this thesis is a product of every debate, whiteboard sketch, code commit and **3 a.m. debugging marathon** we shared. We learned not just how to build an app, but how to build together.

Finally, to our **families and friends**—thank you for your patience, encouragement, and understanding during our most intense weeks. Your support reminded us that while DocuSnap helps organize documents, you helped organize our lives.

This thesis is dedicated to everyone who has ever searched frantically for a scanned receipt named `IMG_RANDOM_NUMBER.jpg`. We hope DocuSnap brings you a little more order—and maybe a little less stress.

*– JI-DeepSleep, The DocuSnap Team*

# DOCUSNAP: YOUR AI-POWERED PERSONAL DOCUMENT ASSISTANT

This work presents DocuSnap, an innovative mobile application that transforms personal document management through artificial intelligence. Addressing critical challenges in handling physical IDs, receipts, and contracts, the system combines advanced computer vision with large language model processing to deliver intelligent document organization. At its core, DocuSnap implements a five-stage processing pipeline that begins with robust image enhancement including perspective correction and glare reduction, followed by precise OCR extraction, and culminating in semantic structuring that converts raw documents into organized JSON metadata with key-value pairs such as {“total”: “$49.99”, “date”: “2023-06-15”}.

The system’s semantic understanding capabilities enable automatic document categorization and intelligent linking of related items, such as associating visas with corresponding travel itineraries. Privacy is maintained through a unique client-server architecture that ensures no decrypted data is stored in the cloud using AES-256 and RSA-2048 encryption, achieving GB/T 45574-2025 compliance.

Performance validation on budget hardware (Xiaomi Redmi 14C) demonstrated the system’s efficiency, with sub-second UI response times, 3-second image processing, and approximately 20-second LLM parsing latency for complex documents. Developed using Jetpack Compose for the frontend, Flask for backend services and GLM 4 Plus as LLM, DocuSnap supports all kinds of documents and forms with a very high accuracy. The intuitive interface, refined through iterative user testing, provides seamless workflows from document capture to secure retrieval, establishing a new standard for privacy-preserving document management systems.