广州大学

本科毕业论文（设计）

课题名称	AI驱动的零售商品全流程管理系统设计与实现
学　　院	计算机科学与软件工程学院
专　　业	软件工程
班级名称	软件（创）211
学生姓名	张景致
学　　号	32106300004
指导教师	闫红洋
完成日期	2025 年 4 月 17 日

教 务 处 制

AI 驱动的零售商品全流程管理系统设计与实现

软件工程专业 软件（创）211 班 张景致

指导教师：闫红洋

摘要 当今时代人工智能技术、人工智能模型训练和推理行业发展迅速，各行各业都在积极利用人工智能技术助力各个产业的数字化、智能化进步，进一步推动社会生产力的创新发展。在这股 AI（人工智能）浪潮之中，本就因为电子商务的兴起而态势疲弱的传统零售行业面临进一步落后的风险。但是具有实际体验并与商品进行各种互动等独有优势的传统零售行业仍有进一步发展的空间、数字化转型的必要。本设计以服务于广大消费者、零售从业者为宗旨，以构建实际零售管理系统为实验方法，在商品文案设计、商品检索、商品推荐与咨询、商品结算、库存管理、销售数据分析等方面，探究并提出了一系列将时代前沿 AI 技术与传统零售行业管理和服务系统相结合的具体办法。

关键词 人工智能；零售；大语言模型；实体经济

ABSTRACT Nowadays the advancement of AI (artificial intelligence) technologies and AI model training and inference industry are significant. Most industries and areas are actively leveraging AI technologies to empower the digitalization efforts, boosting the creative step forward of our society. However, despite the emerging AI innovations, traditional retail businesses, being already behind the waves because of the rising e-commerce industry, face the risk of further decline. But it is also apparent that having the unique feature of allowing end-users to experience and interact with products demonstrates that retail businesses could and should be further advanced and digitalized. This project design, with the consumers and retail owners in mind, in the way of building an functional retail management system, researches and purposes a set of concrete methods to fuse AI and traditional retail together, including the design of product write-ups, product searching, product recommendation and consulting, check-outs, inventory management and analysis of sales data.

KEY WORDS Artificial Intelligence, Retail, Large Language Model

目 录

1 前 言

近几年来，人工智能（AI）技术有了前所未有的深入发展。从上世纪 50、60 年代“机器学习（Machine Learning）”[1]概念被提出时它初次登上历史舞台，到人们利用图像处理单元（GPU）等专用硬件进行处理[2]如 AlexNet[3]和 ResNet[4]等规模成指数级别增长的深度学习（Deep Learning）模型，再到 2017 年 Transformer 模型[5]掀起自然语言处理领域变革，AI 领域已然有了长足的进步。而近几年从 OpenAI ChatGPT[6]引爆大语言模型（LLM）热潮到以 DeepSeek 为代表的一众大幅降低部署、使用成本的开源大模型走进人们生活不难看出，AI 技术势不可挡，并且在可以遇见的未来还会进一步发展壮大。

人工智能技术既是一个独立的领域，又是其他行业和领域进一步深入发展[7]、进行数字化转型[8]的不可或缺的一部分。例如多模态（multi-modal）的大语言模型将图像、音频等不同媒介的信息与一般大语言模型的文字信息连接起来，形成了“看得懂”、“听得懂”的大（语言）模型[9]。从智能制造到智慧医疗，人工智能在垂直领域中逐步渗透，与不同行业、工业相结合，使其得到了新的发展力量。

与朝气蓬勃的人工智能产业现成鲜明对比的一个领域是传统实体零售行业。改革开放以来，直到电子商务（电商）产业兴起以前，实体店铺几乎是民众购买不同产品的唯一方式，担任了将商品从设计、生产和批量分发的企业转移到最终用户的桥梁的角色，既是产品供给的终点，也是收益反馈的起点。以淘宝为代表的网上购物平台（也就是面向最终消费者的电子商务平台）兴起之后，产业链成本、消费便利性等种种因素使得消费者愈发青睐网上购物，足不出户便能选购喜爱的产品。即便没有自主选购的意愿，吸引人的商品也会从各种不同的广告推荐渠道来到消费者眼前。如此突出优势，与近年波动的经济环境和不安定的地缘政治情况，化为了许多中小型实体零售企业、个体户的运营压力，甚至使得其中的许多面临不得不终止运营的极端情况。

然而，实体零售行业仍有无法被取代的优势。消费者在参与线下购物的过程中，可以通过与商品的近距离互动来产生对其直观的印象，这种“零距离”的、有着天然信任的购物体验无法复制；网上购物的方法割裂了顾客支付商品价格和接受实体商品的过程，对物流有着一定程度的依赖，而实体零售则可以购买当时直接获得对应实体产品。此外，线下购物的过程同时也可以是社交的过程，可以营造独特的社会价值，增强消费者的生活体验。

为了在充满激烈竞争的经济环境下保持甚至提升自身的地位，实体零售行业需要积极进行自身的数字化、智能化转型，通过各种不同方法充分发掘实体零售的独特优势。在这样的情势下，“新零售”[10-11]、“智慧零售”[12]等概念、预想应运而生，“人-货-场”匹配的最优化[13]受到广泛研究探讨。但是，这样的转型目前一般只有市场头部企业开始实施，体量较小的企业和个体户尚无资本和技术能力展开；并且其中较为重要的一个方面，实体零售与人工智能技术的结合，还有待深入开发。

本设计项目立足于传统实体零售行业，尤其是成本上受到较大限制的中小型企业和个体户，对产业数字化、智能化的需求日渐急迫的当下，致力于探究在传统零售行业中应用、融入乃至于整合人工智能技术，利用人工智能的便利增强经营者的营业能力并降低运营门槛，将更便利的、更有亲和力的实体购物体验带给消费者。具体来说本项目在实现一套基本可用的分布式零售管理基础设施、管理软件的基础上，利用不同类型的人工智能技术实现了以下几个不同的功能模块：

- 服务器部分
  - 智能分词技术、近义词搜寻技术驱动的关键词搜索引擎
- 管理端部分
  - 大模型驱动的智能商品文案编写助手
  - 大模型驱动的智能业务图表分析、运营建议模块
  - 基于智能条码识别、扫描技术的点货功能
- 门店端部分
  - 基于图像分类的智能商品识别功能
- 客户端部分
  - 大模型驱动的多轮对话、搜索推荐智能导购助手

该部分之后的文章内容从介绍和分析该领域（零售管理）的现有方案（章节 2）开始，其后从整体角度对该项目所实现系统的架构设计作出解释说明（章节 3），然后分别对各个模块的具体设计、实现方案进行详细的描述，再之后对该系统相关的测试和运行效果进行列举和说明。最后对该系统的实现效果、未来改进空间等话题进行讨论，进而结合这个项目的情况对该领域的未来作出预测来对该文章收尾。

2 现有方案

零售管理的需求是跨越客户关系管理（CRM）、企业资源计划（ERP）、销售时点情报系统（POS）和“进销存”（进货、入库、销售）等多种运营管理领域的综合性的管理需求。同样，目前市面上较为常见的各类零售管理方案因此也覆盖了其中部分或全部的领域。

其中十分具有代表性的“微盟”是以微信平台为依托的，覆盖实体店面收银、库存管理、会员管理等各个功能的一个功能较为广泛的软件即服务（SaaS，Software as a Service）管理产品。在门店、导购数字化方案之外，“微盟”还提供私域电商[14]（由一个或多个特定实体主导的私有电子商务平台，有别于“淘宝”等较多不同实体进驻的公共平台）搭建和管理服务，有助于其客户利用平台整合度高的优势推进其自身的数字化和智慧化。

另一个具有一定代表性的是伯俊科技公司的“云 POS”和“BOS Cloud”等零售门店管理服务。该方案以方式种类丰富的 POS 为核心，提供传统桌面式收银台、手持式收银终端、自助收银终端等专用硬件。与“微盟”一样，该方案同样提供了一定程度的在线销售功能。不同的是“云 POS”采用接入一般外卖、直播电商平台的方法实现线下门店的“线上化”，以此促进其客户的数字转型。值得注意的是，该平台具有接入其他第三方管理平台（如“微盟”）的方法，其供应商绑定风险较低。

除了上述的两个方案，还有如“T+Cloud”实体店新零售管理系统、“茗匠”有人或无人零售收银管理系统、“金蝶”小微企业云服务平台和“致心零售”SaaS 零售软件等整合了多个管理维度，提供零售行业一站式管理的各种不同方案。

这些方案都在可以从某个或某些方面上增强对应零售业务的数字化、智能化程度，提高运营便利、自动化程度，进而改善产业生态和客户体验，但是它们都具备相似的问题：注重于将线下门店“线上化”、将电子商务“实体化”，把电子商务的优势整合到传统实体零售行业中，而一定程度上忽视了对这些技术和手段（无论线下还是线上）本身的进一步优化和发展，进而在对经营者门店数字化工作便利性的增强和对消费者核心购物体验改善的工作上有所不足[1]。

而本设计所构建的 AI 驱动的零售商品全流程管理系统，通过利用人工智能这一“通用算法”，实现了以传统商业应用程序及其算法所无法实现的功能和特性，进而探究了从经营者、消费者多角度多层次提高零售行业运作水准的可能性，给出了将人工智能技术和传统零售行业相结合的切实方案、可行方案，为零售行业的数字化、智能化未来添砖加瓦。

[1]上海微盟企业发展有限公司在 2024 年前后发布了“WAI”人工智能工具集合。根据其公布的少量资料，这些工具以面向经营者为主，尚未覆盖到零售业务的各方面。此外，这方面可供参考的资料稀缺，故“WAI”产品未被在正文提及。

3 整体架构

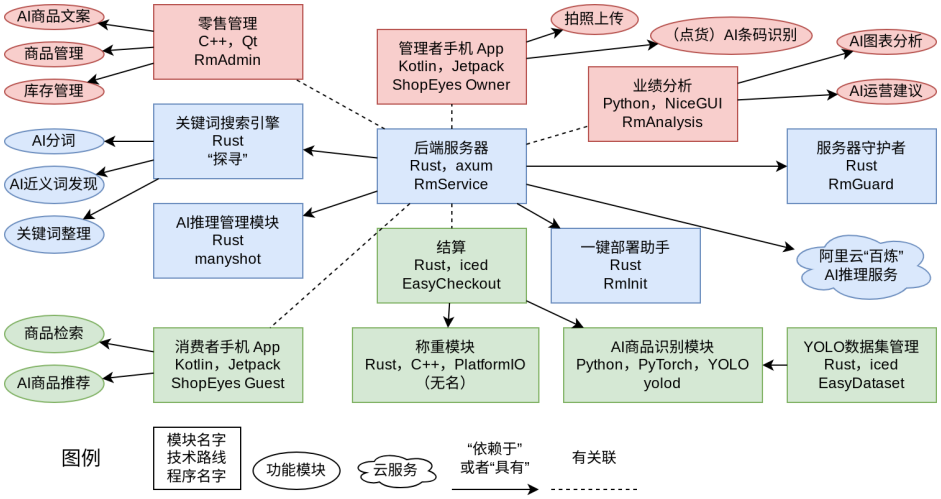

图 3.1 本设计的系统架构图示：其中蓝色部分为服务端，红色部分为商家端、绿色部分为顾客端。图例（不同形状的各自意义）位于图像下侧。

本设计的系统架构如图 3.1 所示，而从图像中不难看出，该系统整体上呈现出客户端-服务器模式的结构，其中客户端部分分为面向零售行业从业者的商家端和面向最终消费者的顾客端。

3.1 服务端

本设计中对服务端功能的期望主要有如下几点：

- 高性能的 HTTP API 服务器
- 商品、库存、订单等运营资料的增删改查
- 高性能、效果良好的智能商品搜索
- AI 大语言模型推理托管

实际部署的情况下，服务器可能需要在短时间内处理来自大量客户端应用程序的服务请求，比如营业高峰期来自许多客人的商品检索、询问 AI 助手索取导购建议的请求，因此服务器本身的效率必须纳入设计的考虑之中。基于这样的考虑，该设计采用 Rust 语言进行开发。Rust 是比较流行的一门强调内存安全的系统编程语言，利用这门语言，服务器的代码执行效率有充足的优化机会，并且开发的便利性得到了一定保证。为了充分利用大部分部署环境都将会具备的多处理器的对称并行处理

（SMP）执行环境，服务器内主要采用由 tokio 第三方库驱动的异步开发的技术手段。为了实现服务器与客户端的有效、高效、泛用性强的信息双向传输，采用通过（商铺内部）局域网的 HTTP。为了与总体的技术选择相统一，采用依托并行计算技术开发的 axum HTTP 服务器框架。

服务器需要能实际储存并处理在零售运营过程中需要的各种数据。为了满足这个期望，该设计中采用基于 SQLite 关系数据库引擎的 rqlite 分布式关系数据库管理软件作为原始数据存储、查询和管理的手段。为了提供统一可靠的开发接口，rqlite 服务器将只面向服务器软件开放，而客户端任何对实际业务数据的访问都必须通过服务器对其结构化、统一化的封装。

为了实现在众多商品中快速找到顾客所需，服务器需要具备通过（顾客提供的）一段可能和商品本身文字内容不尽相同的搜索语句对商品进行匹配的功能。市面上常见的如 Apache Lucene、Elasticsearch 等各种相关产品与该设计的相关理念并不匹配：因为较为复杂而不适合在该设计所期望的低成本设备上部署；对中文没有比较深入的优化，并且搜索对象仅限于搜索目标中出现过的字词。为了解决这个问题，该设计中包含一个由多种人工智能技术驱动的，高效、高匹配率、简明易用的中文特化搜索引擎“探寻”。

鉴于该项目中多处利用到了大语言模型，若是服务器可以向各个客户端软件提供统一的推理接口，将不同模型、不同服务商的区别消除，将有利于项目的整体可用性、可维护性。因此，服务器提供一个专门开发的的推理模块，具备单次推理（“oneshot”）、多次尝试（“manyshot”）和有状态多轮对话管理等功能。

3.2 商家端

本设计中对商家端功能的期望主要有如下几点：

- 高性能、便于使用的用户界面
- 功能丰富、易于操作的商品、库存管理界面
- 完整的销售数据查询界面
- 业绩图表生成、展示界面
- 业绩图表 AI 智能分析、运营建议
- AI 条码识别点货
- AI 商品文案自动生成、批量生成
- 商品识别数据集创建和修改

- 从手机上传用于商品的图片

为了在普通的计算机上高效管理营业资料，商家端需要一套对系统要求较小的、对键盘鼠标操作较为友好的，对屏幕尺寸需求灵活的用户界面。基于这样的缘由，该设计的商家桌面端应用程序采用 Kotlin 作为业务逻辑开发语言以最大化开发效率，进而采用受到工业界广泛采用的 Qt Widgets 应用程序开发框架在 Java、Kotlin 语言上的实现 Qt Jambi。

在实际操作的情况下，经营者不可避免地将需要对各类数据进行条件细致的筛选。为了满足这样的需要，商家桌面端应用程序为多种不同零售资料的查询准备了既功能丰富，又简单易懂的图形化高级搜索条件拟写工具。

将销售数据按表格列出是简单易行的，但这样的数据展示方式往往无法满足营业者的营业数据分析的需要，图表可以更好地呈现数据之中潜在的结构性和规律性。为了实现这样的功能，本设计采用业界惯用的 Python 语言作为数据分析的主要语言，利用 HTTP API 与基于 Rust 的服务器进行通讯，并采用 pandas、numpy 等数据科学库展开数据整理工作，利用 matplotlib 库进行图形的绘制。为了简化开发过程，图表部分的用户界面同样采用 Python 编写，同时选择 NiceGUI 用户界面框架实现与上述第三方库更高的整合度。

为了减轻零售从业者观察图形规律的压力，该设计利用来自阿里云“百炼”AI 推理服务的多个 AI 模型，针对不同类型、复杂程度的营业数据图表进行“先观察后思考”的“接力式”智能分析报告编写或者“边观察边思考”的快速图形规律总结，向从业者提供详尽准确的数据分析。

业务资料中的商品图片较为特殊，在手机设备上处理可能相比在桌面型计算机上更加便捷。因此，该项目采用 Jetpack Compose 安卓应用程序开发框架实现面向商家的管理用应用程序。利用该应用程序营业者可以较为简单地将手机中的照片或现场拍摄的照片上传到服务器中。

实体零售行业常常无法避免对仓库中、货架上产品进行审计（统计），以此确认实际产品数量与系统中库存数量对应关系的需要，而手工记录并比照的方法费时费力并且容易出现错误。因此，该设计的移动商家端应用程序采用谷歌 MLKit 的 AI 条码识别功能开发了利用手机自带摄像头的 AI 条码识别并记录、上传的功能。

为了简化用于智能结算房屋中的商品识别模型的训练工作，该项目具备利用 Rust 和 iced 用户界面框架开发的数据集创建和修改功能。数据集准备的过程分为“类别规划”“图片采集”和“图片管理”三个连贯的部分。从业者能够先根据实际的需要在应用程序中输入需要的商品种类，再利用智能商品识别对应的摄像设备进行数据集中图片的采集，最后在采集的图片中筛选出质量较高者。

3.3 顾客端

本设计中对顾客端功能的期望主要有如下几点：

- 美观大方、便于使用的用户界面
- 推荐商品浏览
- 商品检索和详情浏览
- AI 导购多轮对话商品推荐
- 基本商品结算
- AI 商品识别-计重结算

为了最大程度提高消费者获取商品信息的便利程度，增强新零售购物体验，该项目采用 Jetpack Compose 开发面向消费者的用于浏览商品情况的应用程序。应用程序分为“推荐”“搜索”和“询问 AI”三个板块，分别通过不同的用户界面、不同的对服务器 API 的请求方式来实现相应的功能。其中推荐功能展示由从业人员在商家端预先设置的一系列推荐商品，而搜索功能顾名思义。

另一个功能是“询问 AI”。在该功能界面下，用户可以通过屏幕底部的搜索框输入需要向 AI 询问的导购相关问题，用户输入的问题将会经由服务器中转发送到后端预先指定的推理服务和选择的大模型。在获取输出之后，顾客端应用程序会利用 AI 的答复再次利用 AI 进行关键词的提取，进而利用归纳得出的关键词进行对商品的检索，进而实现对用户推荐相关的商品的功能。

为了同时满足店员辅助结算和消费者自主结算的需要，该项目需要提供一个用于结算设备的简洁明了的用户界面。因此，该设计采用 Rust 语言及 iced 用户界面开发框架设计开发结算用用户界面。用户可以利用通过人体工学输入设备（HID）方式连接到结算设备的激光条码扫描设备来输入任何按件结算的商品。

某些零售细分领域所提供的商品（如生鲜蔬果、熟食等）是按重量结算的，需要称重才能获取实际价格，并且按实际情况有可能商品表明无法粘贴对应的条形码（比如肉类），需要其他手段辅助才能确定商品的具体类型。为了解决这样的问题，本设计利用 YOLO 图像分类模型，结合前文提及的商家端数据集管理工具，开发 AI 智能商品识别算法，用以简化计重商品结算流程。

4 零售管理系统

4.1 服务器

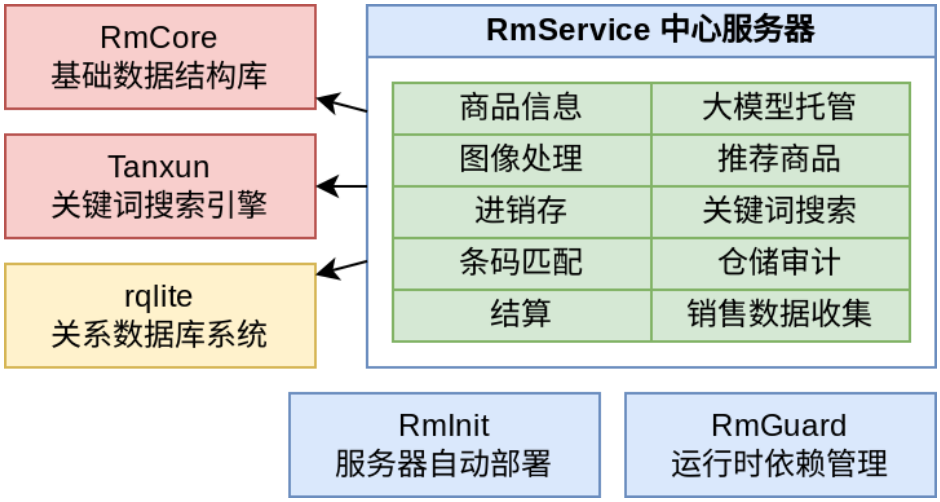

图 4.1 本设计的服务器整体设计图示：其中蓝色表示相互独立的应用程序，绿色表示服务器的功能，红色表示功能库，黄色表示第三方库或应用程序，箭头表示被指方受到依赖。

4.1.1 数据结构

如图 4.1 所示，因为采用了客户端-服务器设计模式，该系统所有业务相关信息统一由 RmService 模块进行管理。因此，为了在系统中体现充分的可维护性、可拓展性，实际对数据进行存储的关系数据库系统 rqlite 原则上无法被客户端（各个商家端、顾客端应用程序）所直接访问，而是由 RmCore 所定义的数据结构进行封装。

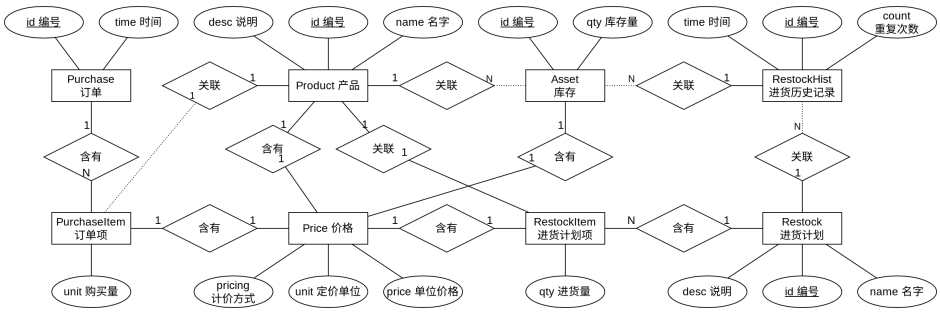

图 4.2 RmCore 模块所定义数据结构的实体关系图

如图 4.2 所示，该模块定义了在零售经营过程中会利用到的各种（需要持久化存储的）数据结构，并且已经以面向对象的、有利于代码复用、序列化和反序列化和有利于在不同开发环境下形成统一接口的方式进行封装。为了便于在数据库中以整数

方式存储许多常常为小数的数据，规定任何直接代表金钱数量的值以人民币分为单位，任何直接代表物体质量的值以克为单位。

其中值得注意的一个类型是库存“Asset”。为了便于对库存所对应的进货批次、进货价格等信息进行跟踪，每一个库存项都与一个进货历史记录项目“RestockHist”相关联。因此，同一个产品可以因为多次不同进货并且较老进货批次剩余产品尚未被消化完毕而存在多个库存条目。同样地，只需要统计一个 RestockHist 所对应的 Asset，便可以统计某一个批次进货的消化情况。

另一个较为特殊的类型是价格“Price”。考虑到不同零售商品类型定价策略的区别，该类型的属性计价方式“pricing”为具有“Package”（按件）和“Weight”（按重量）两个可能值的枚举。而属性定价单位“unit”在按件计价时代表属性价格“price”对应的商品件数，反之则是克数。而将属性 price 和 unit 分离（而不是表示为单位价格）有助于避免如“十元三件”（一件 $\frac{10}{3}$ 元）此类出现整数或 IEEE 754 浮点类型数字无法准确表示的数值的情况。

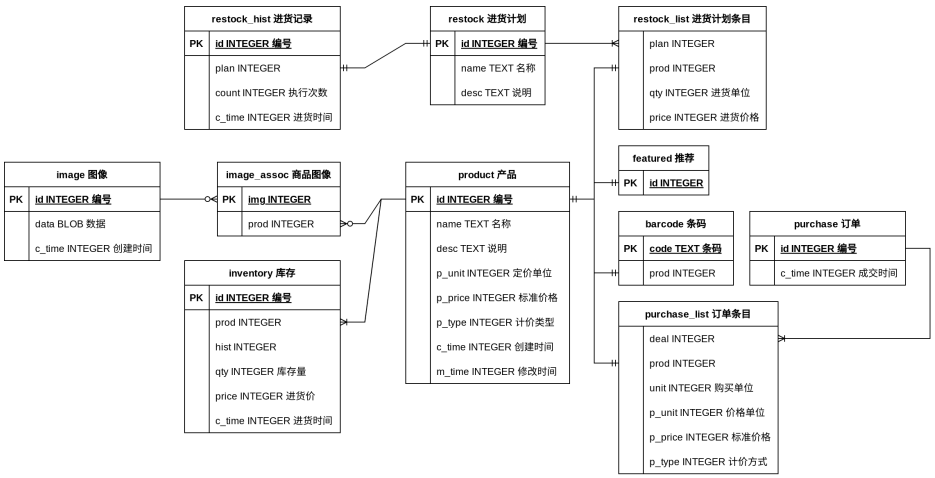

图 4.3 数据库中存储数据格式的实体关系图

图 4.3 所描绘的是 4.2 所示的模型还有少数以其他方式封装的数据的在关系数据库系统中的实际存储方式。除了 Price 类型没有单独表格，而是被扁平化地存储在了各个表中的区别，数据存储的方式基本与封装成一比一对应关系。少部分表格具有额外的创建、修改时间字段（“`c_time`”和“`m_time`”）留作未来拓展使用。值得注意的是，为了改善大规模访问的效率，商品图像数据并不是存储在外部文件系统，而是和其他资料一同存储在数据库中。

4.1.2 应用程序接口

为了实现各个客户端应用程序与服务器的有效信息交换，服务器采用 HTTP 协议进行信息传递，即服务器实现面向客户端应用程序的 HTTP API。由于 API 覆盖范围较广、数量较大并且功能迥异，故该课题代码开发时大部分 API 端点设计遵从如下规范：

- 期望请求主体（body）部分带有实际数据者，接受 POST 方法的请求；否则接受 GET 方法的请求，并且对方法错误的请求响应“Not Found”状态。
- 期望请求主体带有多于一个字段者，接受使用 JSON 格式序列化的数据，否则按实际情况决定主体数据类型。
- 较短、较少的请求参数，可以通过 HTTP 地址参数而不是请求主体传递。
- 除纯二进制数据（如图片）外，方法一律使用 JSON 格式序列化返回值。
- 在端点调用遇到责任在客户端（调用参数格式错误等）的问题时，返回“Bad Request”状态并且尽可能在回复主体中包含对问题根源进行描述的字符串。
- 同上，但责任在服务器（意外情况）时，返回“Internal Server Error”状态并且在主体中包括导致问题的函数调用返回的相关错误信息。
- 客户端请求的资源或任务结果尚未准备完成时，返回“Accepted”状态并在回复主体中标注具体任务现状。

为了支持商家端应用程序中对某些数据条目的高级搜索功能，服务器对某些数据类型提供了封装程度较低的请求方式，即允许部分客户端应用程序自行合成并向服务器提供用于数据获取的 SQL 查询 `SELECT` 调用参数中的 `WHERE` 子句。值得注意的是，该特性因为受到的自动检查、合法性担保较少而具有一定的危险性，不合理的使用可以引发较严重的安全隐患。因此，只有特定商家端的功能将会使用到这个 API 终点，并且子句的合成也会（在客户端）受到一定的规范和限制。

4.1.3 自动部署

不难发现，该设计所实现功能板块领域跨度较大，在原理、技术路线和使用环境要求上不尽相同，甚至在某些情况下相差甚远，所以部署的难度和复杂程度是比较高的。因此，该设计包含一个利用与服务器相似的技术路线开发的自动服务器部署应用程序 RmInit。

该程序具备利用系统命令执行和命令输出检测该设计中各个功能模块在该程序所运行的设备上运行所依赖的第三方软件库或执行环境是否正确安装的功能；具备使用系统自带下载工具（如 `wget` ）自动从互联网上下载对应该程序所运行的设备的体系结构的 rqlite 应用程序合集，并且在部署的系统上运行数据库初始化脚本的功能；能够按照预先准备的应用程序项目列表逐个运行对应的编译任务并复制编译输出的二进制程序，支持使用 `uv` 托管的 Python 项目、使用 Gradle 项目管理工具的 Java-Kotlin 项目、基于 CMake 项目管理工具的 C++ 项目和使用 Cargo 命令管理的 Rust 可执行程序项目目标。

为了适应在多设备分布式系统中正确分配各设备功能特性、规避在特定平台无法完成编译任务的应用程序，RmInit 采用了 `clap` 命令行参数分析器，支持利用命令行参数控制大部分功能的包含与否、部分配置文件的覆盖与否。

此外，该应用程序还实现了可选的配置文件升级功能。配置文件升级是一个递归的过程，每一次调用都由新配置和旧配置（或者它们的一部分）参与。若新旧配置均为基本值类型且类型一致，则旧配置受到保留。若新旧配置类型、结构不一致，则使用新配置取代之。若新旧配置为键值表，则利用旧表内容覆盖新表，每个元素的覆盖递归执行该过程。

4.1.4 服务依赖管理

服务器中不同应用程序可能存在一定依赖关系，比如 RmService 依赖于 rqlite 关系数据库管理系统，此时合理安排各个应用程序或后台服务的启动顺序对系统整体稳定程度将会有所帮助。因此，服务器工具软件中包含 RmGuard 服务器运行时依赖管理，支持按网络请求启动或停止相关服务的服务器守护程序。该应用程序内置服务器中不同模块二进制程序之间的依赖关系，并且启动时将会读取对应的配置文件。不论使用任何方式（自启动或按需启动）来唤起任何模块，都会触发对应程序所依赖的模块（若尚未启动）的启动过程。

4.1.5 图像存储

为了顺利在较为受到限制的存储空间内存放潜在的海量商品图像，并且改善各应用程序获取商品图片的速度、降低网络带宽压力，服务器具备自动对图像进行缩放和压缩的功能。服务器所接受的图片（已编码的数据）首先会被解码，然后将被在保持图片原本宽高比的情况下，将图片较长一边的长度限制在 300 以内，并按需调整另一边，并利用较为快速的双线性插值办法进行重采样。程序将逐个尝试 100、95、90…60、55 的 JPEG 压缩率对图片进行编码，直到使用了最大压缩率或图像大小小于配置文件制定阈值为止。

4.2 管理界面

为了充分利用面向桌面型计算机开发的应用程序的操作效率高的特性，管理界面模块采用如图 4.4 所示的按标签页分文档的 UI 组织方式，其中大部分标签页采用主要内容板块加侧边栏展示额外信息的布局方式，以增强对键盘鼠标操作的友好性。鉴于许多种类的零售业务资料可以被抽象为同一种类型的多个不同实例，并且具有同样的对理解数据内容较为重要的字段（如名称、说明），故决定以表格方式对其进行整理。管理界面具备如下的功能：

- 商品管理

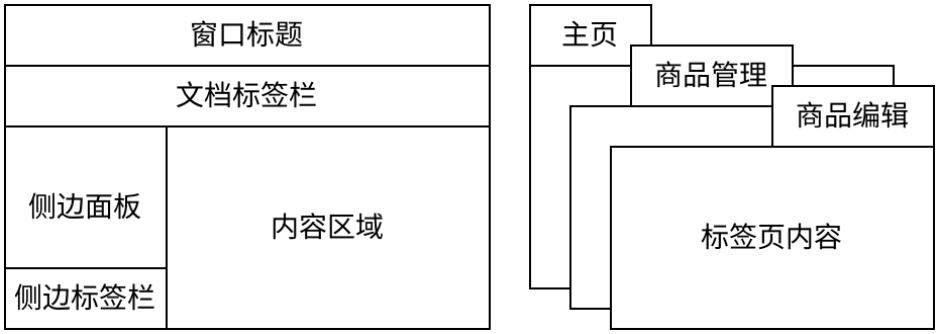

图 4.4 左侧：桌面商家端的用户界面布局；右侧：多个文档标签的实例

  - 创建或编辑商品
  - 检索、查看和管理既有商品
- 库存管理
  - 管理、预览和执行进货计划
  - 查看仓储情况
  - 管理仓储审计批次
- 编辑推荐商品
- 管理商品图片

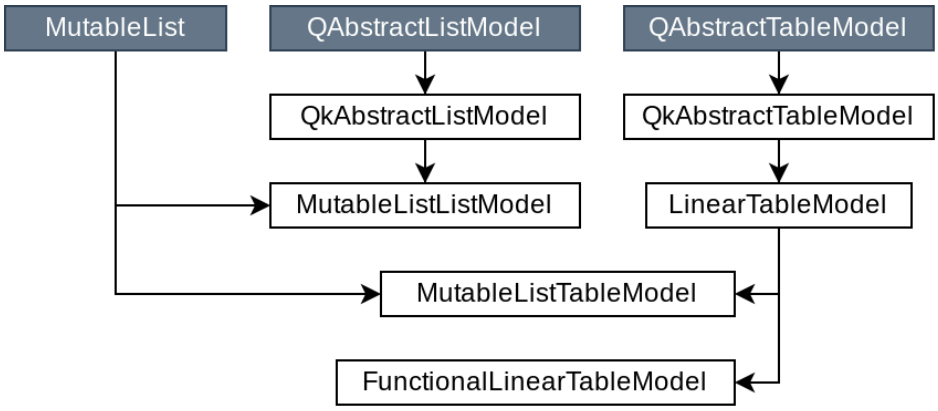

图 4.5 用于表格组件的模型类继承关系：灰色框代表第三方库或开发环境带有的类，箭头指向子类或者实现该接口的类。

不同表格模型（向表格组件提供数据的对象）类之间的继承关系如图 4.5 所示，其中 `MutableListListModel` 和 `MutableListTableModel` 在表格模型中整合了可变不定长数组的特性，可以将开发模型中的对象数组与对应对象集合在 UI 上的展示相同步，避免繁重的界面开发任务，以此简化数据映射的开发复杂程度。

此外，管理界面的许多作为既有信息展示方法的表格组件还具备通过键盘快捷键或右键菜单进行搜索结果整理、复制或粘贴的功能，从业者可以通过在不同的搜索结果、功能模块之间粘贴剪贴板中编码的对象引用来实现较为复杂的查询和修改。例如若需要查询某几种商品的库存情况，可以首先在“商品管理”中对所需商品进行检索，再在该界面内复制所需商品，在库存查询界面粘贴，即可查看对应的库存情况；若需要在编辑商品信息的过程添加图片，可以在图片管理界面检索到所需的图片，在将图片（的引用）复制到商品编辑所对应的标签页。

4.2.1 高级搜索

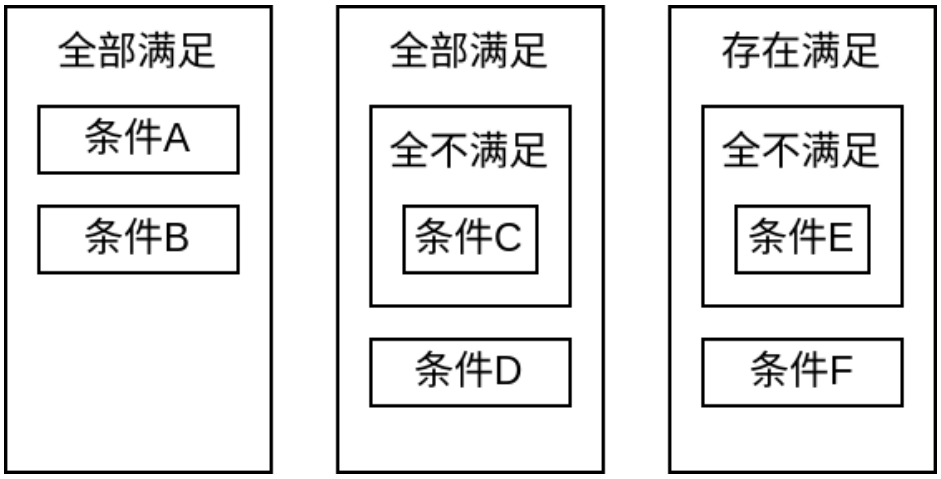

图 4.6 复合查询条件示例：三个条件分别等价于 $A \wedge B$，$\neg C \wedge D$ 和 $E \vee F$

零售业务运作的时候，从业者往往需要从数量较为庞大的数据中筛选出所需的一小部分。为了满足这样的需要，管理界面对某些数据类型提供基于布尔代数思想的复合查询条件设计工具。因此从业者并不需要编写布尔代数表达式或 SQL `WHERE` 子句就能进行条件复杂查询操作，通过使用鼠标在应用程序提供的用户界面上操作，从业者可以自由组合以下几种类型的条件（`Criterion`）：

- 字符串字段：前缀为、后缀为、等价于或包含指定字符串
- 数值字段：等于、大于、小于指定数值
- 日期时间字段：早于、晚于指定日期时间组合
- 编号字段：包含于指定正整数集合
- 以上条件的集合：其中元素全部符合、任意符合、全不符合、任意不符合

输入完成之后，若应用程序没有检测到非法值，将会生成对应的 SQL `WHERE` 子句并发送到服务器，服务器将按照对应语句运行 SQL 查询并返回结果。例如查询名

称（name）包含“可乐”二字，并且于 2025 年 4 月创建（c_time）的商品，与以下 SQL WHERE 子句近似的语句将会被产生：

```
(name like '%可乐%' and c_time > 1743350399 and c_time < 1746028800)
```

4.2.2 移动端程序

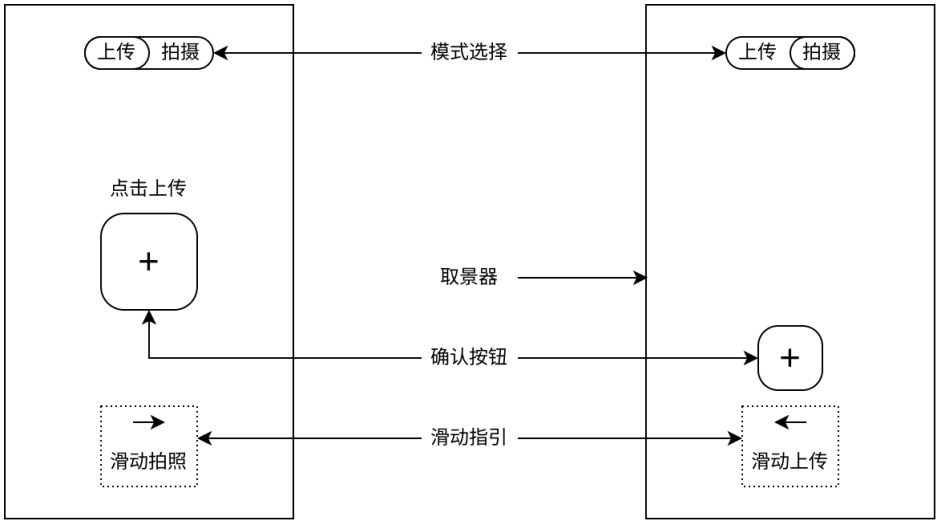

图 4.7 商家端移动应用程序图像上传部分 UI 设计

为了进一步降低营业成本，该设计包含一个利用 Kotlin 语言和 Jetpack Compose 用户界面库开发的商家端应用程序，用以简化向系统提供图片的流程。从图 4.7 可见，用户可以通过屏幕上侧的标签栏或滑动屏幕空白部分来在上传本地图片和拍摄新照片的界面之间切换。在上传界面上，点击确认按钮可以唤起图片选择界面，用户可以选择一个或多个图片进行上传；在拍摄界面，UI 背景部分为取景器画面，在选择完成拍摄角度之后用户只要点击确认按钮即可拍摄照片并即刻上传。

4.3 结算界面

该设计具有采用 Rust 语言和 iced 实验性用户界面框架构建的用于结算的应用程序。该程序通过与条码识别器、质量传感器和摄像头交互来实现获取商品数据，其后可以在其用户界面上展示待结算的产品。由于具备相应的软硬件支持，该程序可以进行不同计价方式的商品（按件和按质量）的混合结算，一定程度减轻了店员结算的压力和用户自主结算的门槛。

4.3.1 条码识别器

为了实现高效识别（计件）商品，结算程序默认通过使用 USB HID 协议通讯的激光条码识别硬件来读取商品条形码。这种硬件将模拟键盘输入条形码内容，应

用程序在商品结算界面将检测是否在较短时间之内从键盘输入符合 EAN（European Article Number）-13 格式的数据，并在需要时与服务器进行通讯。因为服务器存储条形码的方式是使用字符串而不是数字等其他格式，条形码格式和输入方式方面该设计具有较高的可拓展性，实际部署的情况可以根据不同零售行业细分领域的实际对其进行调整。

4.3.2 质量传感器

通过将质量传感器（带有信息传输接口的“电子秤”）整合到结算设备中，设备将具备直接进行计重商品结算的功能，而不需要额外的硬件或设备来进行称重，也不需要额外编写标签并打印。这样，计重商品的结算也可以完全由顾客自助完成，不需要来自店员的帮助。

该设计理论上对质量传感器的规格和信息传输方式没有要求，只要该硬件提供足以断定价格的精度和稳定程度即可使用。本设计默认使用市面上采用率较高的海芯科技 HX711 电子秤用模数转换器，该芯片成本较为可控，并且在多种平台上具备成熟生态。值得注意的是，HX711 的串行通讯功能对相应频率和延迟的要求较为严格。为了解决此问题，本设计包含 Arduino UNO 3 微控制器开发硬件上采用 C++ 语言和 PlatformIO 嵌入式开发技术编写的驱动程序，将 HX711 的专有输出格式转化为了易于处理的文本格式，并通过 USB 模拟串行接口终端通讯与计算机（和结算程序）相连接和消息传递，以此实现了称重和调零的功能。

5 商家端 AI 功能

图 5.1 商家端 AI 功能总览

面向零售行业从业者的 AI 特性主要围绕着商品设计和图表分析展开。这两个业务板块实际上是以宣传为目的、以市场需要为导向的文本编纂；对营业中的各项数值在不同时间段中的变化趋势、不同周期的重复规律的研究。显然，不管是专业文本编纂还是销售数据挖掘都一定程度上超出了一般零售行业在店从业人员的能力范围。

对于一般的连锁店或其他类型的多店面实体零售企业，门店可以通过系统联网、数据同步的方式将这两项工作转移到企业本体的研发、市场部门，运用专业文学工作者、数据分析人员的专业能力来缓解这样的矛盾，一定程度上也有助于同品牌店面行为的标准化和品牌形象的塑造。

然而，这样的做法为每个特定的店面制定相应的经营策略的成本是较高的，并且将会对企业的相关资源造成较大的压力，但若是将每个店面的数据整合处理协同考虑，再分发统一的策略，又会使得各个店面效率的上限下降。并且，这样的运营模式一定程度上忽视了在店管理人员、其他工作人员对店面及其周边市场情况的了解，没有充分利用到不同类型、不同位置工作人员的具体能力素养。

对于自由程度较高的连锁店、加盟店或个体户，实际商品文案与广告的编写、商品备货策略的制定、市场趋势的预测和最终商业决策的施行一般均由该店面所对应管理者（如店长、店主等）进行。明显地，这对相关人员的文学素养、计算机等数字设备使用的熟练程度和“洞察力”（在特定情况下推理、判断并反映事物之间的因果关系的能力）提出了较高的要求。

该部分商家端 AI 功能，如图 5.1 所示，致力于缓解甚至根治这些问题。

5.1 商品设计辅助

为了缓解从业者在执行传统零售数字化的过程中遇到的难以编写高质量商品文案的问题，该设计包含一项特色 AI 功能：商品设计辅助。从业者提供商品对应的图

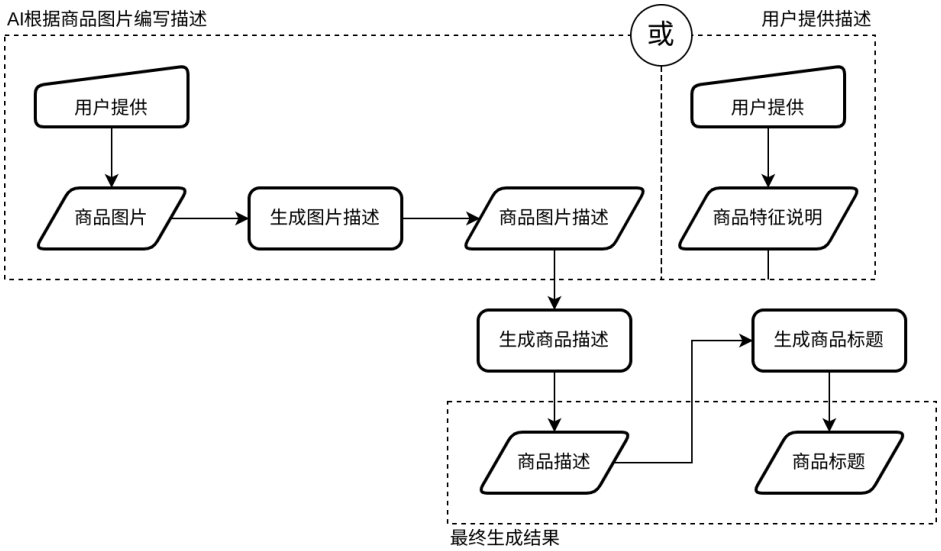

图 5.2 AI 商品设计辅助的执行流程图：其中上部两个的虚线框中内容所示不同输入和执行过程理论上可相互替换，本设计采用左侧部分的流程。

片，也可以继续提供额外的参考资料，而后 AI 便可以根据这些信息产生对应的商品资料，从业者既可以根据实际情况直接采用生成结果，也可以在结果的基础上进行人工修改，较为具体的执行过程如图 5.2 所示。

5.1.1 编写文案

与本设计的执行流程，利用 AI 编写出高质量的文案的条件可以归纳为以下几点：

- LLM 类型适合商业宣传性文案的编写
- LLM 上下文窗口（context window）足够大
- LLM 文字理解、逻辑推理、文学性能足够优越
- 使用的 LLM 具备多模态（multi-modal）特性或所使用的多个 LLM 中包含具有该特性的模型
- LLM 具备足够高的节制度（moderation）
- 具备多模态特性的 LLM 图像细节（包括文本）的识别能力足够好
- 提示词（prompt）诱导 LLM 生成正确的、可自动处理的回复
- 提示词诱导 LLM 从图像提取更多信息
- 提示词最大程度上限制 LLM 在结果中包含臆想（hallucination）内容

对于文本生成的部分，笼统地说，任何适用于一般用途（也就是不是为某特定专业用途设计）的 LLM 都一定程度上具备编写商品文案的条件，只要合成合理的提示词或用于补全的文本就可以使用。本设计所对应的应用程序实现使用了阿里巴巴公司于“百炼”生成式 AI 服务平台提供的`qwen-plus` 商业大语言模型。若选用的 LLM 为进行对话（Instruct）进行过调优（如`qwen-plus`），那么比较合理的输入是与“文案编写助手”的对话。本设计程序中生成商品描述提示词对话如下：

系统：你是一个处理商业资料的智能助手，不会输出问题答案以外的任何信息。

用户：请根据以下提示编写可以吸引到潜在客户的网店产品介绍文案，400 字以上，并且整理成适当长度的自然段。但是请尽可能不要对图片没有提及的商品情况和商店服务作出假设。以下是该产品图片的描述：（*产品图片描述*）…

（提取图片文字时）…以下是从这个图片中发现的文字（若图片中没有文字，则可能会是无意义信息）：（*产品图片文字*）

（提供额外上下文时）…以下是店主附加的信息，请参考：（*附加参考材料*）

从图 5.2 可见用于编写商品描述的材料可以来源于用户直接输入对目标商品的描述性文字，和交由额外模型处理的图片输入具有相似的效果。但是，采用这种输入方式的文案编写过程对营业者提出了新的要求，某种程度上相当于将对文学素养、宣传材料撰写能力的要求化为了对产品特征表达能力的要求，虽然同样是对“高门槛”问题的缓解，但不如输入图片的处理方式彻底[2]。

如果使用性能较为强大的多模态模型（如开源的“千问”系列 `Qwen2.5-VL` 模型），实际上可以将图片理解、描述的步骤和实际商品详情生成的步骤整合。在本设计的实现中考虑到大规模的多模态模型云服务对应的成本较高，并且输出速度一般慢于仅文本的大模型，故将两个步骤拆分，运用不同的模型处理。该设计中运用的视觉理解多模态大模型为“百炼”的商业大模型 `qwen-vl-plus` ，程序中生成商品图片描述的提示词对话如下：

系统：你是一个处理商业资料的智能助手，不会输出问题答案以外的任何信息。

用户：

（商品图片数据）

（文字信息）该图片中包含一个商品，请用尽可能详细的语言描述该商品的外观特点，以便于后续无法查看该图片的其他模型处理。

此外，程序中从商品图片中提取文字所使用的模型为“百炼”提供的商用特化模型 `qwen-vl-ocr` ，提示词对话如下：

[2]实际上，输入图片同样会对用户提出图像质量把控的要求。但明显地，一般用户满足该要求的概率相对而言是更高的，并且学会拍摄清晰照片的难度也更低。

系统：你是一个处理商业资料的智能助手，不会输出问题答案以外的任何信息。

用户：

（商品图片数据）

（文字信息）该图片中包含一个商品，请发现该商品外观中包含的所有文字。但如果没有发现文字，也请如实回答。

5.1.2 建议定价

对于一些特殊的零售细分行业、比较不常见的店面地段和无标准或习惯定价的商品种类，商品的最优定价并不明显。针对这一类较为特殊的情况，该设计提供一个参考定价生成功能，可以利用前文提及的商品描述和（可选的）附加参考材料来生成一个用于参考的商品价格。鉴于定价的过程逻辑性比较强，虽然该过程理论上可以采用一般大模型作为生成价格的算法，此处作为特殊情况采用了具备内省[3]（reflection）能力的大语言模型。这类模型中十分知名的是深度求索公司发布的开源模型“DeepSeek R1”，但为了缓解这类模型输出结果较为缓慢（模型较大并且具有思考过程）的问题，该设计采用（云上托管的）阿里巴巴发行的开源大模型“QwQ”。

显然，该过程需要输出的是价格的表示（于第 4 部分中提及的 RmCore 模块中的 Price 类型），但由于众所周知大语言模型的输出格式是较为随机的，此处采用许多常见文本模型具备的将输出规范为特定结构的 JSON 对象的功能。经过格式优化适合浏览的提示词对话如下：

系统：你是一个处理商业资料的智能助手，不会输出问题答案以外的任何信息。

用户：请根据以下信息猜测该商品的价格。输出格式：

```
{
    "unit": 单位个或单位克
    "price": 单位价格人民币分
    "pricing": 计价方式
}
```

例如“可乐一罐卖 3 元”表示为

```
{"unit": 1, "price": 300, "pricing": "Package"},
```

“苹果一斤卖 6 元”表示为

[3]即所谓“深度思考”（deep thinking）功能

`{"unit": 500, "price": 600, "pricing": "Weight"}`，

还请不要输出任何无关内容。以下是该产品图片的描述：（*产品图片描述*）…

（提取图片文字时）…以下是从这个图片中发现的文字（若图片中没有文字，则可能会是无意义信息）：（*产品图片文字*）

（提供额外上下文时）…以下是店主附加的信息，请参考：（*附加参考材料*）

由于 JSON 格式的技术限制，`pricing` 枚举类型部分无法限制大模型的输出为合法值。桌面商家端应用程序此部分在后期反序列化进一步进行格式检查。

5.1.3 提供额外参考

前文提及用户可以在生成商品详情、标题和（可选地）价格时可以选择提供额外的资料给 AI。因为在执行这些过程的时候，实际上大模型并不知悉门店的目标客群和运营情况、不清楚从业者的盈利目标，也不知道目前社会经济情况和特定商品的具体细节，但所有细节都将成为生成这下结果的重要参考。因此，这个特性是用户自主调优模型输出的重要的，也是较为易于理解的手段。

5.2 营业数据分析

为了缓解从业者在经营过程中需要分析销售数据，但是却不完全具备数据挖掘专家的专业能力的问题，该设计包含另一个特色 AI 功能：智能销售数据图表分析。分析界面软件将收集系统中与时间有关的各项数据指标，并通过图表展示其长时间的趋势、短时间的规律，而后利用 AI 解读图表蕴含的规律，提出合理的经营建议，而经营者可以通过参考 AI 给出的分析文字来轻松得知运营的规律和趋势情况，更可以以 AI 给出的建议为进一步规划营业计划的基础。

营业数据种类是多样的，分析的方式也同样种类丰富。为此，本设计中定义“营业数据”为零售商业运营的过程中与特定事件点（一般精确到秒或分钟）相关联的相互无直接关联的数值数据，比如顾客结算产生营运收入的事件对应的时间-数值（收入）元组，或者门店进货产生费用的时间-数值（支出）元组。通过以不同的时间间隔对这些数据进行整理、分类或累加，可以得出适合用于制图的数据。

5.2.1 图表生成

利用足够高的上下文长度，一定长度的原始数据可以以一般自然语言编码并被直接添加到模型的输入之中。然而这种处理方式在较少的数据（如一个月之间的收入）上就会占用大量的词法标记（lexical token），造成较为高昂的成本，因而这种处理方式在处理较多的原始数据的适合是并不实用的。并且这种处理方式并不能对用户给出较为直观的数据预览，仅适用于自动的分析方式。另一个向 AI 提供图表而不

是原始数据的好处是将时间数列（time series）转化为二维平面上位置（折线图）、位置-强度（热力图）或其他统计学模型的表示，相当于将一部分繁重的统计任务从 AI 转移到了专用的表格数据、数值数据处理模块，理论上有助于 AI 对数据的深入分析。

为了使从业者能够直观的看到数据长期的趋势和短期的规律，也为了便于其后 AI 对数据进行进一步的分析，本设计的统计应用程序对任何输入到系统的时间序列数据，统一利用 Python 编程语言、matplotlib 制图库生成如下几种图表：

名称	x 轴	y 轴	强度	分析模式
周规律热力图	小时	星期	数值	观察后思考
月规律热力图	小时	月份	数值	观察后思考
周框图	星期	数值	—	观察并思考
月框图	月份	数值	—	观察并思考
日期折线图	日期	数值	—	观察并思考
周折线图	周数	数值	—	观察并思考

表 5.1 提供用户浏览和 AI 推理分析的图表列表

其中“周规律热力图”、“月规律热力图”可以形象地展示出对于数值每日的分布规律与一周中不同天或一年中不同月份的关系。用户（和 AI）不但可以从中观察到数值在一天内的变化规律，还可以观察到这个规律本身根据不同条件发生的变化。借此可以推测用户的购买习惯和零售门店的营业情况，并据此制定营业策略。两个框图分别展示按星期和月份整理的对应数值的分布情况。在框图上可以直观地看到数值的 $\frac{1}{4}$、$\frac{3}{4}$ 分位值、中位数等关键统计值。据此可以较为容易地比较不同星期或月份数值的分布规律的区别。最后的两个折线图则是按照（用户）选择的时间段分别按天或按周整理的数字变化折线图。

5.2.2 图表理解

本设计图表理解部分针对不同的图表类型采用两种复杂程度不同的分析流程：

- **先观察再分析（stare then ponder）：**先利用多模态大模型进行图表关键内容、规律的提取，再将提取结果输入到适于处理文本、逻辑推理的大语言模型来分析因果关系和生成营业策略建议。
- **边观察边分析（stare and ponder）：**利用具备推理能力的多模态 LLM 在一轮对话之内完成营业策略建议的生成。

明显地，“先观察再分析”（下称“先观察”，另一种方法同理）的方法需要的时间是显著更长的，但也有机会可以降低在多模态模型上花费的额外成本并利用专门的文本大模型提高生成效果。本设计所对应分析程序利用“百炼”平台托管使用 dashscope 应用程序接口的 `qwen-vl-max-latest` 商业模型作为多模态模型，。

因为两个热力图各包含了三维（x、y 和强度）的数据，密度较高并且逻辑比较复杂，此处使用“先观察”方式来尽可能达到最优效果。观察部分的提示词如下：

用户：

（图表图片数据）

（文字消息）这幅图展示了（*前缀*）（*数值标题*）和（*变量甲*）、（*变量乙*）的关系（*附加内容*），请仔细描述图像内容和规律

其中“前缀”是在针对多数值的元组（比如销售记录可以是时间和销售件数、质量和收入），对同一组（也就是具有同样的时间）的数值使用的共享的标题，也可以包括用户或系统对零售实体一般情况的描述，用以向 AI 提供更多内容提示和引导，“附加内容”同理。若指定的统计数值只有一个变量，则“（*变量甲*）、（*变量乙*）”可以改为“（*变量*）”。

5.2.3 营业策略建议

由于图表内在因果逻辑的分析、营业策略的推测和建议是对逻辑推理能力需求较高的任务，本设计该部分使用带有内省能力的 `qwq-plus` 商业模型作为文本模型，“先观察”分析部分的提示词如下：

用户：以下是一张展示了（*前缀*）（*数值标题*）和（*变量甲*）、（*变量乙*）的关系的图的描述（*附加内容*），请请根据这段描述给出改善营业水平的推论和建议：（*图表图片描述*）

与上文提取图表信息相似，“边观察”方法的提示词如下：

用户：

（图表图片数据）

（文字消息）这幅图展示了（*前缀*）（*数值标题*）和（*变量甲*）、（*变量乙*）的关系（*附加内容*），请分析并给出建议

5.3 库存审计

在语义上来说，零售管理系统服务器中的“库存”（也就是第 4 部分中提及的 RmCore 模块中的 Asset 类型）是在仓库中或者货架上的商品的代表，但实际上因为各种原因二者之间的同步可能会出现偏差。一旦库存信息失去其应有的代表性，营业活动便也就难以继续正常开展。因此，需要对库存中的货物进行审计[4]，其中对许

[4]审计还包括对货物的状态、对于资金定额的分析等过程，实际上更多地是财务方面的事务。然而，本设计仅关注其“点货”的方面，为了书面性使用该笼统说法，特此说明。

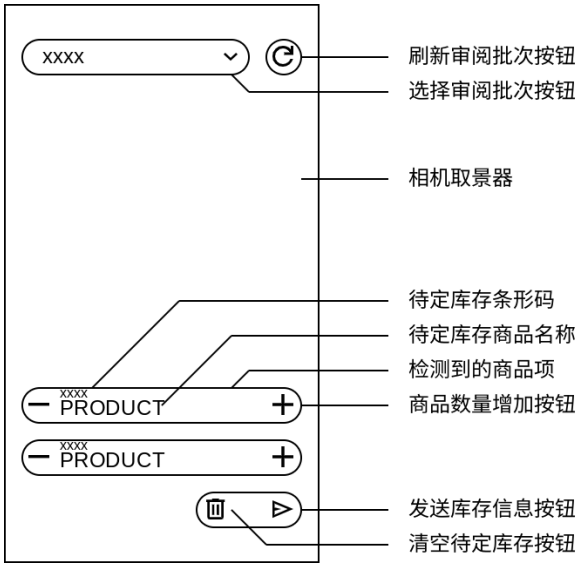

图 5.3 商家移动端商品审阅功能界面设计

多零售店面最为重要的库存清点（俗称“点货”）同时也是重复性最强、最为繁琐并且易于出现失误的操作。

为了最大程度减轻从业人员进行货物清点的负担，该设计商家移动端应用程序包含带有 AI 条码识别功能的智能货物清点模块。该模块采用 Google MLKit 机器学习算法库的条码检测器（barcode detector）功能开发，具有识别多种条码（包括本设计采用的 EAN-13 商品条码）的功能，并且可以通过如图 5.3 所示的简明易懂的 UI 界面操作，并将结果上传至服务器。从业人员可以在第 4 部分提及的 RmAdmin 商家桌面端应用程序中管理库存审计的批次和查看具体的审计情况。

6 顾客端 AI 功能

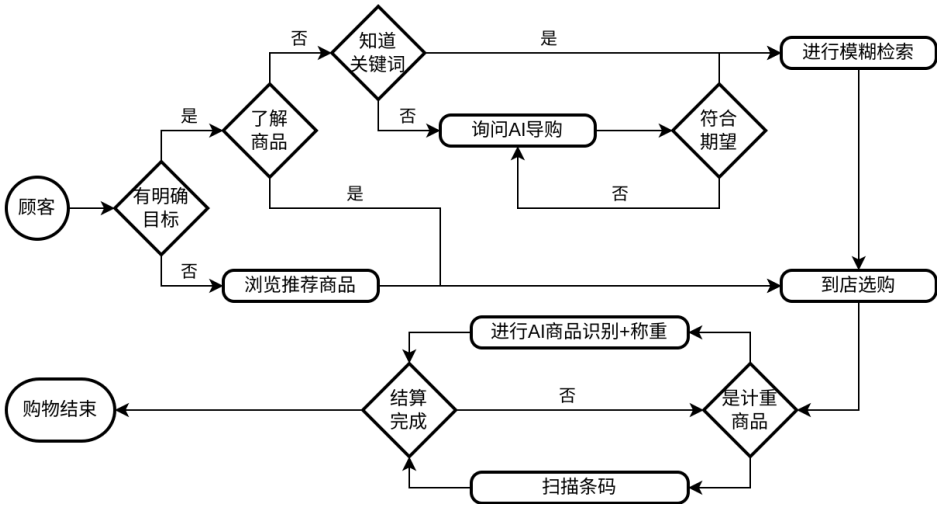

图 6.1 最终消费者商品选购流程

面向最终消费者的 AI 功能主要围绕着发现心仪商品和结算所购买商品的场景（如图 6.1 所示）展开。顾客发现商品的方式大致可以分为目的导向的和非目的导向的商品发现方式，其中非目的导向的商品发现方式与商户的宣传手段及其有效性关联较大，很大程度上取决于顾客是否浏览到该商铺对应宣传内容（实体、在线广告、客户群等）和对于宣传内容的实际吸引力。宣传内容的曝光可以由从业人员对社交媒体的参与来实现，而宣传内容可以借助上一部分提及的 AI 商品文案起草特性来辅助。目的导向的发现方式更为复杂。

目的导向的商品发现方式主要包括用户对商品进行搜索的过程。对于叫法比较单一、名称好记没有歧义的商品，简单分词—匹配的关键词搜索功能是可以满足需要的。然而，时间情况下商品名称匹配的问题可能远复杂于理想的情况。例如笼统和详细说法的区别：“可乐”和“苏打水”都可以叫作“汽水”，但这三个词语之间却无法直接相互匹配，并且若是为此将“汽水”拆为“汽”和“水”，不但仍然无法和“可乐”匹配，还可能会误匹配到与如“水”、“水汽”等词语相关的其他商品。

为了在一定程度上解决该问题，本设计包含模糊搜索引擎项目“探寻”及其对应词典处理脚本。该子项目采用“结巴”分词库[15-16]进行分词，并且利用大语言模型对每个词语的近义词进行枚举，最后将各个来源的处理结果整理为高查询效率的格式在为中文优化的自定义搜索算法中进行部署，以此在消耗比较少的计算资源的情况下达到较高的搜索速度和（中文）搜索的准确率，有助于最终消费者更好地进行目的导向的商品发现活动，推动消费体验、营业质量提升。

然而性能更加强大的模糊搜索系统无法解决在许多情况下顾客不知悉需要搜索的关键词（及其近义词）的问题，这种情况下，顾客可能甚至并不清楚自己实际需要

的商品。这个问题较为明显的解决思想是使得“商品发现”相关功能具有理解消费者对其需求的描述的语义并将需求内容对应于特定商品信息，或者为此生成对应的搜索语句提供给用户进行检索（或自动运行检索）。

为了解决这种情况带来的问题，该设计的顾客端移动应用程序包含 AI 导购助手模块。该模块利用经过特定提示词引导的多轮 LLM 对话及单次 LLM 调用，分别营造与顾客进行导购交流、导购向用户提出购买建议，如此往复的体验；从导购对用户的回复中提取出适用于检索的关键词语句，以此实现消费者只要合理形容需求，便可检索到对应商品的功能。

AI 识别计重商品结算模块是该设计对一般传统实体零售流程的另一个改进。通过利用 AI 物体识别算法，在零售管理系统中整合物体识别 AI 模型的训练数据采集、标注等操作对应的用户界面，自动化模型训练和部署的过程；在结算终端中整合 AI 物体识别前端软件及摄像头、质量传感器等硬件来实现营业者轻松部署 AI 物体识别模型，最终用户轻松自助结算计重商品，去除计重商品结算过程对店员参与的要求。

6.1 导购助手

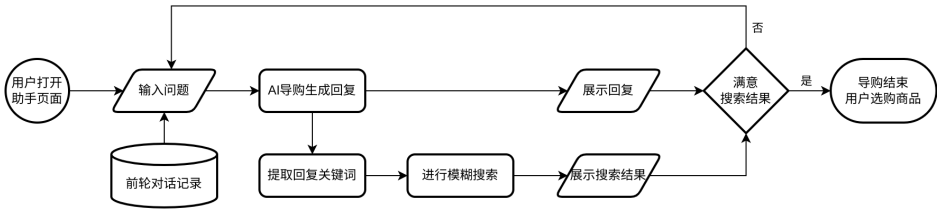

图 6.2 最终消费者导购操作流程

导购助手工作流程如图 6.2 所示，主要包括以下两个部分：

- **对话式 AI 导购专家：**通过与最终消费者的一轮或多轮对话确定消费者的具体需求，并给出相应的购买建议（商品类型、名称等）。
- **搜索关键词猜测算法：**通过利用大语言模型强大的文字处理能力，使用 AI 导购的输出产生出对应的搜索关键词。

6.1.1 对话型生成式 AI

AI 导购专家实质上就是扮演导购身份，可以与用户进行多轮对话，解决用户困扰的人工智能聊天机器人（chatbot）。为了使得输出较为中性的无（行业相关）微调的一般大语言模型输出符合“导购身份”的回复，而不是一般的建议。该设计主要采用“百炼”的商业大模型 `qwen-turbo` ，开发了以下的系统、对话提示模板：

系统：You are a helpful assistant of a retail shop that advise about buying stuffs.[5]

[5]此处为开发方便（并且遵照“百炼”官方文档中系统提示的风格）使用了英文系统提示，但实际

（对每轮对话重复：）

用户：（*提出问题*）

模型：（*回复用户提问*）

利用这样的对话模板，AI 导购助手可以与用户进行（上下文长度范围内的）任意多轮对话，并且每轮对话之间可以产生关联（可以视为大模型聊天机器人的短时记忆特性），从而提高向用户提出正确推荐的概率。同时也因为对话记录参与下一轮对话的特性，应用程序带有由用户手动重置对话的功能，以此避免不同主题、结论的对话记录对新一轮对话大模型推理的干扰。

6.1.2 搜索关键词提取

在每一轮对话 AI 进行回复之后，相应的回复将会作为另一套提示词的一部分输入到模型中，以进行搜索使用的关键词的提取。在该设计对应实现中该特性使用的大模型为“百炼”的商业大模型 `qwen-turbo` 。提示词对话如下：

用户：请根据这段话生成一些搜索商品的关键词，并且不要输出任何无关内容：（*该轮对话中大模型的输出*）

理论上该过程可以和前文提及进行对话的过程进行合并，但该设计在此次选择将二者分开的设计方式，主要是有两个考量。首先是因为大模型输出难以控制格式的问题，此处避免需要大模型根据 JSON 等特定的格式输出结果（语义上就是输出两个不同的字符串），以此减轻对大模型推理、服从指引能力造成压力。其次是通过将大模型的输出输入到该任务的大模型（可以为同一个）之中，截断了单次全部输出的模式的标记（token）之间的线性关联，使得关键词的输出不直接受到用户原始输入（和前轮对话）的影响，从而一定程度上提高可预测性和防止模型受到用户特定语义不明确的输入而产生意外输出的问题。

6.1.3 商品搜索

大模型输出的关键词被用于模糊搜索，搜索结果连同用户的原始输入和 AI 导购的答复将经过如图 6.3 所示的用户界面展示给用户。值得注意的是，前文提及的使得大模型提取 AI 导购回答中搜索关键词的输入之中并无对输出（搜索关键词）格式的规定。这是因为将在后文提及的模糊搜索算法是为词组而不是句子优化的，对实际上搜索语句的格式并无要求，仅仅只需要语句中包含期望的关键词。即便大模型忽视了引导（“关键词”）而输出了相关的连贯句子，搜索步骤也可以正常进行。

测试中 AI 发挥了语言中性的能力，正确地使用中文响应了中文编写的输入。若需要中文提示，可以使用：你是零售商店的一个乐于助人的导购，你向客人提出购物建议。

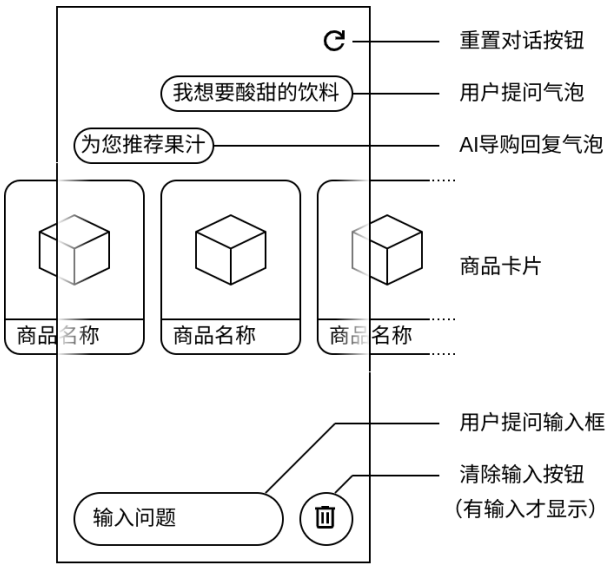

图 6.3 AI 导购用户界面

6.2 称重商品识别

在许多类型的零售细分领域中，不可避免地将会遇到按重量计算，而本身并无包装（俗称“散装”）的产品，这些商品包括但不限于在生鲜蔬果、米面粮油和零食糖果等多种类别中的产品。实际上在社会上的许多商场和超市中，消费者在选购这些“散装”商品的时候，由自己完成的步骤只能持续至商品的挑拣和包装。最为重要的称重环节必须由店员帮助完成，并且因为明显地许多这些商品本身无法贴上条码或其他标签，店员需要在容器上粘贴同时记录了商品类型和质量的“静态标签”或与结算系统联网同步的“动态标签”。这种方式既有重复度较高、专业性较低的人工辅助参与，又需要非标准（EAN-13）的商品信息传递记录手段。

为了缓解这个问题，本设计的结算系统包含利用摄像头和 AI 图像识别技术的智能计重商品结算功能，还配套用于收集、标记训练用数据的应用程序以简化识别模型的创建过程。借此称重设备的功能可以被整合到结算设备之中，使得用户可以轻松自主结算按质量计算价格的商品。

6.2.1 数据准备

如图 6.4 所示，从业者可以在“EasyDataset”应用程序内管理多个不同数据集。在应用程序内打开数据集之后，用户分别可以在“预备”、“拍摄”和“挑拣”三个标签页之间选择需要执行的操作。这三个操作一般情况下为递进顺序。

通过“预备”功能从业者可以准备需要 AI 识别模型检测到的商品类别列表，其中每个类别具有（隐藏的）类别编号、自身的名称和对应的商品编号。名称将被用于其余两个步骤中选择类别的参考，而商品编号将被用于模型实际部署的场景下结算系统将识别到的类别与具体商品联系起来的过程。此外，点击已经创建完毕的类别

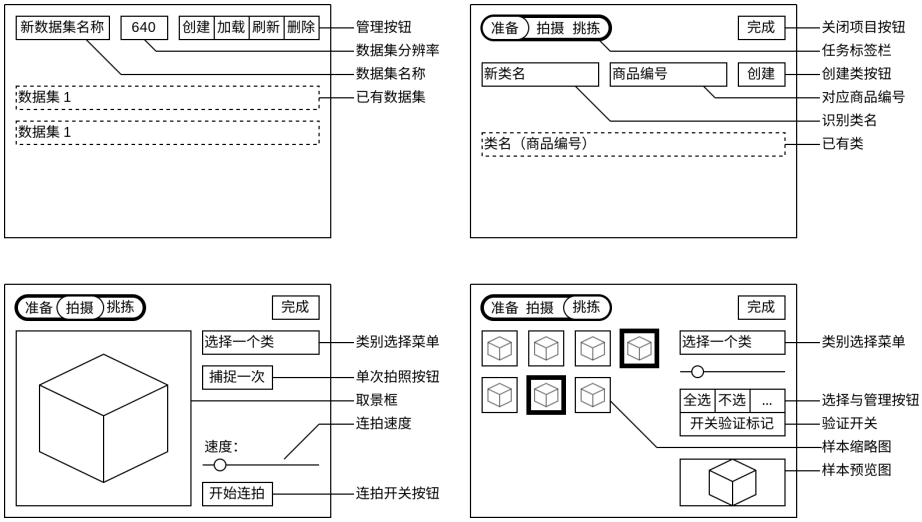

图 6.4 “EasyDataset”数据集管理应用程序用户界面

可以对其名称或者对应的商品编号进行修改。

使用“拍摄”功能可以为选定的类别拍摄图片。用户首先需要选择对应照片将保存到的类别，然后可以通过点击按钮来完成单张照片的拍摄。为了便于从业者在短时间之内拍摄大量不同角度的照片以提高识别模型的判别性能，该应用程序具有连拍功能。用户可以通过可拖动的调节组件调整连拍速度，其后通过点击按钮可以开始连拍。连拍的过程中用户可以不断调整商品的展示角度、摆放方式，而不需要关注用户界面方面的操作，再次点击按钮可以停止连拍。

针对选中类别，“挑拣”页面将展示已经拍摄的照片。在这个页面，从业者可以批量审阅拍摄完成的样本并从中去除图像素质不符合期望者，点击多个图片可以进行多选批量操作。此外，用户可以通过选中图片之后点击验证开关选择将一部分图片标记为模型效果验证（validation）用的图片，有助于在训练之后检测模型的泛化能力和预计效果。

6.2.2 图像处理

该应用程序并不直接保存拍摄的照片。为了考虑到训练和部署环境中可能的区别及数据集在不同版本的商品识别后端（或未来的新版本 AI 商品识别服务）之间的互换性（interoperability）和兼容性，数据集其中的商品图片统一裁切为以数据集定义时指定的边长的正方形。图片先被以二次线性（或其他更好的）插值算法重采样到短边与指定边长一致的相同长宽比新分辨率，再裁切其中央位置的最大正方形作为最终结果。这样，画面的内容被最大程度保留的同时对模型输入格式的要求较为宽松，并且通过（可选地）重采样到较低分辨率，训练的时间成本可以得到有效的控制，进一步降低使用门槛和维护成本。

6.2.3 AI 图像分类

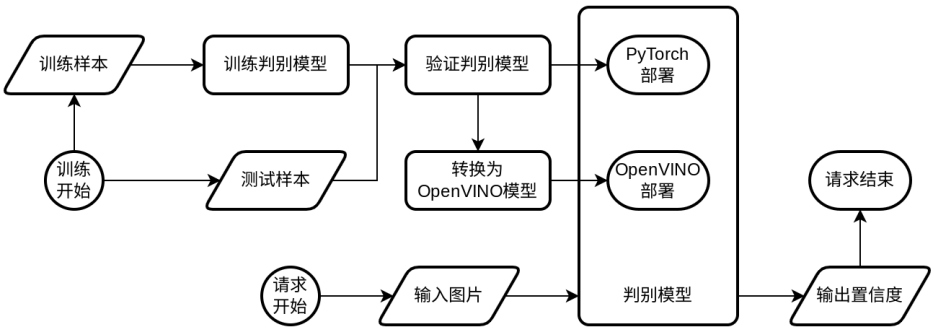

图 6.5 “yolod”图像分类训练-推理服务

该设计中提供 AI 图像分类、分类模型训练的服务“yolod”采用由 ultralytics 开发的 YOLO11（版本 `yolo11n-cls` ）图像分类模型[17]作为模型架构，利用在 CPU 或（可选地）多种不同类型 GPU 上均可加速运行的 PyTorch 深度学习框架（版本 2.6）和 `ultralytics` 一体式 YOLO 开发辅助库进行训练和推理任务，利用 FastAPI 封装推理过程为 HTTP API，结算终端应用程序只需要将编码过后的图片发送到 yolod，服务器便会自动完成后续处理、推理工作并发回判别结果。

然而，直接使用主要为研究、开发场景优化的原生 PyTorch 模型进行推理任务效率是不够理想的，常常无法达到商品识别实时性的需要。为了解决这个问题，yolod 包含透过`ultralytics` 展开的利用英特尔 OpenVINO 模型优化部署技术执行的推理后端，可以极大地提升推理的效率[6]，最大程度利用 CPU 或服务器所配置的兼容的加速硬件。值得一提的是，本设计鉴于特定的实验环境作出了使用 OpenVINO 的决定，但理论上任何有助于模型运行速度提升的部署方式（如 ONNX、CoreML 等）均可按情况进一步开发，投入使用。

6.3 模糊搜索

本设计所包含模糊搜索引擎“探寻”是一个以分词技术为基础，以大语言模型生成近义词作为词典拓展手段开发的、为中文优化的文本关键词模式匹配子系统。该子系统包括用于商品数据处理、AI 近义词搜寻、词典构建的脚本和部署在第 4 部分的搜索算法四个部分。

[6]前期实验中 OpenVINO 上部署的模型相比 PyTorch 在同样利用英特尔 Arc A770 16GB 加速硬件推理的情况下展现出了约 11 倍左右的性能提升。作者推测其中一部分提升幅度是 PyTorch 的 SYCL（oneAPI Level Zero）后端在测试版本上缺乏优化引发的。

6.3.1 商品词典

商品的搜索实质上是检测商品的标题或说明之中是否含有搜索语句相关模式（词语），同时匹配程度（模式命中次数）较高者更可能符合搜索语句对应语义。在这种情况下，任意一次命中于原文中的位置语义上并无作用，故输出匹配位置的字符串搜索算法并不适合该用途。同时，这也意味着可以快速检测模式是否存在而不包含模式位置相关信息的词典（dictionary）是最适合该用途的数据结构。为了更好传达算法的思想，现列出并解释下文将要直接使用的一些操作和类型：

- F_p **过滤（操作）：**该操作接受一个字符串，并返回该字符串移除除了 Unicode 规范[18]所定义的拉丁文字（Latin）、汉字（Han）、假名（Hiragana、Katakana）和谚文（Hangul）以外的文字的版本。
- F_e **扁平化（操作）：**该操作接受一个由同元素类型可空集合组成的可空集合，并返回一个由所有原集合中子集合所包含的元素组成的新集合。
- F_c **分词（操作）：**该操作接受一个字符串，返回字符串的词语集合。
- **词典（类型）：**以字符串（词语或标记）为键，以*以编号数值为键，以频率数值为值的映射*为值的映射类型。

以下为根据标记（token）集合生成本模块对应词典数据的算法：

输入：字符串的集合的序列 x
输出：词典 y
$y \leftarrow \emptyset$
for each s **in** x **of index** i **do**
  for each c **in** s **do**
    if $\nexists y(c)$ **then**
      $y(c) \leftarrow \emptyset$
    end if
    if $\nexists y(c)(i)$ **then**
      $y(c)(i) \leftarrow 0$
    end if
    $y(c)(i) = y(c)(i) + 1$
  end for
end for

可见该算法遍历了每个条目（商品）对应的字符串（关键词），并对每个关键词在不同商品中的出现进行计数，以此来统计每个关键词与不同商品的关联程度。值

得注意的是，该算法对每个商品的输入要求是关键词的集合，根据每个商品的名称和说明生成词语序列的算法如下：

输入：商品名称和说明的二元组的序列 x
输出：字符串的集合的序列 y
$y \leftarrow \emptyset$
for each (a, b) **in** x **of index** i **do**
  $s \leftarrow F_e(F_c(a) \cup F_c(b))$
  $s \leftarrow \{F_p(x) \mid x \in s\}$
  $s \leftarrow \{x \in s \mid x \neq \emptyset\}$
  $y \leftarrow y \cup \{s\}$
end for

由此可知该算法对商品的名称和说明都进行了分词操作，并将分词结果整合为一个序列，按输入的（商品）顺序进行分类，便于后续处理。

6.3.2 AI 近义词搜寻

为了缓解上文提及的搜索算法无法理解搜索语句（及其单独词语）的语义从而无法搜索到意思相关的词语的问题，此处设计利用大语言模型优秀的自然语言处理能力进行前文提及的词典的拓展。考虑到词语数量可能较为庞大，云端处理的成本是较大的，此处使用本地部署的 `qwen2.5-3b` 模型，提示词对话如下：

用户：输出这个词语的所有近义词为 JSON 字符串集合，不要输出无关内容：

（原词语）

通过在所有词语上重复该操作，并指定模型输出格式（schema）为 JSON 字符串数组，可以获得每个词语对应的近义词集合。此外，通过多次执行在所有词语上的遍历可以增加获得到的近义词集合的大小。值得注意的是，这种方法一定程度上依赖于模型对指令的遵从能力，并且生成的词语可能需要进一步分词、过滤操作才能用于后续处理。

6.3.3 近义词词典

现定义类型“近义词词典”为从词汇到其近义词集合的映射。为了便于实现搜索算法，现定义任何一个词语都是其本身的近义词，因此近义词词典的初始化方法是获取词典中的全部词语，并构建一个对全部词语，以该词语作为输入得到只有这个词语一个元素的集合的映射。生成近义词词典算法如下：

输入：AI 模型生成的近义词词典 x_1 ；词典 x_2

输出： 近义词词典 y

$y \leftarrow \emptyset$

for each key k **in** x_2 **do**

  $y(k) \leftarrow \{k\}$

end for

$\hat{y} \leftarrow \{x \mid \exists y(x), |x| \geq 4\}$

for each key k **in** y **do**

  $\hat{y}_k \leftarrow \{x \in \hat{y} \mid k \supset x\}$

  $y(k) \leftarrow y(k) \cup F_e(\{y(x) \mid x \in \hat{y}_k\})$

end for

for each key k **in** y **if** $|k| \geq 4$ **and** x_1 **do**

  for each v **in** $y(k)$ **do**

    if $\nexists y(v)$ **then**

      $y(v) \leftarrow \emptyset$

    end if

    $y(v) \leftarrow y(v) \cup \{k\}$

  end for

end for

从算法上可以看到，首先 y 被初始化为词语到其本身的映射，而后对任何字节长度大于 4（一般可视为大于两个中文字符）的字符串，任何被其包含的其他词语对应的近义词都会被添加到该词语对应的近义词集合中（以弥补潜在分词不够细致的问题）。而后同样对任何字节长度大于 4 的词语其本身都将被添加到其每个近义词对应的近义词集合中，同时将会将这些词语添加为 AI 生成的近义词词典中对应近义词的近义词集合中。

6.3.4 搜索算法

在部署到服务器的搜索算法中，顾客发出的搜索语句首先会被分词和过滤，经过这些操作之后剩余的词语将会被用于实际匹配，搜索算法如下：

输入： 近义词词典 s ；词典 d ；搜索词集合 x

输出： 按匹配程度排序的商品编号序列 y

$f \leftarrow \emptyset$

for each w **in** x **do**

  if $\exists s(w)$ **then**

    continue

  end if

```
    a ← s(w)
    for each ŵ in a do
      if ∃d(ŵ) then
        continue
      end if
      m ← d(ŵ)
      for each key k in m do
        if ∃f(k) then
          f(k) ← 0
        end if
        f(k) ← f(k) + m(k)
      end for
    end for
  end for
  sort f by value
  y ← {x | ∃f(x)}
```

对每个词语，首先从近义词词典查询其对应的近义词列表（由前文可知该列表可能包含该词语本身），若无结果，则说明该词语在所有商品文字的词语（及其近义词）中尚未出现，可以被安全忽略。否则，针对每一个近义词若词典之中有相应的项目其中所有商品编号对应的词频将会被累加到一个映射中，最后根据总词频对其进行排序，输出排序后的商品编号序列。

7 实验

7.1 数据准备

为了更好地开展商品信息相关的实验、测试，现以 Product-10K 数据集[19]中的商品图片为基础，利用本地部署的多模态 LLM 批量生成这些商品对应的商品信息，再将这些信息输入到中心服务器中，以达到替代实际商品商品信息的目的。该数据集中的图片以存货单位（SKU）、商品类别为分类方式。在本实验中属于同一 SKU 的图片将被视为同一个商品的图片，每个 SKU 都会被随机挑出一个图片作为其代表交由 LLM 进行处理。文字部分利用的 LLM 是 30 亿参数版本 `qwen2.5` 。

本实验利用本地部署的 80 亿参数版本 `minicpm-v` 大模型进行商品图像描述的编写，使用的提示词对话如下：

> **用户：**
>
> > **（商品图片数据）**
> >
> > **（文字消息）**细致地描述这个图片中的内容，以便其他模型利用这段描述进行进一步推理。

然而，对生成结果（及其对应图片）的人工抽查显示数据集中某些图片更加接近生活照片，难以被断定为商品图片。因此，以下的提示词对话被用于检测图片是否代表一个商品：

> **用户：**以下是从一个图片产生的说明，请根据这段说明判断该图片是否表示一个商品（布尔值字段 verdict）：（*商品图片描述*）

以上提示词在使用的时候加入了对模型输出的限制（具有布尔值字段 `verdict` 的 JSON 对象）。经过筛选之后的商品图片描述其后进入商品描述生成的过程：

> **用户：**请编写商品描述，商品描述应该尽可能有吸引力，并且不要输出商品描述之外的内容。以下是商品对应图片的描述：（*商品图片描述*）

最后，生成的商品描述被用于进一步生成商品标题：

> **用户：**为这段商品介绍产生一个言简意赅的商品标题：（*商品描述*）

生成的商品标题、描述等数据其后被收集起来，并且与其图片相绑定。之后，这些数据被通过第 4 部分提及的服务器 API 传输到数据库中，用于后续试验。

7.2 商家端

7.2.1 桌面应用

7.2.2 手机应用

7.2.3 数据集管理程序

7.3 顾客端

7.3.1 手机应用

7.3.2 结算程序

8 讨论

9 结论

参考文献

[1] SAMUEL A L. Some Studies in Machine Learning Using the Game of Checkers [J/OL]. IBM Journal of Research and Development, 1959, 3(3): 210-229. DOI: 10.1147/rd.33.0210.

[2] RAINA R, MADHAVAN A, NG A Y. Large-scale deep unsupervised learning using graphics processors[C/OL]//ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning. Montreal, Quebec, Canada: Association for Computing Machinery, 2009: 873-880. https://doi.org/10.1145/1553374.1553486. DOI: 10.1145/1553374.1553486.

[3] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[J/OL]. Commun. ACM, 2017, 60(6): 84-90. https://doi.org/10.1145/3065386. DOI: 10.1145/3065386.

[4] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition [C/OL]//2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016: 770-778. DOI: 10.1109/CVPR.2016.90.

[5] VASWANI A, SHAZEER N, PARMAR N, et al. Attention Is All You Need [EB/OL]. 2023. https://arxiv.org/abs/1706.03762. arXiv: 1706.03762 [cs.CL].

[6] Introducing ChatGPT[EB/OL]. (2024-03-13) [2025-04-09]. https://openai.com/index/chatgpt/.

[7] 苏美文, 杨文爽, 李博文, 等. 推动人工智能与实体经济深度融合加快发展新质生产力[J]. 工业技术经济, 2025, 44(04): 32-59.

[8] 谢捷, 唐声羽, 陈柳钦. 数字化转型视域下人工智能驱动企业新质生产力提升[J]. 南海学刊, 1-15.

[9] RADFORD A, KIM J W, XU T, et al. Robust Speech Recognition via Large-Scale Weak Supervision: arXiv:2212.04356[EB/OL]. arXiv. (2022-12-06) [2025-04-09]. http://arxiv.org/abs/2212.04356. arXiv: 2212.04356[eess].

[10] 赵树梅, 徐晓红. "新零售"的含义、模式及发展路径[J/OL]. 中国流通经济, 2017, 31(05): 12-20. DOI: 10.14089/j.cnki.cn11-3664/f.2017.05.002.

[11] 杜睿云, 蒋侃. 新零售: 内涵、发展动因与关键问题[J/OL]. 价格理论与实践, 2017(02): 139-141. DOI: 10.19851/j.cnki.cn11-1010/f.2017.02.038.

[12] 廖夏, 石贵成, 徐光磊. 智慧零售视域下实体零售业的转型演进与阶段性路径[J]. 商业经济研究, 2019(05): 28-30.

[13] 王先庆, 雷韶辉. 新零售环境下人工智能对消费及购物体验的影响研究——基于商业零售变革和人货场体系重构视角[J]. 商业经济研究, 2018(17): 5-8.

[14] 齐欢欢, 惠银银, 郭洋洋. 浅析私域流量时代的直播电商运营[J]. 今日财富, 2020(24): 58-59.

[15] SUN A. fxsjy/jieba[CP/OL]. (2025-04-15) [2025-04-15]. https://github.com/fxsjy/jieba.

[16] Messense. messense/jieba-rs[CP/OL]. (2025-04-10) [2025-04-15]. https://github.com/messense/jieba-rs.

[17] JOCHER G, QIU J. Ultralytics YOLO11[CP/OL]. 11.0.0. 2024. https://github.com/ultralytics/ultralytics.

[18] CONSORTIUM U. The Unicode Standard, Version 16.0[M/OL]. 2024. https://www.unicode.org/versions/Unicode16.0.0/.

[19] BAI Y, CHEN Y, YU W, et al. Products-10K: A Large-scale Product Recognition Dataset[EB/OL]. 2020. https://arxiv.org/abs/2008.10545. arXiv: 2008.10545 [cs.CV].

致谢