**软件工程课程设计——智能化流浪动物救助与领养平台项目需求分析规约（SRS）**

软件工程课程设计——智能化流浪动物救助与领养平台项目需求分析规约（SRS）

需求规约版本表

1. 项目概述

1.1 项目背景

1.2 项目目标

2. 利益相关者

2.1 开发人员

2.1.1 当前主要开发人员

2.1.2 未来可能开发人员

2.2 审查项目人员

2.3 产品用户

3. 项目约束

3.1 Scrum敏捷开发模式

3.2 资金约束

3.3 用户样本约束

3.4 时间约束

3.5 操作环境约束

4. 项目范围

4.1 项目范围概述

4.2 主要任务

4.2.1 救助流程模块

4.2.2 宠物管理模块

4.2.3 盈利服务模块

4.2.3.1 疫苗套餐服务

4.2.3.2 宠物食粮服务

4.2.3.3 宠物保险方案

4.2.3.4 平台公益捐助

4.2.4 订单管理与收益统计模块

4.2.5 管理员后台支持模块

4.3 交付物

4.3.1 Web 网页端

4.3.2 系统文档与建模文件

5.需求调研

5.1 用户调研

5.1.1 调研目标

5.1.2 调研问题

5.1.3 示例问题

5.1.4 整体结论

5.2 相关从业者调研

5.2.1 调研目标

5.2.2 调研问题

5.2.3 数据结果

5.2.4 整体结论

5.3 调研报告数据总结

5.4 行业财报调研情况

6. 用户故事

7. 术语表

8. 相关事实与假设

8.1 相关事实

8.2 业务规则

8.3 假设

9. 需求建模

9.1 基于场景的建模

9.1.1 用例图

9.1.1.1领养模块用例图

9.1.1.2盈利模块用例图

9.1.1.2.1购买疫苗套餐用例

9.1.1.2.2购买宠物食粮用例

9.1.1.2.3购买宠物保险用例

9.1.1.2.4平台捐助支付用例

9.1.2活动图

9.1.2.1领养模块活动图

9.1.2.2盈利模块活动图

9.2 基于类的建模

9.2.1 领养模块分析类图

9.2.2 盈利模块分析类图

10. 接口需求

10.1 子系统接口

10.1.1 领养流程子系统接口需求

10.1.1.1 接口功能需求

10.1.1.2 接口特点

10.1.1.3 安全性需求

10.1.1.4 性能需求

10.1.2 盈利模块接口需求

10.1.2.1 接口功能需求

10.1.2.2 接口特点

10.1.2.3 接口安全性需求

10.1.2.4 接口性能需求

10.2 接口与原型图

10.2.1 领养管理模块原型

10.2.2 盈利模块原型

10.2.2.1盈利模块主界面模块原型图低保真

10.2.2.1盈利模块疫苗服务模块原型图低保真

10.2.2.1盈利模块食粮服务模块原型图低保真

10.2.2.1盈利模块保险服务模块原型图低保真

10.2.2.1盈利模块捐助服务模块原型图低保真

10.2.2.1盈利模块后台订单管理模块原型图低保真

11. 非功能性需求与解决方案

11.1 性能需求设计与解决方案

11.1.1 性能优化设计

11.1.2 实现方案

11.1.2.1前端静态资源优化

11.1.2.2数据库性能优化

11.1.2.3后端服务优化

11.1.2.4接口性能提升措施

11.1.2.5视频播放优化

11.2 安全性需求设计与解决方案

11.2.1 安全性设计

11.2.2 实现方案

11.3 可扩展性需求设计与解决方案

11.3.1 可扩展性设计

11.3.1.1模块化结构设计

11.3.1.2插件式功能扩展能力

11.3.1.3数据库垂直扩展能力

11.3.1.4内容投放与大模型接口对接能力

11.3.2 实现方案

11.4 可靠性与持久性需求设计与解决方案

11.4.1 可靠性设计

11.4.2 实现方案

11.5 兼容性需求设计与解决方案

11.5.1 兼容性设计

11.5.2 实现方案

11.6 平台计算资源使用设计

11.6.1 服务器资源使用设计

11.6.2 实现方案

12. 其他问题

12.1 开放问题

12.1.1 技术层面

12.1.2 用户体验层面

12.1.3 系统扩展与可维护性

12.2 解决方案

12.3 待办事项

13. 参考内容

**需求规约版本表**



1. **项目概述**

**1.1项目背景**

随着城市化进程的加速，流浪动物数量持续增长，相关管理机制却长期存在信息不透明、流程断裂、领养渠道零散等问题。许多拾主在发现流浪动物后无法获得统一的登记平台；寄养机构之间缺乏标准化记录体系；领养人难以获取动物的真实健康状况和寄养历史；同时，公益组织在运营与资源筹措方面也面临较大压力。因此，构建一个覆盖救助、寄养、健康记录、领养审核与公益捐助等多个环节的一体化平台，已成为提升动物福利、规范领养流程、提高救助效率的重要方向。

传统的领养流程主要依赖人工沟通与线下管理，不仅效率低，而且容易造成信息丢失、重复登记等问题。动物的健康记录难以追踪，领养人与寄养机构之间缺乏可靠的在线审核体系，导致领养决策不透明、寄养负担难以量化。同时，救助组织的运行成本不断增加，但缺乏可持续的盈利模式，使平台长期依赖志愿者或外部捐助，稳定性较弱。

本项目——《智能化流浪动物救助与领养平台》旨在解决这些痛点。通过将救助登记、寄养管理、健康档案、领养流程和盈利系统整合到同一个技术平台中，实现信息流、审核流与资金流的统一管理。平台将基于 Web 端构建完整的用户访问界面，并采用 Node.js + Express + SQLite 的轻量级后端架构，保证系统易维护、可扩展、结构清晰。与此同时，平台引入“公益型盈利模块”，通过疫苗合作服务、宠物食粮销售、保险合作方案和用户捐助等方式，为救助体系提供持续性的资金补充，使平台不仅具备公益价值，也具备实际运营能力。

综上所述，本项目诞生于社会公益需求、救助行业痛点以及技术可行性三方面的推动。它不仅希望提升动物救助效率，更致力于构建一个“人人可参与、全链条透明化”的现代动物福利系统，为未来更多的公益数字化项目提供参考与实践模板。

**1.2 项目目标**

本项目的总体目标是设计并实现一个集“救助登记、寄养管理、健康信息记录、领养审核”与“公益盈利模块”于一体的综合性流浪动物救助平台，为拾主、寄养机构、领养人及合作伙伴提供安全、透明、高效的数字服务。平台旨在实现端到端流程闭环，使动物从被发现、被救助到最终被成功领养的每一个环节都有据可查、有迹可循。

首先，从救助角度出发，平台目标是让拾主能够便捷地提交动物信息；寄养机构能够及时接收信息、更新寄养记录和健康状态；管理员能够对流程进行监管、审批与维护。通过标准化的数据模型与流程化的 UI 交互，系统确保各类信息在传递过程中不丢失、不重复，形成具有高度一致性的数据体系。

其次，在领养流程方面，目标是让领养人能够基于可信、完整的信息做出领养决策。系统会提供动物档案查询、健康记录展示、领养申请入口和审核结果反馈机制，使整个领养流程在线化、透明化。同时，通过记录用户申请行为和平台审核逻辑，有助于提高领养成功率与匹配效率。

最后，本项目特别强调“可持续运营能力”。因此盈利模块的构建也是项目的重要目标之一。通过整合疫苗合作套餐、宠物食粮产品、保险合作服务与公益捐助渠道，平台形成了公益与运营结合的新模式，使救助体系不再完全依赖单一捐助来源，而是具备自主造血能力。盈利模块的订单系统与收益记录系统将为平台提供可量化的运营数据，为后续扩展、优化和商业合作提供依据。

综上，项目目标旨在构建一个以公益为核心，但兼具可持续发展能力的智能化平台，使流浪动物救助流程更加人性化、数字化、可追踪，并推动动物保护事业向高效与规范化方向发展。

**2. 利益相关者**

**2.1 开发人员**

* + 1. **当前主要开发人员**

项目的核心开发团队包括以下成员：

马敏慧智（学号：2351707）

吴昀洁（学号：2352191）

* + 1. **未来可能开发人员**

项目计划在既有技术框架下进行持续迭代，未来可能申请校内大学生创新创业项目，由新团队进一步完善与扩展功能。

**2.2 审查项目人员**

黄杰老师：担任项目的审查者和指导老师，在 Scrum 项目管理中担任审查角色（Scrum Master）。负责监督项目开发进度，提供技术建议，并确保项目质量符合标准。

**2.3 产品用户**

拾主/普通用户：发现流浪动物后希望快速提交信息，促进救助流程启动。

救助机构/寄养点：包括合作宠物医院、救助站等，需及时接收救助信息、更新寄养情况并管理动物健康记录。

领养者：希望通过平台查看动物信息、提交领养申请、了解健康状况与领养流程。

平台管理员：负责对用户提交的信息进行审核，监督救助与领养流程的规范化运行，并管理平台盈利模块的合作信息与订单收益。

盈利合作方：包括疫苗服务机构、宠物食品供应商、宠物保险公司等，通过平台提供服务并与平台形成合作模式。

**3. 项目约束**

本章旨在明确项目在开发过程中面临的各类限制条件，包括开发模式、资金资源、用户样本、时间安排及系统操作环境等方面的约束。这些约束将直接影响项目的开发节奏、功能范围及最终交付质量，是制定合理需求与规划的重要依据。

**3.1 Scrum 敏捷开发模式**

本项目采用 Scrum 敏捷开发模式进行组织与管理。将项目拆分为多个迭代（Sprint），线上汇报工作进度，同时通过Github进行代码协作管理确保项目能够快速响应用户需求的变化，同时逐步交付功能模块并优化用户体验。

这种模式有助于提升项目灵活性与透明度，但也对团队成员的沟通效率、时间管理以及版本迭代速度提出更高要求。

**3.2 资金约束**

由于项目为教学项目，不依赖真实资金投入，开发过程中的所有功能实现需基于开源框架（如 Node.js、Express、SQLite、原生 Web 技术等），不能依赖付费组件、商业服务器、高级云服务等资源。避免功能过度复杂化，确保资源消耗控制在课程项目允许范围内。

未来若平台扩展为真实公益项目，则需重新评估部署成本、带宽需求与第三方合作费用。

**3.3 用户样本约束**

在项目开发过程中，团队无法获得大量真实救助机构、领养用户或合作品牌的实际行为数据。因此：需求调研主要依赖线上调研与假设情境分析；盈利模块的合作对象仅基于模拟情境（疫苗套餐数据、宠物粮食数据、保险方案等均为示例）；无法验证系统在大规模用户情况下的真实使用反馈。

**3.4 时间约束**

项目开发周期：2025年11月初~2025年12月中（两个月内）完成：

需求调研；

需求文档编写（包括 SRS、Proposal）；

系统建模（用例图、类图、活动图）；

前后端开发：功能测试与 Bug 修复、系统演示与最终文档整理。

由于时间有限：无法加入过于复杂的 AI 模块（如动物图片识别、自动健康状况推断等）；

盈利模块仅开发核心功能（展示、下单、订单记录、收益统计），暂不支持退款、修改订单、第三方支付平台接入等高级功能。

**3.5 操作环境约束**

设备条件为2台windows电脑、Visual Studio Code、SQLite、主流浏览器（Chrome、Edge、Firefox）。

**4. 项目范围**

**4.1 项目范围概述**

《智能化流浪动物救助与领养平台》旨在构建一个涵盖“救助登记—寄养管理—领养审核—公益盈利”全流程的数字化管理平台。项目面向拾主、寄养机构、领养人、管理员以及合作方等多类用户，提供统一的信息入口、高效的审核流程以及可持续运营的盈利机制。

系统范围包括救助流程模块、宠物管理模块、盈利服务模块（疫苗套餐、宠物食粮、宠物保险、平台捐助）、订单管理与收益统计模块以及管理员后台支持。平台最终呈现为基于 Web 的可视化应用，结合轻量级后端服务实现端到端功能闭环。

**4.2 主要任务**

**4.2.1 救助流程模块**

该模块用于接收用户提交的流浪动物相关信息（如照片、位置描述、健康状况等），并将这些信息传递给管理员或相关救助机构进行决策。主要任务包括：救助登记界面设计与实现；流浪动物基础信息收集；救助记录提交接口；管理员审核或分配救助流程；救助状态的跟踪与更新等，此模块作为系统入口，负责建立动物救助生命周期的第一步。

**4.2.2 宠物管理模块**

宠物管理模块用于平台内部管理救助动物，包括其档案信息、寄养记录与健康状况。此模块使管理员或寄养机构能够维护动物信息的完整性，为领养展示与盈利服务提供基础数据。主要内容涉及：动物档案创建、修改与查询；寄养信息管理；健康记录（体检、疫苗、疾病治疗等）；领养展示信息维护等。

**4.2.3 盈利服务模块**

盈利模块为平台提供可持续运营能力，采用公益型商业合作模式，包含以下子服务：

**4.2.3.1 疫苗套餐服务**

疫苗套餐服务：展示合作医院的疫苗套餐并支持下单；

**4.2.3.2 宠物食粮服务**

宠物食粮服务：展示合作品牌宠物粮并支持下单；

**4.2.3.3 宠物保险方案**

宠物保险方案：展示保险内容、金额并允许用户投保；

**4.2.3.4 平台公益捐助**

平台捐助：用户可自主填写捐助金额用于支持流浪动物救助。

盈利模块主要任务包括：前端服务卡片展示；商品/套餐详情展示；支付流程模拟（生成订单与收益数据）；服务类型分类与业务流程一致化处理。此模块与救助场景协同，使平台具备“公益 + 运营”的能力。

**4.2.4 订单管理与收益统计模块**

该模块负责系统收入记录与订单管理，是盈利模块的后台支撑部分。主要任务包括：用户下单后生成订单记录；在收益表中记录对应服务类型的金额；向管理员提供订单明细查询；对收益进行汇总统计（按类型、按日期等）。

**4.2.5 管理员后台支持**

管理员后台模块承担系统全局管理任务，包括：查看与审核救助信息；管理宠物档案与寄养信息；管理盈利服务内容（可扩展）；查看订单与收益统计；管理用户反馈与平台运行状态。

**4.3 交付物**

* + 1. **Web 网页端**

作为主要交互端，需实现：救助流程页面；宠物管理模块界面；盈利服务四类页面（疫苗、食粮、保险、捐助）；订单生成与支付成功展示；收益统计页面；管理员后台界面。界面需具备完整的逻辑与可运行的交互。

**4.3.2 相关文档与系统模型**

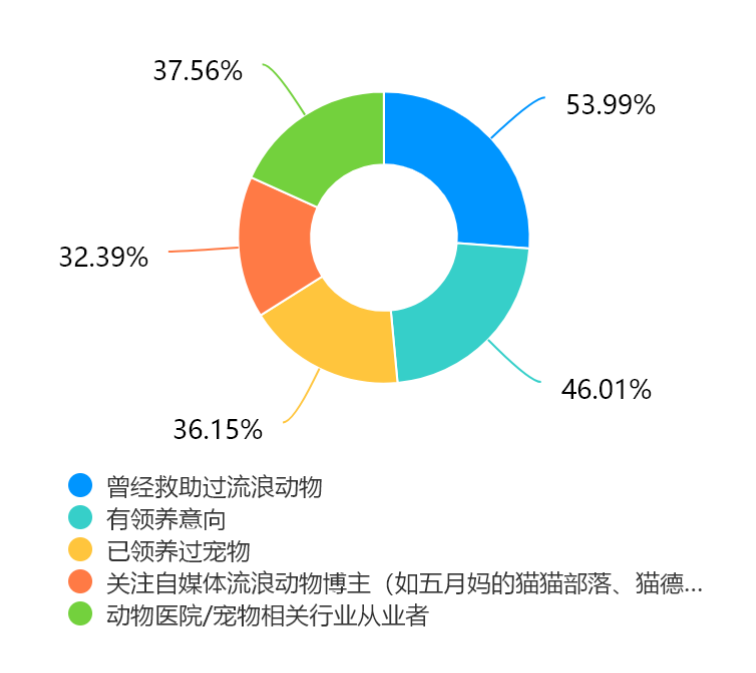
包括：软件需求规格说明书（SRS）；用例图、活动图、顺序图、类图；系统演示文档；项目源代码与部署说明。

**5. 需求调研**

**5.1 用户调研**

**5.1.1 调研目标**

用户调研的主要目的在于识别不同角色用户在救助、寄养、领养及公益捐助过程中的核心需求和痛点。通过调研普通用户、线上流浪动物科普博主（如抖音“五月妈的猫猫部落”“猫德学院林老师”等）、领养意向者等群体，旨在明确：用户希望平台提供哪些救助途径和信息展示方式；用户在领养流程中最为关注的透明度与可信度问题；用户对公益捐助和盈利服务（疫苗、食粮、保险）是否具有明确需求；用户在浏览、提交或查看动物信息时的操作习惯；可视化的健康档案是否有助于提升领养成功率与整体体验感。



（用户身份调研结果）

**5.1.2 调研问题**

本次调研围绕平台可能出现的使用场景设计了若干开放式与封闭式问题，涵盖救助、动物管理、领养及盈利模块使用意愿等方面，主要包括：

1. 您是否曾经遇到需要救助流浪动物却无处反馈的情况？
2. 如果平台提供流浪动物登记服务，您最希望提交哪些信息？
3. 在查看可领养动物时，您最关注哪些内容（如健康状况、性格、寄养环境等）？
4. 您是否愿意通过平台购买宠物疫苗、粮食或保险来支持公益救助行为？
5. 您是否愿意向可信赖的平台进行捐助？
6. 使用平台时，您更倾向于简洁快速的流程，还是深入展示详细信息的页面？
7. 您认为系统需要具备哪些信息透明度保障，让您在领养或捐助时更安心？

**5.1.3 示例问题**

为提升调研质量，设计了以下示例问题供访谈与问卷使用：

“您是否认为我们需要一款包揽全流程的流浪动物领养平台？”

“您是否愿意在平台查看动物的疫苗接种记录、体检情况等具体信息？”

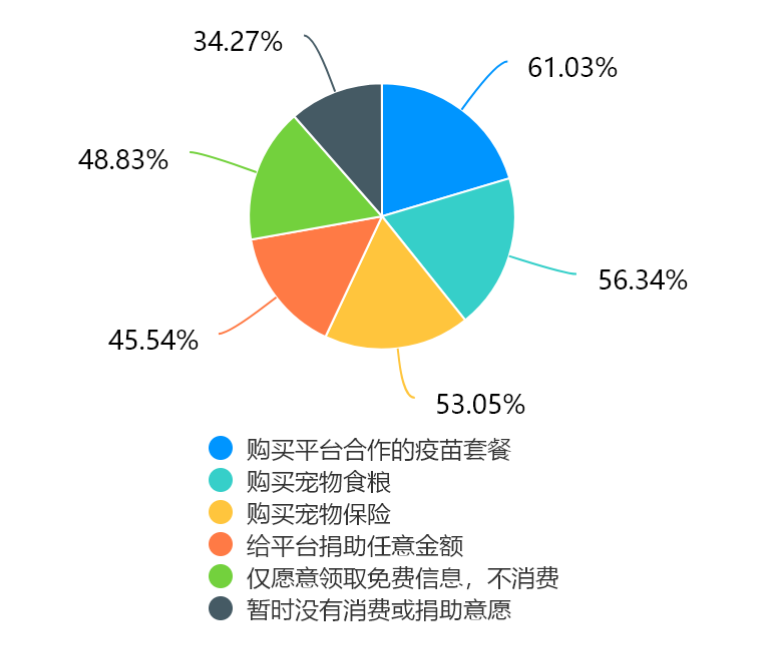
“自媒体救助者常做信息公开，您认为是否应该在平台加入类似的数据透明机制？”

“若平台整合食粮优惠、疫苗套餐与保险，您更偏好哪类公益型服务？”

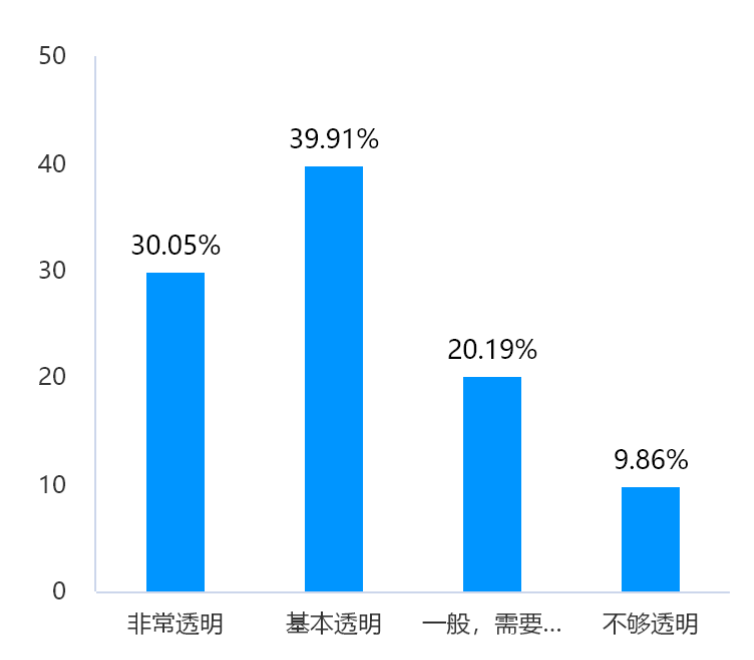
“您对于一款符合您满意程度的领养平台的外观需求是什么样子的？”

**5.1.4 整体结论**

通过用户调研，得出以下总结性结论：用户普遍希望简化救助流程：多位受访者表示，在发现流浪动物时，最缺乏的是快速、统一的救助登记入口，因此平台的救助模块需求十分明确。透明的动物信息极大提升领养信任度：学习“猫德学院林老师”等救助团队的公开模式后，用户普遍希望能够查看动物的健康档案、体检记录、性格描述等，便于做出领养决定。公益型盈利服务获得正向反馈：大部分用户表示愿意通过平台购买疫苗套餐或宠物食粮，以“顺便帮助流浪动物”。优质服务的组合能够提升用户黏性与参与度。捐助意愿与透明度直接相关：若平台能够展示收益记录和使用去向，用户的捐助信任度显著提升。整体来看，用户强烈期待一个透明、高效、可信、公益友好的平台。



（用户对流浪动物资金救助意向度调研结果）



（用户对平台信息透明情况满意度调研结果）

**5.2 相关从业者调研**

**5.2.1 调研目标**

为了使系统需求更加贴近真实救助场景，本次调研选择了：线下流浪狗救助站工作人员；长期从事线上救助科普的自媒体（如“猫德学院林老师”“五月妈的猫猫部落”）；宠物医院工作人员。

调研目标是明确：救助工作中实际的信息记录需求；领养审核与流程规范化的关键要素；宠物医院在疫苗合作与保险合作中的业务流程；救助组织对公益资金透明度与管理工具的需求；盈利模块是否符合真实公益机构的运营逻辑。

**5.2.2 调研问题**

1. 当前救助动物信息记录方式是否存在不便？
2. 救助站是否需要统一管理救助记录、健康档案与领养审核？
3. 您认为一个线上平台能够在哪些方面提升救助效率？
4. 用户捐助、疫苗合作、宠物粮食合作等是否属于实际需求？
5. 您是否愿意使用一个线上系统来记录动物情况？

**5.2.3 数据结果**

调研从业者反馈主要集中在以下几个方面：

救助流程需要标准化：救助站工作人员表示，目前救助信息分散在微信群、私信、电话中，极难统一管理。

动物档案管理非常必要：多数救助人员指出，动物的体检报告、寄养记录、疫苗接种信息是领养成功率的关键因素。

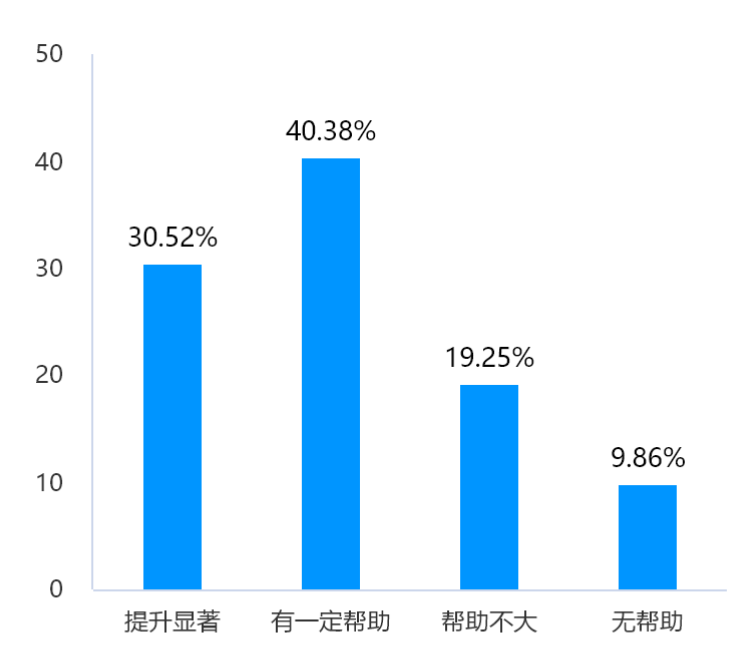
平台透明度至关重要：参考“猫德学院林老师”的模式，救助者更愿意使用能够公开动物档案与救助情况的平台。

盈利服务符合真实需求：宠物医院工作人员确认，疫苗合作套餐是常见合作方式，部分保险公司也愿意与救助组织建立合作。

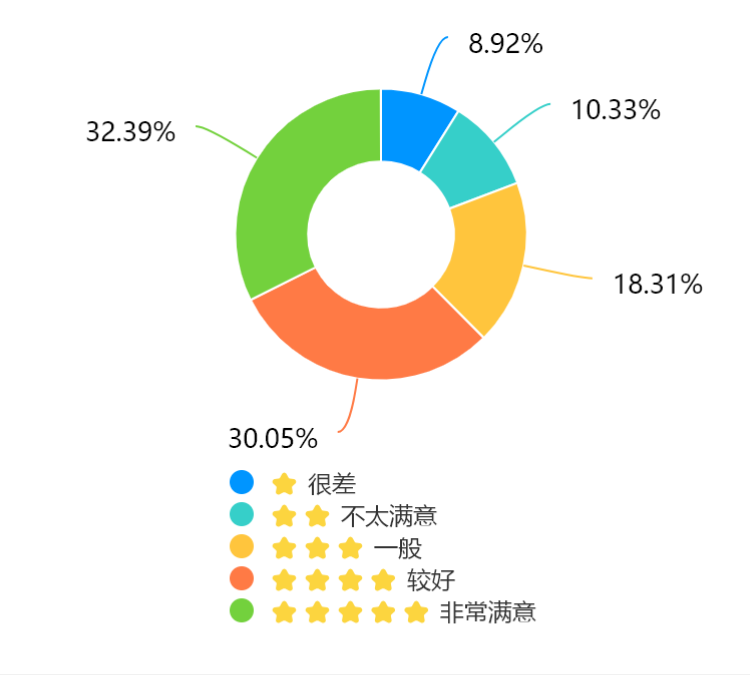
收益统计模块对公益运营至关重要：救助站普遍缺乏数字化记账工具，平台若能统一记录收益收入，将显著提升管理效率。

**5.2.4 整体结论**

相关从业者调研结果表明：救助体系急需数字化工具来规范流程与提高信息可追踪性；透明化的宠物档案与寄养记录能有效提升公众对救助机构的信任；盈利模块不仅是可行的公益商业模式，更能为救助组织持续输送资源；救助组织普遍期待一个“统一登记—信息共享—领养审核—捐助与收益记录”闭环式平台；本项目的功能设计高度契合现实需求，具有实际落地价值。（问卷链接：https://v.wjx.cn/vm/PdS0f3J.aspx）



（相关从业者认为平台是否有助于提升流浪动物救助效率调研结果）



（相关从业者对该平台的整体体验评分）

**5.3调研报告数据总结**

根据对213 份有效问卷的统计分析，本次调研从用户参与动物救助行为、对平台流程理解程度、操作速度满意度、界面感受、公益参与意愿等方面进行了系统评估，所得结果可为系统需求设计提供直接依据。

首先，在用户的现实行为基础上，53.99% 的用户曾经参与过流浪动物救助行为，而 46.01% 的用户从未参与但有兴趣参。这说明平台潜在参与人群基础较大，具有较高的公益行为激发空间。

在对平台流程清晰度的评价中，约 71% 的用户给予“非常清晰”或“基本清晰” 的正向评价，但仍有 29% 的用户认为流程一般或不太清。这提示产品在关键流程（如下单、捐助、保险购买）上仍需做进一步引导和优化，以降低使用门槛。

在操作速度方面，用户的整体满意度较高，“非常满意 + 较满意”占比 70.43%，但仍有约 30% 的用户对响应速度表示一般或不满意。这反映了后端服务性能仍有提升空间，尤其是在高并发或支付流程中。

界面观感方面，用户对平台整体美观度与布局的评价良好，例如：界面美观舒适（63.85%）；布局清晰（59.15%）。但在易用性方面仍存在不足：功能区块过多且层级复杂（47.89%）；信息呈现不够全面（40.38%）。

在信息透明度方面，约 70% 用户认为信息透明或基本透明，但仍有 30% 用户认为一般或不够透明。这表明平台在后续需进一步突出“公益链路透明化”，例如：救助记录更新、资金流向可视化等。

关于用户是否愿意通过平台的盈利服务（如疫苗套餐、食粮合作、保险）支持救助事业：61.03% 愿意购买宠物食粮以支持救助；56.34% 愿意购买疫苗套餐；53.05% 愿意购买宠物保险。

这是一个非常关键的数据，说明用户更倾向于以“消费 + 公益捐助”的模式支持救助体系，这为盈利模块的设计提供了核心依据。

在总体体验方面：62.44% 用户给出 4–5 分（正面评价）；一般评价为 32.39%；不太满意为 8.92%；很差为 3.29%。

结合开放式建，用户主要反馈集中在：捐助流程不够直观；记录透明度需提升；宠物领养/救助链路信息不足；夜间模式、简化按钮等易用性需求；希望平台能提供更多真实救助内容与案例。

综合上述分析，可得以下结论：目前用户对平台的整体认可度较高，尤其在界面美观度、功能完整性和公益价值方面表现优异；但在流程引导、速度表现、透明度、功能易用性方面仍有明确提升空间。尤其值得注意的是，超过 50% 的用户愿意通过购买宠物商品或服务来支持救助事业，这说明 盈利模块在平台生态中具有强需求基础，并可通过透明化公益链路进一步扩大用户参与度。

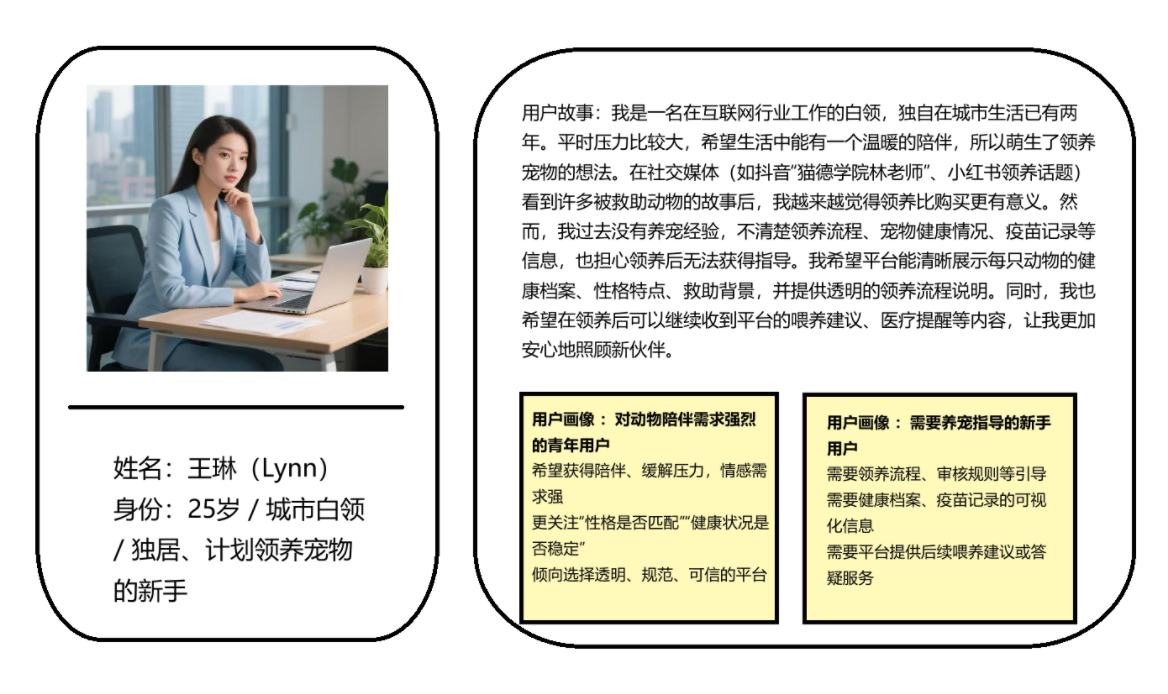
综上，本次调研可为平台在后续版本中进行流程优化、性能强化、公益透明度提升及盈利模块扩展提供可靠的数据支撑。

**5.4 行业财报调研情况**



通过对行业财报与官方白皮书调研发现，宠物行业整体规模持续扩大，宠物食品、疫苗、医疗、保险及公益捐助均呈快速增长趋势。其中宠物食品与医疗支出占比最高，宠物保险增速最快，而公益捐助参与人数最多。该行业结构与本平台盈利模块的四个业务方向高度对应，为系统的可持续运营提供了坚实的数据支撑。

**6.用户故事**

****

****

****

**7.术语表**



**8. 相关事实与假设**

**8.1 相关事实**

1. 流浪动物救助具有较高的不确定性：动物来源多样，健康状况差异大，信息更新频率高，救助流程复杂且依赖人工处理。
2. 用户角色明确且多样化：平台涉及救助者、领养者、捐助者、合作机构以及管理员，不同角色具有不同使用目标。
3. 平台盈利模块为公益辅助性质：疫苗套餐、宠物食粮、宠物保险和捐助服务均带有公益属性，其收益用于平台运营和救助工作。
4. 多数救助信息具有时效性：救助动物的信息（健康、寄养状态、领养情况）需要定期更新，否则会减少其被领养的可能性。
5. 系统基于 Web 技术实现：当前版本采用前后端分离结构，后端基于 Node.js + SQLite3，前端基于 HTML/CSS/JS。
6. 用户对界面友好度和操作简易度有较高要求：多数用户为普通公众，对技术产品理解有限，系统必须保持操作流程清晰。
7. 捐助行为对平台透明度高度敏感：用户是否愿意捐助，往往取决于系统在收益记录公开方面的透明度。
8. 平台需兼容多种设备访问：救助者和领养者大多通过手机浏览救助信息，因此平台需要良好的移动端显示效果。

**8.2 业务规则**

1. 所有救助动物必须具备完整的基础档案：必须至少包括：照片、发现地点、初步健康评估。
2. 领养申请须经管理员审核才能通过：平台不允许未经审核的领养信息直接流转，确保动物领养合法性和安全性。
3. 每个订单必须对应唯一的交易编号：便于收益统计与历史记录追溯。
4. 捐助金额必须真实记录，并不可更改：确保平台公益行为的公正透明。
5. 平台的收益仅用于救助及运营相关目的：合作机构提供服务后所产生的收益需要按规则分配。
6. 合作机构才可发布疫苗、食粮、保险内容：普通用户无权上传商业服务信息。
7. 动物档案状态需保持一致性：当动物被领养后，其状态应统一更新为“已领养”，并从领养列表中移除。
8. 支付成功后必须自动生成订单记录与收益记录：若支付失败，则不得写入数据库。

**8.3 假设**

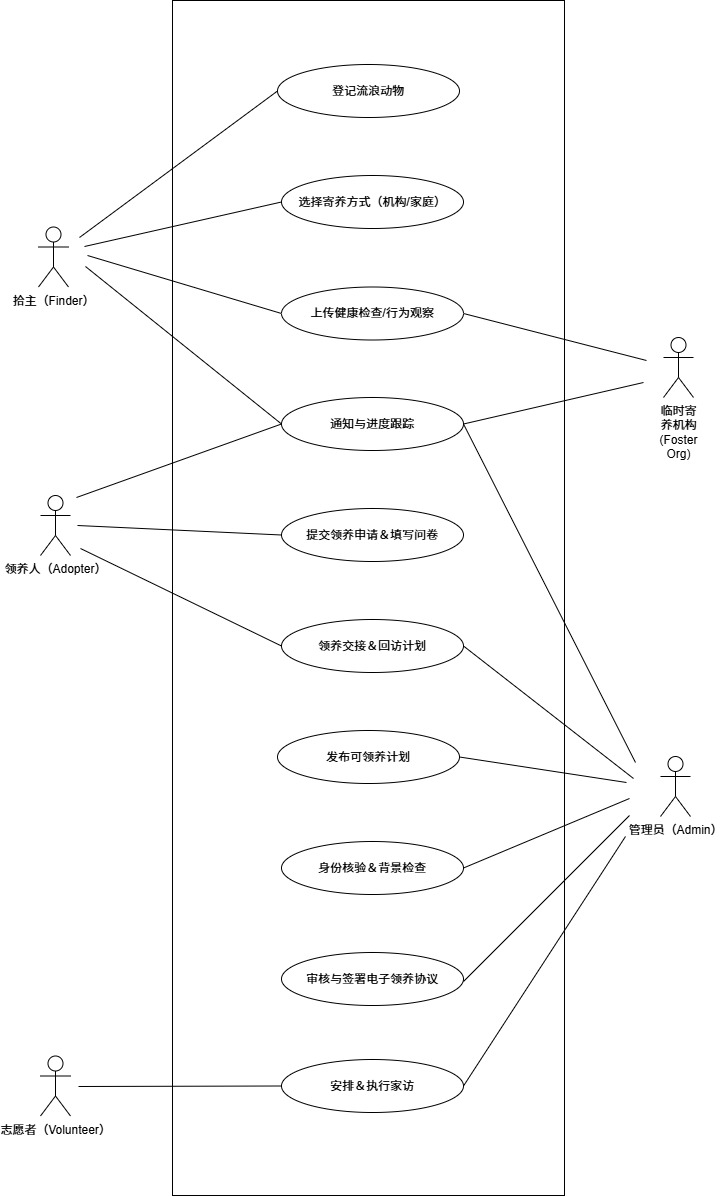
1. 用户愿意通过系统提供准确信息：假设救助者上传的动物状态、照片真实可靠，极端情况下管理员可介入核实。
2. 合作机构具备稳定服务能力：假设合作医院、品牌和保险公司持续提供有效的服务内容，不会突然停止合作。
3. 平台用户具有基础的互联网使用能力：假设用户能理解网页按钮、订单、支付流程等基本概念。
4. 平台访问量在可控范围内：当前使用 SQLite3 数据库，假设初期用户规模较小，不会出现高并发需求。
5. 大部分用户愿意公开部分救助信息：假设救助行为本身具备公益属性，用户愿意公开动物的基本情况以获得帮助。
6. 用户对透明的收益记录有积极反馈：假设透明化能提升用户信任度，并促进捐助行为发生。
7. 移动端访问需求将持续增长：假设未来用户主要通过手机访问平台，因此系统需长期优化移动端体验。
8. 动物救助与领养流程的法律环境稳定：假设不存在政策性重大调整，如救助相关法规变化；如有变化需及时调整系统规则。

**9. 需求建模**

**9.1 基于场景的建模**

**9.1.1 用例图**

**9.1.1.1领养模块用例图**

****

本用例图展示了系统中领养业务流程相关的主要角色和系统交互。整个系统的核心目标是实现从拾主登记流浪动物到领养完成与回访跟踪的全流程闭环管理。

在领养业务中，共涉及五类主要参与者，每一类都承担着独立而关键的角色：

首先是拾主（Finder）。拾主是系统的起点，他们在发现流浪动物后通过平台登记相关信息，包括动物种类、外观特征、健康状况等。同时，拾主需选择寄养方式——可将动物送往临时寄养机构，或由自己进行家庭寄养。拾主还可上传健康检查结果或行为观察记录，为后续的领养审核提供参考依据，并通过系统接收进度通知，了解动物的后续安置情况。

其次是领养人（Adopter）。领养人是系统的主要服务对象，他们通过平台浏览可领养动物的公开信息，填写并提交领养申请与问卷，配合身份核验、家访与协议签署等流程，最终完成领养并参与后续的回访计划。

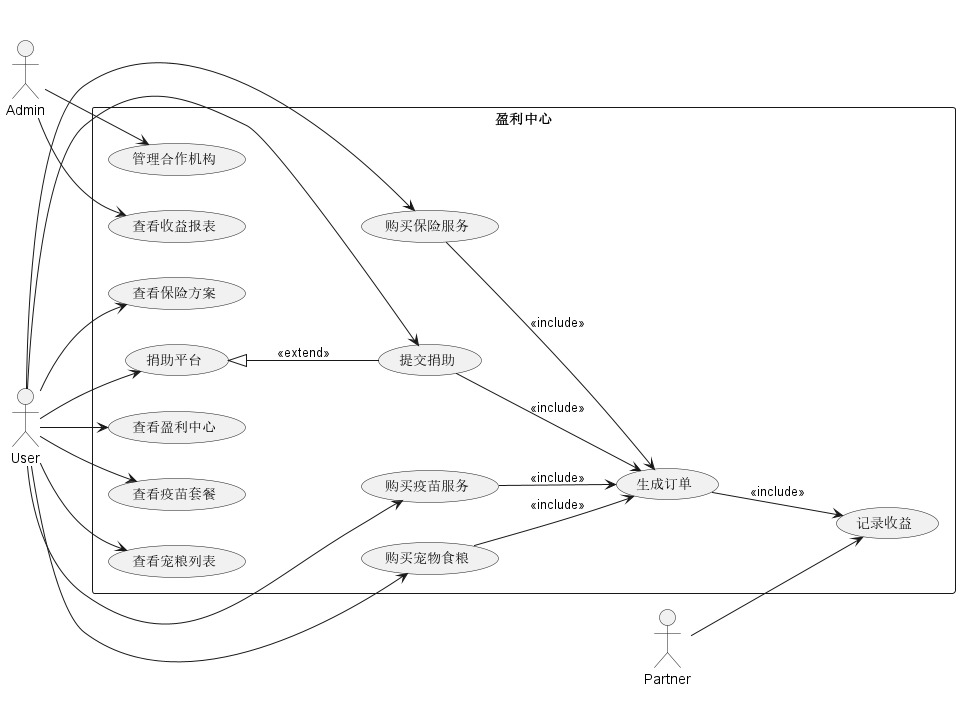
第三类是管理员（Admin）。管理员是系统运作的核心角色，负责动物信息的审核与发布、领养申请的审批、身份验证与背景调查、电子协议的签署管理等工作。同时，管理员也负责调度志愿者执行家访任务，并确保系统中各环节信息的准确与同步。

第四类是临时寄养机构（Foster Org）。该角色为动物提供短期照护与安置服务，负责上传健康检查记录、体检结果以及行为评估信息，并与拾主及管理员保持沟通，以保证动物在等待领养期间得到妥善照顾。

最后是志愿者（Volunteer）。志愿者由管理员安排，承担家访执行与评估记录的任务。通过实地走访或线上交流，志愿者评估领养人的居住环境、照护条件和责任意识，并将结果反馈给管理员，为最终的领养审核提供依据。

系统通过引入多方角色共同协作，使得每一只流浪动物都能被规范地记录、救助、审核与领养。

**9.1.1.2盈利模块用例图**



上图展示了本系统盈利模块的用例结构与各参与者的交互关系。盈利模块主要提供四项面向用户的服务：疫苗合作、宠物食粮合作、保险合作以及平台捐助，同时配合后台的订单与收益管理，实现平台的可持续运营。

主要用例使用者为 普通用户（User），其可执行的核心用例包括：

查看盈利中心；

查看疫苗套餐并购买疫苗服务；

查看宠物食粮列表并购买宠粮；

查看保险方案并购买保险服务；

进入捐助平台并提交捐助。

其中，「购买疫苗服务」、「购买宠物食粮」、「购买保险服务」以及「提交捐助」均通过 «include» 关系复用系统内部通用功能：

生成订单（Generate Order）；

记录收益（Record Revenue）。

此设计保证了不同盈利渠道共享统一的订单处理与收益记录流程，提升模块的可维护性与一致性。

对于捐助功能，“提交捐助”通过 «extend» 扩展“捐助平台”功能，说明用户进入捐助页面后，是否实际捐助是可选行为。

其他参与者包括：

管理员（Admin）：负责查看平台收益报表及管理合作机构。

合作机构（Partner）：可查看与其合作内容相关的用例，例如疫苗套餐、宠粮商品或保险方案。

整个用例模型明确展示了盈利模块的角色、功能结构与内部逻辑，为后续系统设计、数据库设计及前端 UI 设计提供了完整的功能依据。

**9.1.1.2.1购买疫苗套餐用例**



**9.1.1.2.2购买宠物食粮用例**



**9.1.1.2.3购买宠物保险用例**

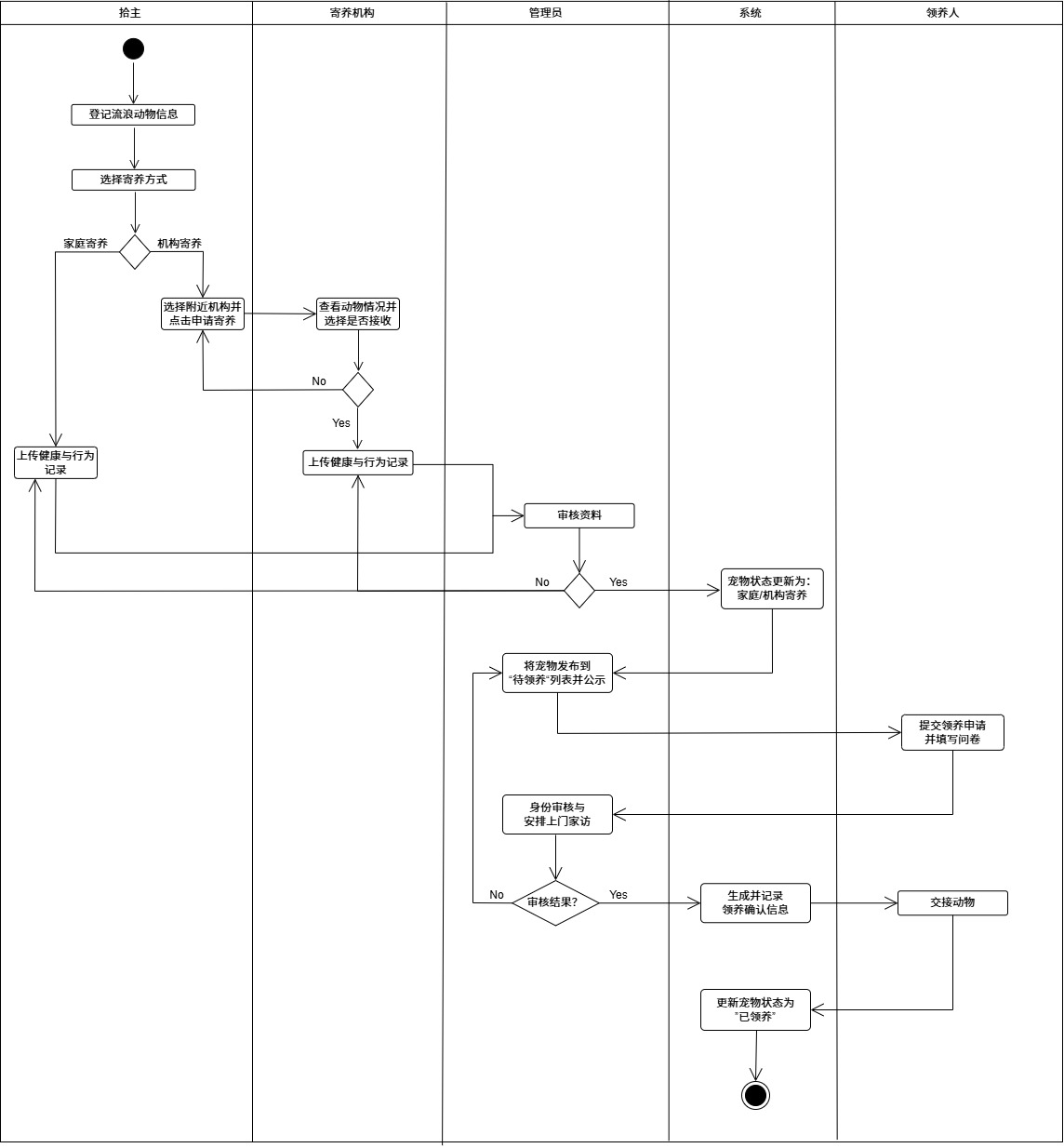


**9.1.1.2.4平台捐助支付用例**



**9.1.2活动图**

**9.1.2.1领养模块活动图**

****

1.拾主登记与寄养方式选择阶段

流程开始于拾主发现流浪动物后，在系统中填写并提交基本信息。拾主需要选择寄养方式：

若选择家庭寄养，拾主可直接在系统中维护动物日常状态；

若选择机构寄养，则拾主需选择合作的临时寄养机构，将动物送往寄养点。

2.寄养记录上传与资料审核

动物寄养后，拾主或寄养机构需上传健康检查报告和行为观察记录。管理员接收资料后进行审核：

若资料不完整或不合格，系统会退回至拾主或寄养机构补充信息；

若审核通过，动物的资料即进入“待领养”阶段。

3.管理员审核与信息公示

当管理员确认资料合规后，系统自动将动物状态更新为“可领养”，并将其信息发布在“待领养动物列表”中，供领养人查看与申请。

4.领养申请与身份核验阶段

领养人浏览待领养动物信息后，可提交领养申请并填写基本问卷。管理员收到申请后启动身份核验与背景调查：

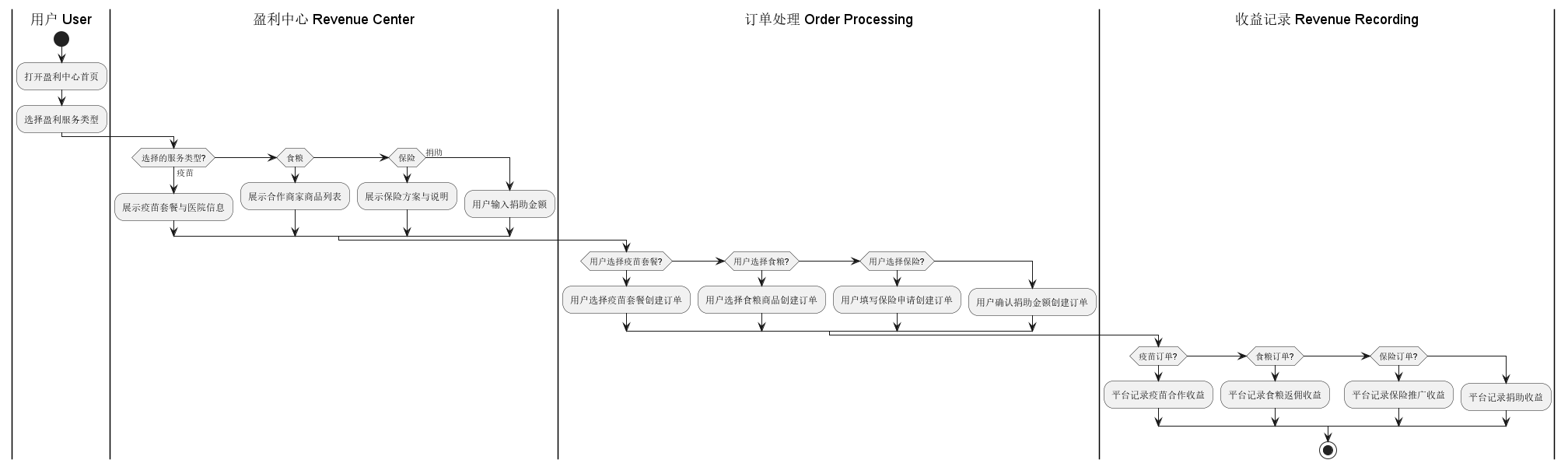
若审核不通过，系统记录审核结果并通知申请人；

若审核通过，则进入交接确认阶段。

5.交接与状态更新

当管理员确认领养通过后，系统生成领养记录并通知双方进行动物交接。交接完成后，管理员在系统中更新动物状态为“已领养”，并归档领养信息。

**9.1.2.2盈利模块活动图**



上图展示了本系统盈利模块的整体业务流程明确区分了 用户（User）、盈利中心（Revenue Center）、订单处理（Order Processing） 与 收益记录（Revenue Recording） 四类参与者在各个环节中的职责。

系统流程从用户访问盈利中心页面开始。用户选择进入系统提供的四类盈利服务，包括：

疫苗合作服务；

宠物食粮合作服务；

保险合作服务；

平台捐助服务。

在盈利中心模块中，系统依据用户选择的服务类型展示相应内容：如疫苗套餐及医院信息、合作宠粮商品列表、保险方案信息，以及捐助金额输入界面。

随后，用户在各自泳道内执行交互操作，例如选择套餐、选择商品、填写保险申请或输入捐助金额。之后流程切换至订单处理（Order Processing）模块，由系统自动生成对应的业务订单，包括：

疫苗预约订单；

食粮商品订单；

保险服务申请订单；

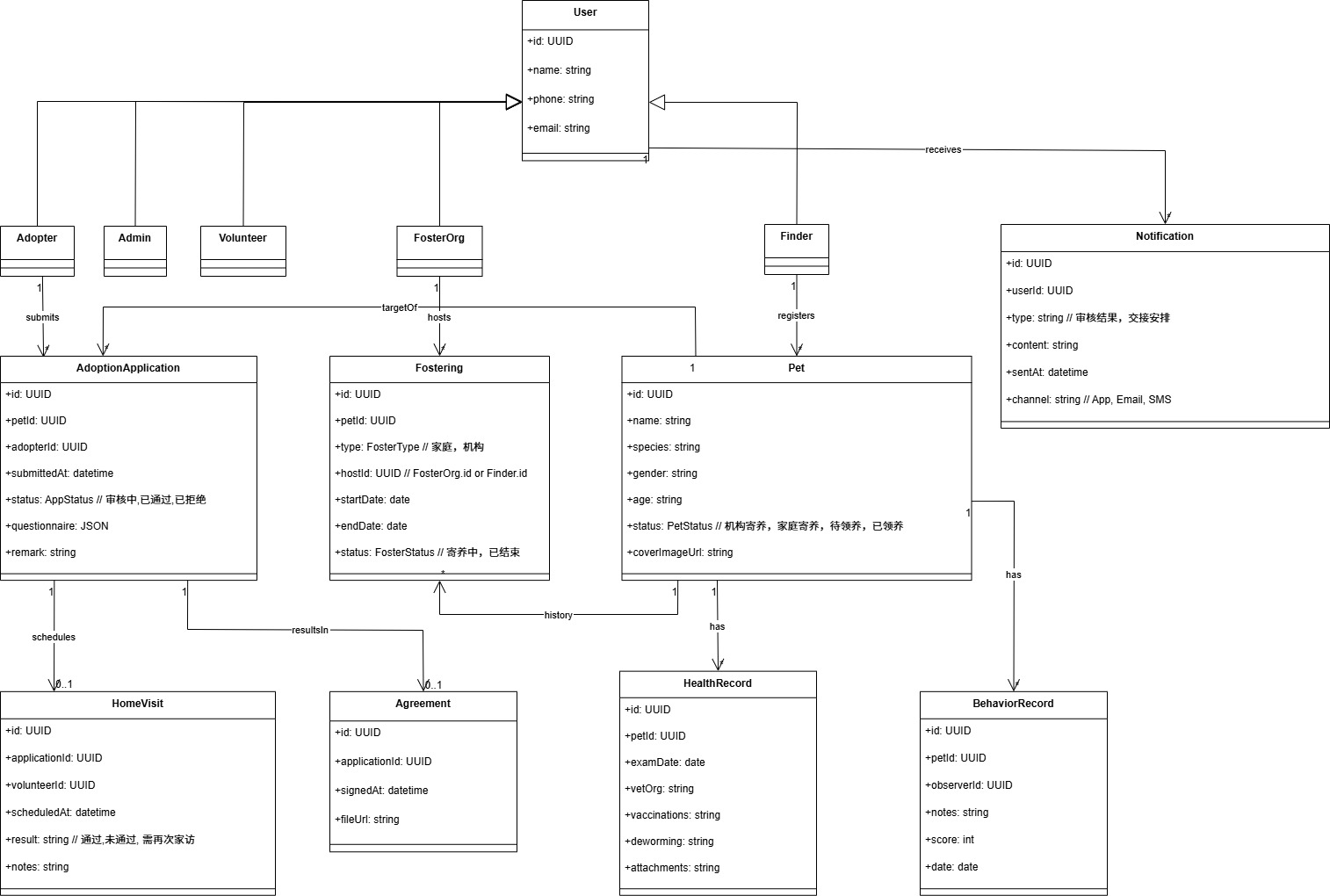
捐助订单。

订单生成后，流程最终进入收益记录（Revenue Recording）模块，系统根据不同类型的服务完成相关收益记录，如疫苗合作收益、宠粮返佣收益、保险推广收益与捐助收益。

每条流程均以独立的流程终止节点结束，体现各盈利分支的封闭性与模块化设计。通过泳道方式展示，能够清晰呈现系统内部各子模块在盈利业务流程中的协作关系。

**9.2 基于类的建模**

**9.2.1 领养模块分析类图**

****

本类图展示了系统中主要实体类及其关系，反映了“救助—寄养—领养”业务全过程的核心数据结构。系统以User为基础抽象类，通过继承区分不同角色（拾主、管理员、志愿者、寄养机构、领养人），并围绕核心对象Pet（流浪动物）建立了一系列关联实体，用于描述寄养记录、健康档案、领养申请、协议签署、家访记录及消息通知等业务逻辑。

用户角色类如下：

1. User（基类）

包含系统中所有用户的公共信息，如 id、name、phone、email。其他具体角色均继承自该类。

2. Adopter、Finder、Admin、Volunteer、FosterOrg（子类）

这五个子类分别表示不同的用户身份：

Finder（拾主）：负责登记流浪动物并选择寄养方式；

FosterOrg（寄养机构）：承担动物短期寄养任务；

Adopter（领养人）：提交领养申请、签署协议；

Volunteer（志愿者）：执行家访任务；

Admin（管理员）：审核、发布、协调系统整体流程。

核心业务类如下：

1. Pet（流浪动物）

记录动物的基础信息，如名称、种类、性别、年龄及状态等。status字段用于标识动物当前阶段：机构寄养、家庭寄养、待领养、已领养。拾主通过Finder登记动物信息。

2. Fostering（寄养记录）

表示一段动物寄养周期。包括寄养类型（家庭或机构）、寄养方、起止日期以及寄养状态（寄养中、已结束）。寄养状态与宠物状态联动，用于追踪动物的暂时安置情况。

3. AdoptionApplication（领养申请）

由领养人发起，记录申请时间、申请状态（待审、已通过、已拒绝）、问卷内容及审核备注。该类是领养流程的核心实体，管理员可根据其状态控制协议生成和宠物状态变化。

4. HomeVisit（家访记录）

对应每个领养申请的家访安排。记录家访时间、执行志愿者、结论（通过、未通过、需再次家访）及备注。管理员根据家访结果决定是否批准申请。

5. Agreement（领养协议）

在申请审核通过后生成，保存协议签署时间及电子文件路径。与AdoptionApplication一对一关联，是领养正式成立的凭证。

6. HealthRecord（健康记录）

存储宠物的体检信息，包括检查日期、兽医机构、疫苗与驱虫情况及附件证明。为管理员评估领养条件提供依据。

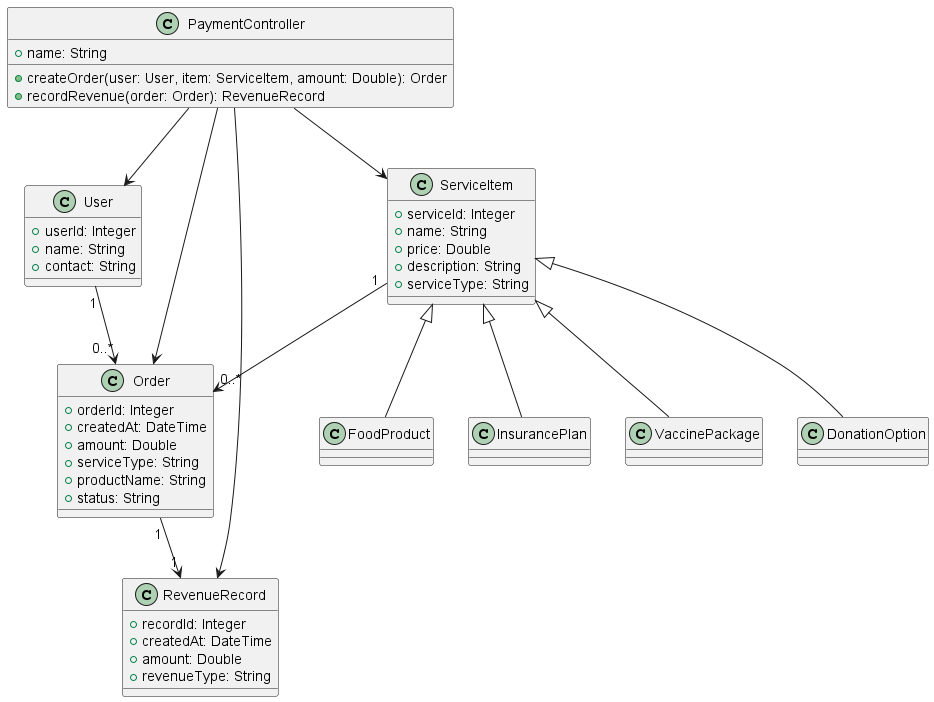
7. BehaviorRecord（行为记录）

记录动物在寄养期间的行为观察、评分和备注。由拾主或寄养方维护，用于后续领养评估。

8. Notification（系统通知）

记录平台发送给用户的消息内容，包括类型（审核结果、交接安排等）、发送时间及渠道（应用内、邮件、短信）。用于实现系统内的信息提醒与进度同步。

**9.2.2 盈利模块分析类图**



如上图，盈利模块类图展示了系统在处理平台盈利业务时所涉及的核心实体及其相互关系，是对盈利服务流程（疫苗套餐、宠物食粮、宠物保险、平台捐助）的结构化抽象。该类图主要包含用户实体、服务项抽象类、服务项具体分类、订单类、收益记录类以及支付控制器类。通过类之间的继承关系、关联关系和组合关系，图示表达了盈利模块在运行过程中各对象的职责划分与交互逻辑。

User 类：User 表示平台使用者，是盈利模块中所有购买行为的发起方。类中包含用户 ID、姓名、联系方式等基本信息。盈利模块中，一个用户可以创建多笔订单，因此 User 与 Order 之间形成 1 对多的关联关系。

ServiceItem 抽象类与四种子类：ServiceItem 是所有盈利服务商品的抽象父类，用于统一描述“可以购买的项目”。该类包含服务 ID、名称、价格、描述及服务类型（疫苗、食粮、保险、捐助）等共有属性。

四种实际业务类型分别继承自 ServiceItem：VaccinePackage —— 对应疫苗合作服务套餐；FoodProduct —— 对应宠物食粮商品；InsurancePlan —— 对应宠物保险服务；DonationOption —— 对应平台捐助金额选项。通过继承结构，将公共属性抽象上移，提高模型可复用性并简化系统扩展。

Order 类：Order 表示一笔实际支付行为所产生的订单记录，是盈利模块的核心交易对象。其属性包括订单编号、下单时间、金额、服务类型、商品名称及订单状态。每一笔订单都由用户创建，并且与某一个具体 ServiceItem 实例对应。

RevenueRecord 类：RevenueRecord 表示平台从订单中生成的收益记录，与 Order 构成一对一的组合关系，即收益记录依赖订单而存在。该类包含收益编号、时间、收益金额以及收益类型。组合关系体现了：“订单删除，则收益记录自动失效”的业务逻辑。

PaymentController 类：PaymentController 是系统中负责处理支付流程的控制器类，对应实际后端 POST /api/payment 的业务逻辑，其主要职责：接收用户购买请求；创建订单（createOrder）；生成收益记录（recordRevenue）。该类与 User、ServiceItem、Order、RevenueRecord 形成协作关系，用于组织盈利操作的业务流程。

通过 ServiceItem 抽象类，将不同盈利类型的共性进行统一抽象，使系统可以通过简单扩展子类来加入新的盈利方式（如宠物托运服务、体检套餐等）。Order 与 RevenueRecord 分离，使交易记录与收益记录两类不同业务数据能够明确区分，提高数据分析的可维护性。PaymentController 作为控制器角色，不存储业务数据，仅负责组织业务流程，符合 MVC 结构分层思想。使用组合关系表示收益记录依赖订单，有助于维持数据一致性。

**10. 接口需求**

**10.1 子系统接口**

**10.1.1 领养流程子系统接口需求**

领养流程子系统负责实现宠物从“登记—寄养—申请—审核—签约”的全过程管理。系统通过一组标准化接口，支持用户提交领养申请、管理员审核、志愿者执行家访、生成电子协议、更新宠物状态，并向相关用户发送通知。本节将从接口功能、特点、安全性与性能要求四个方面进行说明。

**10.1.1.1 接口功能需求**

领养流程子系统的接口遵循 RESTful 规范，通过 HTTP 请求实现数据交互，主要包括申请提交、审核管理、家访记录、协议生成与通知推送等功能。

（1）领养申请类接口

POST /api/adoption/apply

功能：提交领养申请，包含申请人信息、目标宠物 ID 及问卷内容。

输入数据：petId、adopterId、questionnaire（JSON 格式）。

输出：status（成功/失败）、applicationId。

目的：记录用户的领养意愿并进入审核流程。

GET /api/adoption/list

功能：获取用户已提交的领养申请列表。

输出：申请编号、宠物名称、状态（待审、通过、拒绝）、提交时间。

目的：供用户查看申请进度。

（2）审核与家访接口

POST /api/adoption/review

功能：管理员审核领养申请并记录审核意见。

输入：applicationId、decision（approved/rejected）、remark。

输出：审核状态与时间戳。

目的：确定申请结果并触发通知推送。

POST /api/adoption/homevisit

功能：志愿者记录家访结果。

输入：applicationId、volunteerId、result（Pass/Fail/Revisit）、notes。

输出：更新后的家访记录。

目的：为管理员提供审核依据。

（3）协议与通知接口

POST /api/adoption/agreement

功能：生成电子领养协议并保存签署信息。

输入：applicationId、signedAt、fileUrl。

输出：agreementId、状态。

目的：在领养通过后生成正式协议文档。

GET /api/adoption/notifications

功能：返回与领养流程相关的系统通知，如“审核通过”、“家访安排”。

输出：通知类型、内容、发送时间、渠道。

目的：实现用户端消息展示与进度同步。

**10.1.1.2 接口特点**

业务闭环性强：从申请提交、审核、家访到协议生成的全过程均由独立接口实现，可在不同客户端完整复现领养流程。

模块解耦清晰：申请、家访、协议、通知模块相互独立，仅通过ID进行引用关联，降低系统耦合度。

统一数据格式：所有接口返回统一的JSON格式数据，字段命名规范化（如 id、status、createdAt），便于前端统一解析。

状态驱动设计：接口均依托 AdoptionApplication.status与HomeVisit.result状态字段变化驱动后续逻辑，保证流程自动衔接。

易扩展性：未来可添加“领养跟踪回访接口”或“多阶段审核接口”，无需破坏原有数据结构。

**10.1.1.3 安全性需求**

由于领养流程涉及用户身份信息、审核记录与协议文件，必须保证数据传输与存储安全。

1.身份认证与权限控制：

用户提交申请需登录验证；

仅管理员可访问/api/adoption/review接口；

志愿者仅可操作其分配的家访记录。

2.数据完整性与校验：

所有 POST 请求必须验证必填字段完整性；

后端应检查petId与adopterId是否有效且未重复提交申请；

上传文件（协议）需校验格式与存储路径合法性。

3.防伪造与防注入：

采用 Token 鉴权机制防止伪造请求；

所有数据库操作使用参数化语句，防止SQL注入。

4.敏感信息保护：

屏蔽领养人联系方式等隐私字段的对外暴露；

协议文件需存储于安全路径，访问需授权。

**10.1.1.4 性能需求**

为保证领养流程流畅性，接口需满足以下性能要求：

1.响应时间：

GET 请求（列表类）平均响应时间应 < 200ms；

POST 请求（申请/审核）平均响应时间应 < 300ms；

协议文件上传接口应在 1s 内完成存储并返回链接。

2.并发能力：

支持 50 次/秒的领养申请提交请求；

支持 20 次/秒的家访结果记录与状态更新操作。

3.数据库优化：

对 AdoptionApplication.submittedAt 与 status 建立索引；

对 Notification.userId 添加外键索引，提高用户消息查询速度。

4.可扩展性要求：

支持后续多租户部署（不同救助站独立数据空间）；

当申请记录超过 10 万条后，需升级数据库至 MySQL/PostgreSQL 并使用分页查询机制。

**10.1.2 盈利模块接口需求**

盈利模块主要提供疫苗套餐购买接口、宠物食粮购买接口、宠物保险购买接口和平台捐助接口；同时还包括统一的支付接口与订单/收益查询接口。本节将从接口功能、特点、安全性与性能要求四个方面进行说明。

**10.1.2.1 接口功能需求**

盈利模块的接口由前端页面通过 HTTP 请求访问后端服务，实现商品展示、支付流程、订单生成及收益记录等功能。主要接口需求如下：

获取服务商品类接口：

GET /api/vaccine

功能：返回疫苗合作套餐列表，包括名称、套餐说明、价格信息。

目的：前端展示疫苗套餐选择页面。

GET /api/food

功能：返回宠物食粮商品列表，包括商品名称、规格与价格。

目的：支持用户在宠物食粮模块浏览并选择商品。

GET /api/insurance

功能：返回保险方案列表，包括保障范围、方案名称、价格。

目的：用于保险服务页面展示与对比方案。

支付与订单类接口：

POST /api/payment

功能：统一支付接口，创建订单并记录收益。

输入数据包括：

service\_type（业务类型）

product\_name（商品名称）

amount（支付金额）

输出：

status（成功 / 失败）

order\_id（创建的订单 ID）

目的：完成订单创建与收益记录，是盈利模块核心接口。

订单/收益查询接口：

GET /api/orders

功能：返回所有订单及对应收益记录。

包含字段：订单编号、商品名称、金额、创建时间及收益金额。

目的：用于管理员或收入统计页面展示。

**10.1.2.2 接口特点**

统一性：盈利模块采用统一支付接口（POST /api/payment），无论用户购买疫苗、保险、食粮或捐助，均通过相同入口处理，保证业务逻辑一致，降低后端维护成本。

独立性：获取商品信息的接口（/api/vaccine、/api/food、/api/insurance）具备完全独立性，不会依赖登录机制或其他业务逻辑，有助于前端自由布局页面结构。

高扩展性：服务商品列表可随时拓展，接口返回格式采用同构数据结构（id、name、amount），未来可轻松增加新盈利方式，如“宠物托运服务”或“宠物体检套餐”。

轻量化设计：所有接口均基于 RESTful 模式，使用 JSON 数据传输，便于在多端（Web、移动端）集成，网络开销小、开发效率高。

强耦合性控制良好：订单与收益记录明确分离，通过 join 查询返回组合数据，结构清晰，可灵活用于报表统计。

**10.1.2.3 接口安全性需求**

盈利模块涉及用户支付金额、订单流水与平台收益等敏感数据，因此必须满足以下安全性要求：

数据合法性校验：后端必须校验 service\_type 是否为预定义类型，校验 amount 是否为合法金额且与商品价格一致，避免前端篡改金额造成财务漏洞。

防止伪造请求：所有 POST 请求需进行请求体合法性检查，禁止未带有效 JSON 数据的非法访问，建议未来接入 Token / Session 鉴权防止接口被恶意调用。

数据库安全：使用参数化语句防止 SQL 注入，对价格、订单号等关键字段进行格式约束。

错误保护机制：payment 接口在写入失败时必须回滚，查询接口不得返回多余系统信息（如服务器路径）。

未来支付安全扩展：若接入真实支付方式（微信/支付宝），必须遵守：HTTPS 加密传输，签名机制，回调验签等安全规范。

**10.1.2.4 接口性能需求**

盈利模块的接口需在保证稳定性的前提下，满足基本性能要求：

响应时间要求：商品类接口（GET）在正常情况下响应时间应 < 200ms，支付创建订单接口（POST）在数据库写入正常情况下应 < 300ms，查询订单接口应在 1 秒内返回结果（随着数据增长需优化索引）。

并发能力需求：允许至少 100 次/秒的查询访问量（商品浏览），支付接口需能承受至少 20 次/秒并发写入（适用于活动高峰）。

数据库性能优化需求：对 orders.created\_at 添加索引以提高统计效率，对 revenue\_records.order\_id 添加唯一索引确保快速关联查询。

前端性能要求：列表接口返回 JSON 不应超过 200KB，建议分页查询订单数据，避免前端卡顿。

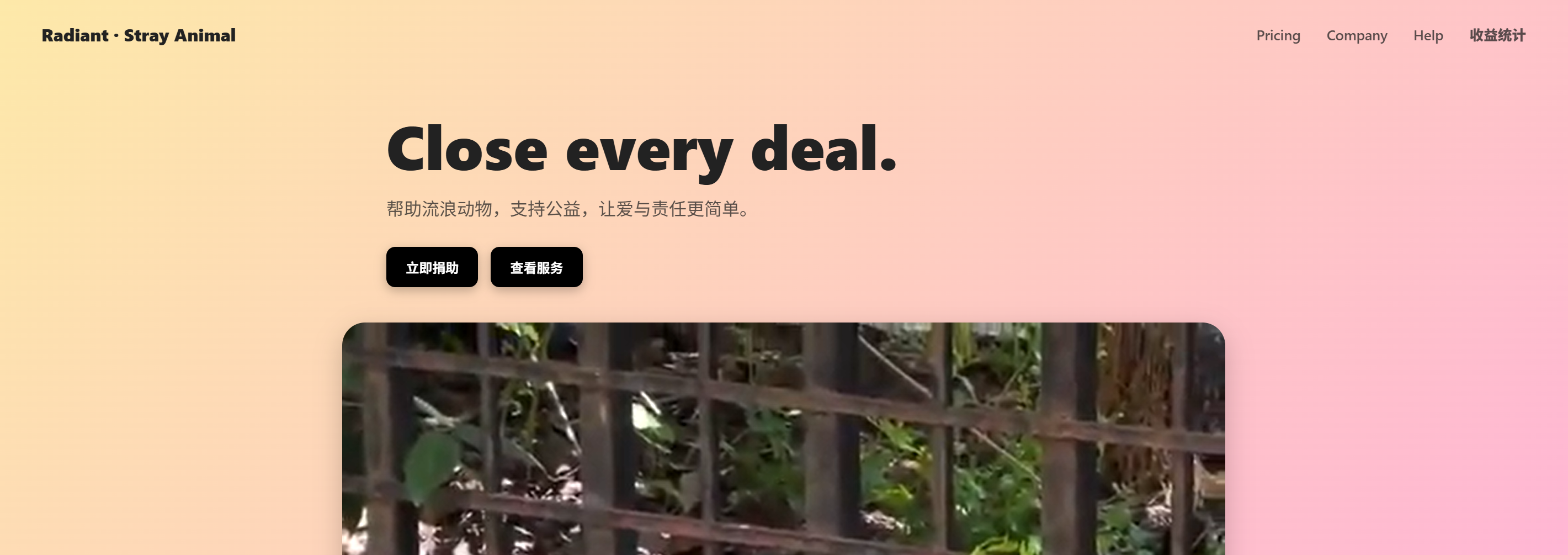
可扩展性性能要求：当订单数量超过 10 万笔后，建议升级 SQLite → MySQL/PostgreSQL，支持更高查询效率与数据分析能力。

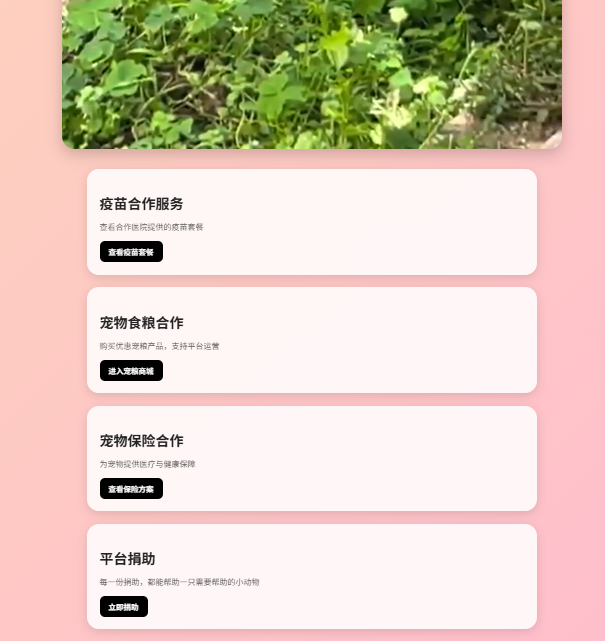
**10.2 接口与原型图**

**10.2.1 领养管理模块原型**

**10.2.2 盈利模块原型**

**10.2.2.1盈利模块主界面模块原型图低保真**

****

****

(盈利系统主界面部分预览图)

**10.2.2.1盈利模块疫苗服务模块原型图低保真**

****

(盈利系统疫苗服务预览图)

**10.2.2.1盈利模块食粮服务模块原型图低保真**

****

(盈利系统食粮服务部分预览图)

**10.2.2.1盈利模块保险服务模块原型图低保真**

****

(盈利系统保险服务预览图)

**10.2.2.1盈利模块捐助服务模块原型图低保真**

****

(盈利系统捐助服务预览图)

**10.2.2.1盈利模块后台订单管理模块原型图低保真**

****

(盈利系统后台订单管理模块预览图)

**11. 非功能性需求与解决方案**

**11.1 性能需求设计与解决方案**

**11.1.1 性能优化设计**

流浪动物救助平台属于多角色、多场景的 Web 系统，需要保证在多用户访问、动物信息更新、服务购买与订单生成等操作下仍保持稳定和流畅的体验。性能需求主要集中在以下几个方面：页面加载速度关键界面（首页、动物列表、收益统计页）需在合理网络环境下保持 1–3 秒以内的加载速度，避免用户流失；数据库查询性能随着动物档案、订单数量增长，数据库需支持快速查询与排序，尤其是收益统计页面的聚合查询，应保持响应时间在 1 秒以内；前端渲染需减少 DOM 重绘次数，保证滚动浏览动物列表、查看图像、展示视频时不卡顿；虽然平台早期并发需求较低，但在出现热点事件（如博主曝光某紧急救助案例）时可能出现瞬时访问高峰，平台需具备一定的应对能力；优化系统中包含的大量动物照片与视频静态资源，应对其进行压缩与缓存处理，降低带宽消耗。

**11.1.2 性能需求实现方案**

**11.1.2.1前端静态资源优化**

使用浏览器缓存与 CDN 加速常用资源（CSS、JS、封面图）；

将图片资源通过 WebP 压缩以减少体积，提升加载速度；使用懒加载（Lazy Load）技术，避免用户一次性加载大量动物图片。

**11.1.2.2数据库性能优化**

在 SQLite 向更大规模迁移前，添加必要索引（如动物编号、订单时间）；后期可替换为 MySQL/PostgreSQL 以支持更高并发与复杂聚合查询；使用分页查询减少一次加载数据量。

**11.1.2.3后端服务优化**

利用 Node.js 的异步事件机制减少阻塞操作；对订单与收益统计采用批处理或预计算方式，减少重复计算。

**11.1.2.4接口性能提升措施**

对常用数据接口启用服务器缓存，提高响应速度；使用 gzip/br 压缩所有接口返回内容。

**11.1.2.5视频播放优化**

采用流媒体播放方式，避免一次性加载整个视频；视频转码为 H.264/H.265 格式，兼顾质量与加载效率。

**11.2 安全性需求设计与解决方案**

**11.2.1 安全性设计**

平台包含救助信息、订单信息与收益数据，涉及公益资金，因此安全性必须保证。

安全性需求包括账号权限安全：管理员拥有更高权限，应严格区分管理端与用户端功能；支付流程安全：虽为模拟支付流程，但须保证订单参数不可被篡改；数据输入安全：防止用户提交恶意脚本导致 XSS 或数据库注入攻击；数据隐私保护：用户手机号（如未来加入）、领养申请信息等敏感数据需保护；服务器访问安全：后端接口需要限制跨域访问与非法调用。

**11.2.2 实现方案**

权限控制：管理员后台与普通用户界面完全隔离、后端进行角色校验，避免越权访问；输入验证与过滤：所有输入字段进行类型检查与长度限制、防止 SQL 注入（使用 SQLite3 预编译语句）；加密与脱敏处理：对用户联系方式、领养申请内容进行加密存储或部分脱敏显示；日志与行为监控：对管理员关键操作进行日志记录，出现数据错误可溯源；接口安全策略：使用 Token 验证避免接口被频繁恶意访问、限制部分 API（如收益统计）仅在登录状态下可访问。

**11.3 可扩展性需求设计与解决方案**

可扩展性是本平台的重要非功能需求，由于救助平台未来可能快速增长，并引入多种功能与合作模式，因此系统架构需具备良好的模块化设计与功能扩展能力。

**11.3.1 可扩展性设计**

**11.3.1.1模块化结构设计**

平台各功能（救助模块、领养模块、盈利服务、订单系统、后台管理）均采用独立模块设计，未来可单独替换、扩展或升级。

**11.3.1.2插件式功能扩展能力**

为未来加入新服务（如宠物托运、寻宠服务、宠物论坛功能）预留接口结构。

**11.3.1.3数据库垂直扩展能力**

随着平台数据量增加，可平滑迁移至更强大的数据库引擎（如 MySQL）。

**11.3.1.4内容投放与大模型接口对接能力**  
平台需支持接入 LLM（例如 GPT 类模型）用于内容审核、推荐、论坛引导等。

**11.3.2 实现方案**

实现宠物论坛功能并引入大模型精准内容投送,该功能旨在提升用户活跃度，形成宠物社区文化，论坛内容模块化、用户行为数据采集、大模型用于内容推荐、模型用于内容审核、模型用于生成引导文案。

加入“宠物走失寻宠模式”功能，此功能旨在帮助用户快速寻找走失宠物，提高平台公益价值，可以实现走失登记、相似度匹配、区域推送、管理员介入流程。

加入“宠物托运功能联动模式”面向跨城市领养用户或寄养需求，可以实现托运服务选择、订单联动，形成闭环服务、费用展示、安全提醒。

系统组件可替换设计，前端可从原生 JS 升级至 Vue / React、支付模块可从模拟支付扩展至真实支付接口、图像模型可从简单比对扩展至深度学习模型识别。

**11.4 可靠性与持久性需求设计与解决方案**

**11.4.1 可靠性与持久性设计**

订单与收益记录必须保持一一对应关系，不能出现孤立记录，后端需能处理无效输入、网络中断等异常情况，所有救助记录、动物档案、订单等数据需长期保存，定期备份数据库以防止误删或损坏。

**11.4.2 实现方案**

在订单生成与收益写入过程中采用事务处理，确保数据一致性，后端增加错误捕获逻辑，在接口失败时返回明确错误信息，定期备份 SQLite 文件，未来可使用自动备份工具连接云数据库，在备份损坏时可恢复最近版本，确保平台继续运行。

**11.5 兼容性需求设计与解决方案**

**11.5.1 兼容性设计**

平台用户来源广泛，需要确保系统在不同设备与不同操作系统上均可正常运行，PC、平板、手机均可正常访问，Chrome、Edge、Safari、Firefox 适配，根据不同屏幕宽度自动调整布局，在较差网络环境下仍能加载核心内容。

* + 1. **实现方案**

使用响应式 CSS 框架（如 Flex、Grid）自动适配不同屏幕尺寸，对常见浏览器进行 UI 和接口测试，避免样式错乱，缩减资源体积和静态文件缓存，在弱网环境下优先加载关键信息（动物标题、照片缩略图），按钮间距增大，字体适配不同分辨率，视频自动适配比例播放

* 1. **平台计算资源使用设计**

**11.6.1 服务器资源使用设计**

本平台在早期建设阶段采用轻量级计算资源策略，以保证系统在低成本前提下具备稳定运行能力。由于平台主要由动物档案展示、服务购买、订单生成与收益统计等功能构成，对实时高性能计算需求并不强烈，因此服务器资源设计以稳定性、可扩展性与资源合理分配为核心。当前系统后端基于 Node.js 运行于轻量服务器环境，数据库使用 SQLite3，以保证在用户规模较小时即可提供快速响应和低延迟的数据访问体验。同时，平台对静态资源（如动物图片、视频介绍、服务封面图）的加载采用前端缓存和延迟加载策略，以减少服务器压力，提升用户端渲染速度。随着用户数量的增长与功能扩展（如宠物论坛、大模型内容推荐、走失寻宠模式）的引入，平台需逐步增加计算资源，包括升级数据库至 MySQL/PostgreSQL、提升 CPU 性能以支撑模型推理调用、扩展磁盘资源用于存储大量动物档案与用户论坛内容，并可能引入分布式资源管理方案以支持高峰并发需求。

**11.6.2 实现方案**

在具体实现层面，平台将采用渐进式资源扩展策略。初始部署阶段采用单机服务器架构，运行 Node.js 后端服务并存储本地 SQLite 数据库，以满足低访问量情况下的性能需求。随着论坛模块上线和媒体资源增多，平台将迁移至云服务器环境，并启用对象存储服务用于保存图片与视频，以降低服务器带宽消耗。大模型推理任务不在本地运行，而采用调用外部 API（如 OpenAI、阿里通义、百度文心等方式），从而避免在本地部署 GPU 或高算力设备所带来的高成本压力。后期若用户规模增长，可将后端拆分为多个服务模块，通过负载均衡方式分发请求，并利用云数据库实现读写分离与自动备份，以保障系统长期稳定运行。整体方案确保平台根据业务规模动态扩容，在控制成本的前提下逐步提升计算能力与服务质量。

**12. 其他问题**

**12.1 开放问题**

**12.1.1 技术层面**

在系统早期采用轻量级架构（Node.js + SQLite3）的情况下，数据库性能在用户规模扩大后可能出现瓶颈，尤其是在动物档案数量增长、订单交易频率提升、论坛模块引入后，查询性能与数据同步方式需要进一步优化。此外，未来集成大模型用于内容推荐或审核时，将带来较高接口调用频率，对响应速度、外部 API 的稳定性、以及成本控制提出新的要求。同时，平台若引入地理位置相关功能（如寻宠模式推送），需处理定位精确度、坐标转换以及地理数据库查询的问题，这部分在当前版本尚未完全实现。

**12.1.2 用户体验层面**

尽管平台已具备核心功能，但用户界面在多设备上的一致性、交互提示的清晰度、页面层级深度等方面仍有进一步提升空间。部分用户可能会在首次使用时对救助流程或领养流程的理解不足，需要设计更多可视化引导或动态说明。此外，随着论坛模块、寻宠模式等新功能加入，界面复杂度可能提高，需要权衡功能丰富性与操作简易度之间的平衡。用户对数据透明度（如捐助流向、救助状态）有较高期望，该部分也需持续优化呈现方式。

**12.1.3 系统扩展与可维护性**

系统未来将引入更多合作机构、服务品类、数据模块，现有架构在扩展性方面仍需强化。例如目前并无完善的模块化插件体系，在功能增多后可能导致代码耦合度上升。同时，后台管理模块的权限管理、内容审核流程等机制需要进一步完善。如果未来加入实时通信（如寻宠紧急通知、论坛回复提醒），则需要新的 WebSocket 通信架构。此外，平台缺乏成熟的自动化测试体系，随着功能增多，缺乏测试可能导致更新风险上升。

**12.2 解决方案**

针对上述问题，平台计划逐步实施结构化优化策略。在技术上，系统将从 SQLite3 迁移至 MySQL 或 PostgreSQL，并利用数据库索引优化、读写分离或缓存机制提升性能；同时采用云对象存储处理大规模媒体资源，减轻服务器压力。在内容审核与推荐方面，大模型 API 的调用将启用缓存与分级策略，例如将静态信息的分析结果长期缓存、对高频内容启用低成本模型以减少开销。对于用户体验问题，平台将进行多轮 UI 优化与真实用户测试，简化关键操作，并加入可视化引导机制提升信息传达效率。针对维护与扩展问题，将逐步实施模块化重构，构建可插拔式业务模块，使论坛、寻宠模式、托运服务等均可独立部署和维护。同时引入版本管理、自动备份机制和自动化测试体系，确保系统更新的可靠性。对于实时通信需求，可在后期采用 Socket.IO 或消息队列解决方案，保证系统在功能扩展后的稳定运行。

* 1. **待办事项**

1.完成数据库升级至 MySQL/PostgreSQL 的可行性分析与迁移方案设计。

2.启动 UI 第二轮迭代设计，包括流程简化与多设备适配优化。

3.设计并实现宠物论坛原型界面，明确帖子结构、推荐逻辑及审核机制。

4.开发“宠物走失寻宠模式”的基础功能模块，包括走失登记、位置上传与简易推送。

5.完成合作机构后台权限系统的优化与内容审核流程制定。

6.引入大模型内容服务（审核/推荐）的 API 封装层，确保可控成本与稳定性。

7.设计并部署自动化测试脚本，为订单、捐助、收益统计等关键功能提供持续验证。

8.规划实时通信模块（如寻宠推送、论坛通知）的技术选型和架构设计。

9.引入数据库自动备份机制，完善系统的容灾能力。

10.编写开发文档与维护指南，为后续团队扩展提供支持。

**13. 参考内容**

本章节列出本项目在需求分析、功能设计、流程规划与盈利模式研究过程中参考的资料，包括国内外动物救助组织、公益平台、宠物行业服务机构、自媒体救助者等公开信息。

1.猫德学院（北京猫德动物救助中心）  
官方微博： https://weibo.com/catdeacademy

2.领养日（AdoptDay）公益组织  
官网： https://www.lingyangri.org/

3．汪汪队流浪狗救助基地（广州）  
微信公众号：汪汪队流浪狗救助基地

4.中华动物保护协会（China Animal Protection Association）  
官网：http://www.cap.org.cn/

5.美团宠物疫苗服务  
美团 App 内搜索“宠物疫苗”

6.京东宠物食粮类目（真实电商结构）  
官网：https://www.jd.com/pet  
7.平安宠物保险  
官网：https://baoxian.pingan.com/pet-insurance

8.抖音账号：五月妈的猫猫部落

9.抖音账号：猫德学院林老师和“公益信任机制”提供实际参考。

10.B站账号：流浪猫小院

11.微信公众平台：寻宠宝案例文章  
公众号搜索：寻宠宝