**概要设计说明书**

**《**翼界探秘系统**》**

**编写日期：**2024年6月18日

**项目组：**

|  |  |  |
| --- | --- | --- |
| **学号** | **姓名** | **角色** |
| **202231060926** | **魏辰睿** | **组长** |
| **202231060923** | **胡承波** | **组员** |
| **202231060920** | **李浩楠** | **组员** |
| **202231060924** | **董成阳** | **组员** |
| **202231060921** | **王皓宇** | **组员** |
| **202231045205** | **董美萱** | **组员** |

**修改日志**

|  |  |  |
| --- | --- | --- |
| 修改者 | 修改日期 | 备注说明 |
| 魏辰睿 | 2024.6.11 |  |
| 胡承波 | 2024.6.15 |  |
| 董成阳 | 2024.6.18 |  |
| 董美萱 | 2024.6.19 |  |

**目 录**

[1. 引言 5](#_Toc170500702)

[1.1 编写目的 5](#_Toc170500703)

[1.2 定义 5](#_Toc170500704)

[1.3 参考资料 6](#_Toc170500705)

[2. 范围 6](#_Toc170500706)

[2.1 系统主要目标 6](#_Toc170500707)

[2.2主要软件需求 7](#_Toc170500708)

[2.3设计约束、限制 8](#_Toc170500709)

[3. 软件系统结构设计 8](#_Toc170500710)

[3.1 软件体系结构 8](#_Toc170500711)

[3.1.1 软件程序结构图 8](#_Toc170500712)

[3.1.2 模块描述 9](#_Toc170500713)

[3.1.3 模块描述表 9](#_Toc170500714)

[3.2 功能需求追溯 10](#_Toc170500715)

[4. 数据设计 11](#_Toc170500716)

[5. 接口设计 12](#_Toc170500717)

[5.1 用户界面设计规则 12](#_Toc170500718)

[5.2 内部接口设计 12](#_Toc170500719)

[5.3 外部接口设计 12](#_Toc170500720)

[6. 出错处理设计 13](#_Toc170500721)

# 引言

## 编写目的

* **明确项目需求**： 详细阐述鸟类识别App的需求背景、目标用户群体、核心功能需求及非功能需求，确保项目团队对项目的需求有清晰的认识。
* **规划项目架构**：设计合理的系统架构，包括前端用户界面、后端服务逻辑、数据库设计以及第三方服务集成等，确保系统的稳定性、可扩展性和可维护性。
* **制定技术实现路径**：根据系统架构和功能模块划分，确定具体的技术实现方案，包括开发语言、框架、工具及第三方库的选择等，确保项目的技术可行性和高效性。
* **确立设计标准**：为系统设计提供一套标准化的指导原则和规范，确保设计的一致性和可维护性。
* **指导开发过程**：作为开发团队进行详细设计、编码和测试工作的依据，确保开发活动与设计目标相符。
* **促进团队沟通**：作为项目组成员之间沟通的桥梁，帮助团队成员理解自己的职责和工作重点。
* **管理项目预期**：明确项目的功能需求和性能指标，为项目管理者监控项目进度和质量提供参考。
* **风险评估**：通过概要设计，评估项目可能面临的技术风险和实现难度，及时调整项目计划以降低风险。
* **利益相关者沟通**：向项目利益相关者（如投资者、用户代表等）展示项目设计概貌，获取他们的理解和支持。
* **支持后续文档编写**：为编写其他项目文档（如用户手册、操作手册、测试计划等）提供基础信息和数据。
* **预期读者：**项目开发人员：包括前端开发、后端开发、数据库管理、UI设计等相关人员。团队成员通过本说明书了解项目的整体架构、功能模块划分、技术实现路径及开发规范等，以便更好地进行后续的开发工作。项目经理：通过本说明书了解项目的整体设计思路、开发计划、里程碑节点及关键风险点等，以便更好地进行项目管理和进度控制。利益相关者：包括投资方、合作伙伴等。他们可以通过本说明书了解项目的市场前景、商业价值及潜在风险点等，以便做出更明智的决策。

## 定义

**深度学习（Deep Learning）**：深度学习是机器学习的一个子领域，它使用深度神经网络（例如卷积神经网络、循环神经网络、全连接神经网络等）来模拟人脑处理信息的方式，通过多层次的学习过程自动提取数据的特征，并进行分类、预测、生成等任务。

**日志（Log）**：日志是计算机系统中用于记录系统运行信息、用户操作、错误事件等的文件或数据集合。它们通常用于故障排除、性能监控、安全审计等目的。

**数据记录（Data Record）**：数据记录是存储在数据库或文件系统中的一组相关数据项的集合，通常表示一个实体（如用户、订单、产品等）的完整信息。数据记录由多个字段（Field）组成，每个字段包含该实体的一个属性（如用户名、订单号、产品名称等）。

**操作系统（Operating System）**：操作系统是管理计算机硬件与软件资源的计算机程序，同时也是计算机系统的内核与基石。它负责控制和管理计算机的硬件和软件资源，提供用户与计算机硬件之间的接口，以及为用户提供各种服务。

**接口（Interface）**：接口在编程和系统设计中有多种含义。通常，它指的是两个不同系统或组件之间的交互点，通过预定义的协议、规范或方法，使得这些系统或组件能够相互通信和协作。在面向对象编程中，接口是一组方法的声明，但不包含方法的具体实现，由类来实现这些接口。

**扩展点（Extension Point）**：扩展点是指在软件系统中预留的、用于添加新功能或修改现有功能的点。通过扩展点，开发人员可以在不修改系统核心代码的情况下，为系统添加新的插件、模块或扩展包，从而增强系统的功能和灵活性。

## 参考资料

* **项目计划任务书或合同**：包含项目目标、范围、时间线和资源分配等关键信息的官方文件。
* **上级机关的批文**：如有，涉及项目批准和指导方针的正式文件。
* **需求规格说明书**：详细描述了用户需求和系统功能的文档，作为设计输入的基础。
* **技术可行性报告**：评估技术方案可行性和风险的报告，为设计决策提供依据。
* **项目开发计划**：概述项目开发的整体计划，包括各阶段的时间表和里程碑。
* **软件架构设计标准**：指导软件架构设计的行业标准或企业内部标准。
* **编程语言和开发环境文档**：使用的开发语言（如Java、Python等）的官方文档和开发工具的手册。
* **第三方库和框架文档**：项目中使用的任何第三方库或框架的官方文档。

# 范围

开发一款集成人工智能技术的移动应用程序，专注于鸟类的快速识别和相关信息的普及，以提升用户对自然和鸟类多样性的认知。

## 系统主要目标

本鸟类识别App旨在为用户提供一个便捷、准确且富有教育意义的鸟类识别工具，以满足广大鸟类爱好者、自然观察者和教育工作者在户外探险、生态学习及科研工作中的需求。

* **提供快速准确的鸟类识别：**通过先进的图像识别技术和深度学习算法，系统能够实时、准确地识别用户拍摄或上传的鸟类图片，并提供详细的鸟类信息，如种类、学名、分布区域等。
* **增强用户体验：**设计直观友好的用户界面和交互方式，使用户能够轻松上手并享受使用过程。同时，提供个性化的设置选项，如识别偏好、界面主题等，以满足不同用户的需求。
* **提供丰富的鸟类信息：**除了基本的识别功能外，系统还将提供详细的鸟类信息，包括物种介绍、生活习性、栖息地描述等，帮助用户更深入地了解鸟类知识。
* **支持社区互动与分享：**建立一个用户社区，允许用户分享自己的鸟类观察记录、图片和心得。用户可以在社区中交流经验、提问解答，共同促进鸟类知识的传播和学习。
* **持续优化与更新：**通过收集用户反馈和数据分析，不断优化系统的识别算法和用户体验，提高识别的准确性和效率。同时，定期更新鸟类数据库，以覆盖更多的鸟类种类和分布区域。
* **提供教育支持：**针对教育工作者和学生群体，提供教育版App或相关资源，帮助他们更好地利用鸟类识别功能进行生态教育、科学研究和项目实践。
* 促进生态保护意识：通过展示鸟类的美丽和多样性，以及它们在生态系统中的重要作用，激发用户对自然环境的热爱和保护意识，促进生态环境的可持续发展。

## 2.2主要软件需求

* **鸟类识别功能**：用户可以通过上传鸟类照片，利用系统内置的深度学习算法来实现快速和准确的鸟类识别。系统将返回鸟类的基本信息、生态习性和保护状况等详细资料，帮助用户增加对鸟类的认知。
* **社交互动平台**：提供一个社交平台，使用户能够分享自己的鸟类观察成果，与全球的鸟类爱好者交流心得和经验，增加社区的互动性和活跃度。
* **教育普及模块**：集成教育内容，普及鸟类保护知识，提升公众对自然环境保护的意识，特别是对鸟类多样性的认识和保护。
* **数据记录功能**：允许用户记录和存储个人的鸟类观察数据，系统将帮助用户整理和生成个人的鸟类观察日志，便于用户回顾和分享。
* **智能推荐系统**：根据用户的地理位置、季节变化等信息，智能推荐可能观察到的鸟类种类，增加用户发现新物种的机会。
* **用户界面设计**：设计一个直观、易用的用户界面，确保用户能够轻松访问和操作所有功能，包括鸟类识别、社交分享、教育内容和数据记录。
* **多平台兼容性**：确保软件能够在iOS、Android等多种操作系统上无缝运行，提供一致的用户体验。
* **数据安全和隐私保护**：采取必要的技术和管理措施，保护用户数据不被未授权访问或泄露，同时遵守相关的隐私保护法规。
* **可扩展性**：在软件设计中预留接口和扩展点，确保系统能够适应未来功能的增加或技术升级。
* **用户反馈机制**：建立有效的用户反馈渠道，收集用户对软件的意见和建议，不断优化和改进产品。
* **帮助和支持**：提供详尽的帮助文档、常见问题解答（FAQ）以及客户支持服务，确保用户在使用过程中能够得到及时的帮助。

## 2.3设计约束、限制

1. **技术实现的可行性**：所有功能需求必须在当前技术框架内可行。我们需要评估现有技术是否能够支持深度学习算法的集成、用户界面的交互性以及数据存储和处理的需求。
2. **成本控制**：开发预算将限制我们能够投入的资源和人力。这包括硬件、软件、人员培训和市场推广等方面的成本。
3. **时间限制**：项目必须在既定的时间框架内完成，包括需求收集、设计、开发、测试和部署等各个阶段。
4. **用户隐私和数据保护**：系统必须符合数据保护法规，确保用户数据的安全和隐私。
5. **硬件兼容性**：应用需要在不同型号和操作系统版本的移动设备上运行，这要求我们在多种设备上进行彻底的测试，以确保兼容性和性能。
6. **可维护性和可扩展性**：软件架构应该允许未来的功能扩展和技术升级，同时保持代码的清晰性和可维护性。
7. **性能要求**：系统需要快速响应用户操作，特别是在处理图像识别和数据检索时，以提供流畅的用户体验。
8. **法律和合规性**：遵守所有相关的法律法规，包括但不限于版权、商标和专利法律，以及特定地区的行业规定。

# 软件系统结构设计

## 软件体系结构

### 软件程序结构图

1. **根节点**：《翼界探秘系统》
   * 描述：整个系统的顶级容器，包含所有子系统和模块。
2. **第一层分支**：主要子系统
   * **用户界面子系统**（UI System）
     + 负责展示信息和用户交互。
   * **业务逻辑子系统**（Business Logic System）
     + 处理应用程序的核心业务逻辑。
   * **数据访问子系统**（Data Access System）
     + 负责数据的持久化和检索。
   * **服务子系统**（Services System）
     + 提供系统功能相关的服务。
3. **第二层分支**：用户界面子系统模块
   * **登录模块**（Login Module）
     + 处理用户登录逻辑。
   * **主界面模块**（Main Interface Module）
     + 展示应用的主要功能入口。
4. **第二层分支**：业务逻辑子系统模块
   * **鸟类识别模块**（Bird Identification Module）
     + 执行鸟类识别算法。
   * **社交互动模块**（Social Interaction Module）
     + 处理用户间的社交互动。
5. **第二层分支**：数据访问子系统模块
   * **用户数据访问模块**（User Data Access Module）
     + 管理用户数据的存储和检索。
   * **鸟类数据库访问模块**（Bird Database Access Module）
     + 提供鸟类信息的数据库访问。
6. **第二层分支**：服务子系统模块
   * **智能推荐服务模块**（Intelligent Recommendation Service Module）
     + 根据用户数据提供个性化推荐。
   * **通知服务模块**（Notification Service Module）
     + 处理系统通知和提醒。
7. **第三层分支示例**（以鸟类识别模块为例）：
   * **图像处理组件**（Image Processing Component）
     + 负责图片的基本处理。
   * **识别算法组件**（Recognition Algorithm Component）
     + 执行深度学习识别算法。
8. **附加组件**：
   * **安全模块**（Security Module）
     + 负责用户认证和数据加密。
   * **异常处理模块**（Exception Handling Module）
     + 统一处理系统异常。

### 模块描述

### 模块描述表

| **模块名称** | **子系统名称** | **系统名称** | **接口说明** | **输入** | **输出** | **功能说明** | **运行环境说明** | **调用关系说明** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 登录模块 | 用户界面子系统 | 翼界探秘系统 | 用户登录接口，密码加密传输 | 用户名（字符串），密码（字符串） | 登录成功（布尔值），用户信息（对象） | 验证用户凭据，提供系统访问权限。 | 需要支持HTTPS的安全网络连接，兼容主流浏览器和操作系统。 | 调用模块：无 被调用模块：用户信息验证模块 |
| 鸟类识别模块 | 业务逻辑子系统 | 翼界探秘系统 | 图像识别接口 | 用户上传的图片（二进制数据） | 识别结果（鸟类名称，置信度） | 利用深度学习算法识别图片中的鸟类。 | 需要高性能GPU进行图像处理，服务器需具备足够的存储空间。 | 调用模块：图像上传模块 被调用模块：数据库访问模块 |
| 数据记录模块 | 数据访问子系统 | 翼界探秘系统 | 数据存储接口 | 用户观察数据（结构化数据） | 数据存储确认（布尔值） | 存储用户观察到的鸟类数据，生成观察日志。 | 需要数据库支持，如MySQL或MongoDB，服务器需具备数据备份机制。 | 调用模块：用户界面模块 被调用模块：无 |
| 智能推荐模块 | 服务子系统 | 翼界探秘系统 | 推荐服务接口 | 用户信息，地理位置（可选） | 推荐鸟类列表（鸟类名称，图片链接） | 根据用户偏好和条件推荐可能观察到的鸟类。 | 需要访问用户数据库和鸟类数据库，服务器需具备数据分析能力。 | 调用模块：用户界面模块 被调用模块：数据记录模块 |
| 社交互动模块 | 业务逻辑子系统 | 翼界探秘系统 | 社交分享接口 | 用户分享内容，分享渠道 | 分享成功确认（布尔值） | 允许用户分享观察成果到社交平台。 | 需要集成社交媒体API，服务器需支持高并发处理。 | 调用模块：用户界面模块 被调用模块：通知服务模块 |

## 功能需求追溯

|  |
| --- |
|  |
| **功能需求编号** | **功能需求描述** | **模块1** | **模块2** | **...** | **模块M** |
| FR-01 | 用户账户注册与管理 | √ |  | ... |  |
| FR-02 | 鸟类图片上传与存储 |  | √ | ... |  |
| FR-03 | 鸟类识别与分类 | √ | √ | ... |  |
| FR-04 | 社交分享功能 |  | √ | ... | √ |
| FR-05 | 教育内容展示 |  |  | ... | √ |
| FR-06 | 数据记录与查询 | √ |  | ... | √ |
| FR-07 | 智能推荐系统 |  |  | ... | √ |
| FR-08 | 用户反馈收集 | √ |  | ... |  |
| ... | ... | ... | ... | ... | ... |
| FR-m | 系统性能监控 |  | √ | ... |  |

# 数据设计

#### 4.1 数据模型设计

* 定义数据模型以支持系统功能，包括用户信息、鸟类数据库、观察记录等实体。
* 实体属性包括但不限于用户ID、用户名、密码（加密存储）、鸟类名称、分类、描述、图片链接等。

#### 4.2 数据库结构

* 设计关系型数据库或非关系型数据库的结构，以优化查询效率和数据完整性。
* 创建必要的表、索引和关系，例如外键约束，以确保数据的一致性。

#### 4.3 数据库规范化

* 应用数据库规范化原则，减少数据冗余，提高数据的一致性和可维护性。

#### 4.4 数据访问对象（DAO）设计

* 设计数据访问对象层，以抽象和封装对数据库的访问逻辑。
* DAO模式提供统一的接口来执行CRUD（创建、读取、更新、删除）操作。

# 接口设计

## 用户界面设计规则

用户界面（UI）设计是确保《翼界探秘系统》提供直观、易用且吸引人的用户体验的关键环节。以下是用户界面设计的基本规则：

* **一致性**：整个应用的UI元素应保持风格和行为的一致性，包括按钮、图标、字体和颜色方案，以减少用户的学习成本。
* **直观性**：界面设计应直观易懂，让用户能够快速理解如何操作，避免复杂的导航结构。
* **反馈**：系统应对用户的操作提供即时反馈，无论是通过动画效果、提示信息还是声音提示。
* **可用性**：设计应考虑所有用户，包括不同年龄、技术和文化背景的用户，确保界面易于访问和使用。
* **布局**：界面布局应合理，将最重要的信息和功能放在显眼位置，同时保持内容的逻辑性和清晰性。
* **导航**：提供清晰的导航路径，使用户能够轻松地在不同的功能模块间切换。

## 内部接口设计

内部接口设计关注软件内部模块间的交互和数据流动。以下是内部接口设计的关键点：

* **定义清晰的接口规范**：为每个模块定义清晰的输入输出接口，包括数据格式、参数类型等。
* **模块解耦**：设计接口以减少模块间的依赖，提高系统的可维护性和可扩展性。
* **数据封装**：确保模块内部数据处理的逻辑不外泄，通过接口与外界交互。
* **服务契约**：明确服务提供者和消费者之间的契约，包括调用协议、数据交换格式等。
* **异常处理**：设计统一的异常处理机制，确保模块间的稳定交互。
* **性能考量**：接口设计时考虑性能影响，如避免频繁的数据复制、合理使用同步异步调用等。
* **安全性**：确保接口设计满足安全要求，如认证、授权、数据加密等。
* **版本控制**：接口可能随着时间而变化，设计时需考虑版本控制和向后兼容性。

## 外部接口设计

* **标准化**：使用标准化的协议和数据格式，如RESTful API、JSON或XML，以便于不同系统之间的互操作性。
* **安全性**：确保所有外部接口都有适当的安全措施，包括认证、授权、数据加密和安全传输（如使用HTTPS）。
* **版本管理**：为外部接口提供版本控制，以支持向后兼容性和平滑过渡到新版本。
* **错误处理**：定义清晰的错误响应机制，以便调用方能够理解并适当处理接口调用中的错误。
* **文档化**：提供详细的接口文档，包括请求和响应的数据结构、参数说明、状态码等，以便于第三方开发者理解和集成。
* **限制和配额**：对于可能影响系统性能的接口，设置合理的使用限制和配额，以防止滥用。
* **监控和日志记录**：实施监控机制以跟踪接口的使用情况，并记录必要的日志信息，以便于问题诊断和性能优化。
* **依赖管理**：明确外部接口对第三方服务的依赖，并评估这些依赖的稳定性和可靠性。
* **数据交换**：设计数据交换机制，确保数据在不同系统间传输的准确性和完整性。
* **集成测试**：与外部系统进行集成测试，确保接口在实际运行环境中的兼容性和性能。
* **用户隐私**：在设计涉及用户数据的外部接口时，严格遵守隐私保护法规，确保用户信息的安全。

# 出错处理设计

#### 6.1 错误分类与编号

1. **用户输入错误**（Error Code: UI-001）
2. **系统响应超时**（Error Code: SYS-002）
3. **数据访问失败**（Error Code: DA-003）
4. **网络连接问题**（Error Code: NET-004）
5. **权限不足**（Error Code: AUTH-005）
6. **服务不可用**（Error Code: SVC-006）
7. **资源文件缺失**（Error Code: RES-007）
8. **第三方服务失败**（Error Code: 3RD-008）

#### 6.2 出错时的信息显示界面

* 对于所有错误，系统应提供一个清晰的错误提示界面，包括错误代码和用户友好的错误描述。
* 错误提示应避免技术性语言，使用易于理解的日常用语。
* 提供错误发生时的上下文信息，帮助用户理解错误发生的可能原因。

#### 6.3 相应的补救措施

1. **用户输入错误**：
   * 显示提示信息，指明输入错误的性质，并允许用户重新输入。
2. **系统响应超时**：
   * 建议用户检查网络连接或稍后重试，并提供一个重试按钮。
3. **数据访问失败**：
   * 记录错误详情并通知技术支持团队，同时提示用户联系客服。
4. **网络连接问题**：
   * 检测网络状态，如果可能，自动重连；否则提示用户检查网络连接。
5. **权限不足**：
   * 明确告知用户所需权限，引导用户进行权限设置或联系管理员。
6. **服务不可用**：
   * 通知用户服务当前不可用，并提供一个预计恢复时间。
7. **资源文件缺失**：
   * 尝试恢复缺失文件或重新下载，如果失败则提示用户重新安装应用。
8. **第三方服务失败**：
   * 记录失败详情，提示用户该功能依赖的第三方服务当前不可用，并建议稍后重试。