系统设计说明书

福州大学 2022 级软件工程实践“研途无忧”小组

2024 年 10 月 30 日

目录

第一章　引言

1.1　编写目的

本系统设计说明书旨在明确“福小研”软件的系统设计方案，为项目的开发、测试和维护提供指导。通过本说明书，可以使项目组成员对系统的整体架构、功能模块、技术选型等有清晰的了解，为后续工作的开展奠定基础。

1.2　项目背景

- **软件名称**：福小研
- **项目任务提出者**：福州大学计算机与大数据学院教学办
- **项目任务**：软件工程实践
- **项目开发者**：福州大学 2022 级软件工程实践“研途无忧”小组

本项目旨在为考研学子提供全面的备考支持，包括智能学习规划、资源推荐、心理支持等功能。项目需求基于对实际用户的问卷调查和小组讨论，与其他软件或组织机构无关。

1.3　参考材料

1.《计算机软件需求规格说明规范》（GB/T 9385-2008）

2.《概要设计说明书》

第二章　系统总体设计

2.1　整体架构

系统采用客户端-服务器（C/S）架构，基于 **uni-app** 和 **uniCloud** 云开发平台。整体架构包括以下部分：

- **客户端**：使用 uni-app 框架，基于 Vue3 和 JavaScript 进行开发，生成跨平台移动应用，支持 iOS、Android 和小程序等多端。
- **服务器端**：使用 uniCloud 云开发平台，处理业务逻辑、数据存储和接口提供。
- **数据库**：基于 uniCloud 提供的云数据库，存储用户数据、学习资源、社区内容等信息。

2.2　整体功能架构

系统功能主要分为以下模块：

- **个人办公**：通知公告、待办事宜管理。
- **学习管理**：日历打卡、学习计划、自习室、学情分析。
- **资源中心**：课程推荐、题库、考研资讯。
- **社区互动**：论坛交流、问题求助、活动打卡。
- **智能助手**：AI 问答、智能解答、个性化服务。

2.3　整体技术架构

系统技术架构包括：

- **前端技术**：
  - 使用 **uni-app** 框架，基于 **Vue3** 和 **JavaScript** 开发，支持多端发布。
  - 采用 **HTML** 和 **CSS** 进行页面布局和样式设计，提升用户体验。
- **后端技术**：

  - 使用 **uniCloud** 云开发平台，提供云函数、云数据库、云存储等服务。
  - 业务逻辑在云函数中实现，前端直接调用云函数接口。

- **数据库**：
  - 采用 **uniCloud 云数据库**，存储结构化数据，支持高并发和大数据量。

- **人工智能**：
  - 集成 **OpenAI** 的 **GPT-4** API，实现 AI 问答和智能解答功能。

2.4 设计目标

系统设计目标如下：

- **满足用户需求**：提供全面的考研备考支持，提升用户学习效率。
- **高可用性**：系统稳定运行，支持大量用户同时在线。
- **安全性**：保护用户数据安全，防止信息泄露。
- **可扩展性**：系统架构设计支持功能扩展和升级。
- **易维护性**：代码规范，文档齐全，便于后期维护。

2.4.1 总体原则

2.5.1 总体原则

系统设计遵循以下原则：

- **用户至上**：以用户需求为导向，提供良好的用户体验。
- **技术先进**：采用先进技术，提升系统性能和竞争力。
- **安全可靠**：确保系统的安全性和可靠性。
- **标准规范**：遵循相关行业标准和规范。

2.4.2 实用性和先进性

2.5.2 实用性和先进性

系统功能设计以实用性为基础，采用先进的技术手段，如 uni-app 框架、云开发平台、人工智能等，提升用户体验和系统性能。

2.4.3 标准化、开放性、兼容性

2.5.3 标准化、开放性、兼容性

系统遵循国际标准，采用开放的技术架构，确保与其他系统的兼容性和可扩展性。使用 uni-app 框架可以兼容多端，满足不同用户的使用需求。

2.4.4 高可靠性、稳定性

2.5.4 高可靠性、稳定性

依托 uniCloud 云服务的高可靠性，通过云函数和云数据库的稳定性能，保证系统的高可用性和稳定性。

2.4.5 易用性

2.5.5 易用性

界面友好，操作简便，采用 Vue3 的组件化开发方式，提高开发效率和用户交互体验。

2.4.6 灵活性和可扩展性

2.5.6 灵活性和可扩展性

模块化设计，支持功能的灵活组合和扩展，云开发平台便于后期功能的快速上线和升级。

2.4.7 经济性和投资保护

2.5.7 经济性和投资保护

利用云开发平台减少服务器投入和运维成本，合理控制开发和维护成本，保护投资，实现资源的高效利用。

第三章　系统功能模块详细设计

3.1　个人办公

3.1.1　通知公告

3.1.1 通知公告

功能描述：提供系统通知和公告的发布与展示，用户可以查看最新的通知信息。
实现方案：

- 后端在 uniCloud 中创建云函数，提供通知公告的 CRUD 接口。
- 前端使用 uni-app 实现通知列表和详细内容的展示，支持推送通知功能。

3.1.2　待办事宜

3.1.2 待办事宜

功能描述：用户可以添加、查看、编辑和删除待办事项，帮助管理学习和生活任务。
实现方案：

- 待办事项数据存储在 uniCloud 云数据库中，关联用户 ID。
- 前端提供待办事项的管理界面，使用 Vue3 组件实现，支持提醒功能。

3.2　学习管理

3.2.1　日历打卡

3.2.1 日历打卡

功能描述：用户可以在日历上记录每天的学习进度和完成情况，形成学习打卡习惯。
实现方案：

- 后端创建云函数，管理用户的打卡数据，包括添加、查询和更新打卡记录。
- 前端使用 uni-app 的日历组件，展示用户的打卡历史，并允许用户进行打卡操作。
- 实现打卡提醒功能，通过云函数定时推送通知，提醒用户进行每日打卡。

3.2.2 学习计划

3.2.2 学习计划

功能描述：用户可以制定详细的学习计划，设定学习目标、科目和进度，系统根据计划进行跟踪和提醒。

实现方案：

- 后端在云数据库中设计学习计划集合，提供云函数接口进行计划的 CRUD 操作。
- 前端提供学习计划的创建、编辑和查看界面，使用 Vue3 组件实现。
- 系统根据学习计划设定的时间节点，自动发送提醒通知，帮助用户按时完成计划。
- 实现进度统计和分析功能，展示用户的学习进展情况。

3.2.3 自习室

3.2.3 自习室

功能描述：提供虚拟自习室，用户可以在此集中学习，系统提供番茄钟等工具，提升学习效率。

实现方案：

- 后端创建自习室集合，记录用户的自习时间和使用情况。
- 前端实现自习室界面，包括番茄钟计时器、学习时长统计等功能。
- 系统提供学习资料的实时共享功能，用户可以在自习室内分享学习资源。
- 实现学习社区互动功能，用户可以在自习室内进行交流和讨论，互相激励。

3.2.4 学情分析

3.2.4 学情分析

功能描述：系统根据用户的学习数据，生成学情分析报告，帮助用户了解自己的学习状态和改进方向。

实现方案：

- 后端收集和处理用户的学习数据，包括打卡记录、学习时长、学习计划完成情况等。
- 使用数据分析和可视化工具，生成图表和报告，展示用户的学习成果和不足。
- 前端提供学情分析报告的展示界面，用户可以查看详细的学习分析结果。
- 系统根据分析结果，提供个性化的学习建议和改进措施，帮助用户优化学习策略。

3.3 资源中心

3.3.1 课程推荐

3.3.1 课程推荐

功能描述：根据用户的学习需求和兴趣，智能推荐相关课程资源，提升学习效果。

实现方案：

- 后端集成第三方课程平台 API，获取最新的课程资源。
- 创建课程推荐算法，根据用户的学习历史、兴趣标签和学习目标，生成个性化的课程推荐列表。
- 前端展示推荐课程列表，提供课程详情、学习路径和报名功能。
- 实现课程收藏和分享功能，用户可以将感兴趣的课程收藏或分享给其他用户。

3.3.2 题库

3.3.2 题库

功能描述：提供丰富的考研题库，用户可以进行在线练习和模拟测试，提升应试能力。

实现方案：

- 后端设计题库集合，存储不同科目的试题和答案。
- 创建云函数，提供题库的查询和管理接口，包括随机抽题、题目分类等功能。
- 前端实现题库浏览和在线练习界面，支持单选、多选、填空等题型。
- 实现练习记录和成绩分析功能，帮助用户了解自己的薄弱环节。

3.3.3 考研资讯

3.3.3 考研资讯

功能描述：提供最新的考研资讯和动态，包括考试政策、报名信息、复习资料等，帮助用户及时获取相关信息。

实现方案：

- 后端集成新闻 API 或自行采集考研相关资讯，存储在云数据库中。
- 创建云函数，提供资讯的查询和管理接口。
- 前端展示考研资讯列表，用户可以浏览最新的考研动态和相关信息。
- 实现资讯分类和搜索功能，方便用户快速找到感兴趣的内容。
- 提供资讯收藏和分享功能，用户可以将重要资讯保存或分享给他人。

3.4 社区互动

3.4.1 论坛交流

3.4.1 论坛交流

功能描述：为用户提供一个交流平台，用户可以在论坛上发帖、回复和互动，分享学习经验和备考心得。

实现方案：

- 后端设计论坛帖子和评论集合，存储用户的发帖和回复内容。
- 创建云函数，提供帖子和评论的 CRUD 接口。
- 前端实现论坛界面，包括发帖、回复、点赞、收藏等功能。
- 实现帖子分类和标签功能，帮助用户快速找到感兴趣的讨论主题。
- 提供搜索和筛选功能，方便用户查找特定的帖子和讨论内容。

3.4.2 问题求助

3.4.2 问题求助

功能描述：用户可以在社区中发布问题，寻求其他用户或专家的帮助和解答，解决学习中的困惑。

实现方案：

- 后端设计问题求助集合，存储用户发布的求助问题和回复内容。
- 创建云函数，提供问题求助的 CRUD 接口。
- 前端实现问题发布和查看界面，支持问题分类、标签和回复功能。
- 实现问题提醒和通知功能，当有新回复时，及时通知提问用户。
- 提供专家认证和回答功能，邀请资深学长学姐或专家参与问题解答，提升求助质量。

3.4.3 活动打卡

3.4.3 活动打卡

功能描述：组织各类学习活动，用户可以参与并进行打卡，形成良好的学习习惯和社区互动。

实现方案：

- 后端设计活动和打卡集合，存储活动详情和用户的打卡记录。
- 创建云函数，提供活动发布、参与和打卡的接口。
- 前端实现活动列表和打卡界面，用户可以查看活动详情并进行打卡操作。
- 实现活动奖励和排行榜功能，激励用户积极参与和打卡。
- 提供活动分享功能，用户可以将活动信息分享至朋友圈或其他社交平台，扩大活动影响力。

3.5 智能助手

3.5.1 AI 问答

3.5.1 AI 问答

功能描述：集成 AI 技术，用户可以通过自然语言与系统进行互动，获取学习建议和问题解答。

实现方案：

- 后端集成 OpenAI 的 GPT-4 API，处理用户的自然语言输入并生成响应。
- 创建云函数，作为 AI 问答的中介，接收用户问题，调用 GPT-4 API，并返回回答。
- 前端实现 AI 问答界面，用户可以输入问题并查看 AI 的回答。
- 实现会话记录功能，保存用户与 AI 的对话历史，方便用户回顾和参考。
- 提供个性化学习建议，根据用户的学习数据和目标，生成定制化的学习计划和建议。

3.5.2 智能解答

3.5.2 智能解答

功能描述：系统根据用户的学习情况和数据，提供智能解答和辅助决策，帮助用户优化学习策略。

实现方案：

- 后端分析用户的学习数据，结合 AI 技术，生成智能解答和建议。
- 创建云函数，处理用户的学习数据请求，返回智能分析结果。
- 前端展示智能解答和建议，用户可以根据系统提供的信息调整学习计划和策略。
- 实现学习数据的可视化展示，如图表和报告，帮助用户直观理解学习情况。

3.5.3 个性化服务

3.5.3 个性化服务

功能描述：根据用户的学习习惯和需求，提供个性化的学习资源推荐和服务，提升学习效果。

实现方案：

- 后端收集和分析用户的学习数据和行为，生成个性化推荐模型。
- 创建云函数，提供个性化推荐的 API 接口，返回适合用户的学习资源和服务。
- 前端展示个性化推荐内容，用户可以浏览、收藏和使用推荐的资源。
- 实现用户偏好设置，允许用户自定义推荐参数和偏好，提高推荐的准确性。
- 提供个性化学习提醒和通知，根据用户的学习进度和计划，发送定制化的学习提醒。

第四章　性能设计

4.1　响应时间

4.1 响应时间

系统应保证关键操作的响应时间不超过 1 秒，一般操作的响应时间不超过 3 秒，确保用户操作的流畅性。

4.2　并发用户数

4.2 并发用户数

系统设计支持同时在线用户数达到 1000 人，并利用 uniCloud 的弹性伸缩能力，满足高并发需求。

第五章　数据库设计

数据库采用 uniCloud 云数据库，主要数据表（集合）设计如下：

- **users**：存储用户信息（用户名、密码、个人简介、头像、福币等）。
- **tasks**：存储待办事宜（任务内容、开始时间、结束时间、提醒设置等）。
- **announcements**：存储通知公告（标题、内容、发布时间等）。
- **study_plans**：存储学习计划（学习目标、科目、进度等）。
- **resources**：存储学习资源（课程、题库、资料链接等）。
- **posts**：存储社区帖子（帖子内容、作者、评论、点赞等）。
- **comments**：存储帖子评论（评论内容、作者、回复对象等）。
- **courses**：存储课程信息（课程名称、课程链接、是否免费等）。
- **leaderboard**：存放用户的福币和更新信息，用于显示排行榜。
- **transactions**：存放福币的支出收入交易记录。
- **dailytasks**：存放每日任务的相关信息。

5.1　用户集合（users）

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成，唯一标识每个用户
account	String	用户账号
password	String	用户密码
avatarUrl	String	头像文件 URL
nickname	String	用户昵称
gender	String	用户性别
year	String	考研年份
major	String	报考专业

school	String	目标院校
createdAt	Long	注册时间戳
updatedAt	Long	资料更新时间戳

5.1.1 MongoDB Schema

以下是用户集合的 MongoDB Schema 设计：

```
{
  "bsonType": "object",
  "required": ["account", "password", "createdAt"],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成，唯一标识每个用户"
    },
    "account": {
      "bsonType": "string",
      "description": "用户账号"
    },
    "password": {
      "bsonType": "string",
      "description": "用户密码"
    },
    "avatarUrl": {
      "bsonType": "string",
      "description": "头像文件URL"
    },
    "nickname": {
      "bsonType": "string",
      "description": "用户昵称"
    },
    "gender": {
      "bsonType": "string",
      "description": "用户性别"
    },
    "year": {
      "bsonType": "string",
      "description": "考研年份"
    },
    "major": {
      "bsonType": "string",
      "description": "报考专业"
    },
    "school": {
```

```
      "bsonType": "string",
      "description": "目标院校"
    },
    "createdAt": {
      "bsonType": "long",
      "description": "注册时间戳"
    },
    "updatedAt": {
      "bsonType": "long",
      "description": "资料更新时间戳"
    }
  }
}
```

Listing 5.1: 用户集合 MongoDB Schema

5.1.2 约束条件

- **必填字段**：`account`、`password` 和 `createdAt`。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **唯一性**：`account` 字段应设置为唯一，防止重复账号注册。
- **数据格式**：`createdAt` 和 `updatedAt` 应为时间戳，确保时间的准确性。

5.2 社区帖子集合（posts）

根据最新的数据库设计，社区帖子集合的详细设计如下：

5.2.1 字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成，唯一标识每个帖子
userAvatar	String	用户头像的 URL
userNickname	String	用户昵称
content	Object	发帖内容（可选）
content.text	String	帖子文字内容（可选）
content.images	Array	帖子中包含的图片 URL 数组
content.topics	Array	帖子关联的话题标签（可选）
likesCount	Int	点赞人数，最小值为 0
favoritesCount	Int	收藏人数，最小值为 0
sharesCount	Int	转发人数，最小值为 0

postDate	Date	发帖日期
hot	Boolean	是否是热门帖子

5.2.2 MongoDB Schema

以下是社区帖子集合的 MongoDB Schema 设计：

```
{
  "bsonType": "object",
  "required": ["userAvatar", "userNickname", "postDate"],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成，唯一标识每个帖子"
    },
    "userAvatar": {
      "bsonType": "string",
      "description": "用户头像的URL"
    },
    "userNickname": {
      "bsonType": "string",
      "description": "用户昵称"
    },
    "content": {
      "bsonType": "object",
      "description": "发帖内容（可选）",
      "properties": {
        "text": {
          "bsonType": "string",
          "description": "帖子文字内容（可选）",
          "minLength": 0
        },
        "images": {
          "bsonType": "array",
          "items": {
            "bsonType": "string",
            "description": "图片的URL"
          },
          "description": "帖子中包含的图片URL数组",
          "minItems": 0
        },
        "topics": {
          "bsonType": "array",
          "items": {
            "bsonType": "string",
            "description": "关联的话题标签（可选）"
```

```
            },
            "description": "帖子关联的话题",
            "minItems": 0
          }
        }
      },
      "likesCount": {
        "bsonType": "int",
        "description": "点赞人数",
        "minimum": 0
      },
      "favoritesCount": {
        "bsonType": "int",
        "description": "收藏人数",
        "minimum": 0
      },
      "sharesCount": {
        "bsonType": "int",
        "description": "转发人数",
        "minimum": 0
      },
      "postDate": {
        "bsonType": "date",
        "description": "发帖日期"
      },
      "hot": {
        "bsonType": "boolean",
        "description": "是否是热门帖子"
      }
    }
}
```

Listing 5.2: 社区帖子集合 MongoDB Schema

5.2.3 约束条件

- **必填字段**：`userAvatar`、`userNickname` 和 `postDate`。
- **可选字段**：`content`（包含 `text`、`images` 和 `topics`）。
- **数值约束**：`likesCount`、`favoritesCount` 和 `sharesCount` 最小值为 0。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。

5.3 课程集合（courses）

根据最新的数据库设计，课程集合的详细设计如下：

5.3.1 字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成，唯一标识每个课程
courseName	String	课程名称
courseLink	String	课程链接
isFree	Boolean	是否免费（true 表示免费，false 表示收费）

5.3.2 MongoDB Schema

以下是课程集合的 MongoDB Schema 设计：

```
{
  "bsonType": "object",
  "required": ["courseName", "courseLink", "isFree"],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成，唯一标识每个课程"
    },
    "courseName": {
      "bsonType": "string",
      "description": "课程名称"
    },
    "courseLink": {
      "bsonType": "string",
      "description": "课程链接"
    },
    "isFree": {
      "bsonType": "boolean",
      "description": "是否免费（true表示免费，false表示收费）"
    }
  }
}
```

Listing 5.3: 课程集合 MongoDB Schema

5.3.3 约束条件

- **必填字段**：`courseName`、`courseLink` 和 `isFree`。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **值范围**：`isFree` 为布尔类型，只接受 `true` 或 `false`。

5.4 学习计划集合（study_plans）

根据最新的数据库设计，对学习计划集合进行了扩展，新增了 ‘planName‘ 和 ‘status‘ 字段。详细设计如下：

5.4.1 字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成，唯一标识每个学习计划
planName	String	学习计划具体名称
status	String	学习状态（未开始，已完成）
userId	ObjectId	关联用户 ID
createdAt	Date	创建时间
updatedAt	Date	更新时间

5.4.2 MongoDB Schema

以下是学习计划集合的 MongoDB Schema 设计：

```
{
  "bsonType": "object",
  "required": ["planName", "status", "userId", "createdAt"],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成，唯一标识每个学习计划"
    },
    "planName": {
      "bsonType": "string",
      "description": "学习计划具体名称"
    },
    "status": {
      "bsonType": "string",
      "description": "学习状态（未开始，已完成）",
      "enum": ["未开始", "已完成"]
    },
    "userId": {
      "bsonType": "objectId",
      "description": "关联用户ID"
    },
    "createdAt": {
      "bsonType": "date",
      "description": "创建时间"
    },
    "updatedAt": {
```

```
      "bsonType": "date",
      "description": "更新时间"
    }
  }
}
```

Listing 5.4: 学习计划集合 MongoDB Schema

5.4.3 约束条件

- **必填字段**：`planName`、`status`、`userId` 和 `createdAt`。
- **值范围**：`status` 仅接受 未开始和 已完成两种值。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **参照完整性**：`userId` 必须关联到 `users` 集合中的有效 `_id`，确保数据的关联性。

5.5 leaderboard 集合（leaderboard）

排行榜数据库存放用户的福币和更新信息，通过用户 id 关联 users 数据库，获取头像和昵称等用户个人信息，显示在福榜上。

5.5.1 字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成
user_id	ObjectId	用户 ID，关联 users 集合中的 _id
score	Int	用户的福币分数
updatedAt	Date	分数的最后更新时间

5.5.2 MongoDB Schema

以下是排行榜集合的 MongoDB Schema 设计：

```
{
  "bsonType": "object",
  "required": [
    "user_id",
    "score",
    "updatedAt"
  ],
  "properties": {
    "_id": {
      "bsonType": "objectId",
```

```
      "description": "ID, 系统自动生成"
    },
    "user_id": {
      "bsonType": "objectId",
      "description": "用户ID, 关联 users 集合中的 _id"
    },
    "score": {
      "bsonType": "int",
      "description": "用户的福币分数"
    },
    "updatedAt": {
      "bsonType": "date",
      "description": "分数的最后更新时间"
    }
  }
}
```

Listing 5.5: 排行榜集合 MongoDB Schema

5.5.3 约束条件

- **必填字段**：`user_id`、`score` 和 `updatedAt`。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **参照完整性**：`user_id` 必须关联到 `users` 集合中的有效 `_id`，确保数据的关联性。
- **权限控制**：`delete` 操作被禁止，仅允许读取、创建和更新。

5.6 transactions 集合（transactions）

交易记录数据库存放福币的支出收入交易记录。

5.6.1 字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成
user_id	ObjectId	用户的唯一 ID，关联用户表
description	String	交易描述，如’ 签到奖励’、’ 兑换商品’
amount	Int	交易金额，正数表示收入，负数表示支出
date	String	交易日期，格式如’2020.9.19’

5.6.2 MongoDB Schema

以下是交易记录集合的 MongoDB Schema 设计：

```
{
  "bsonType": "object",
  "required": [
    "user_id",
    "description",
    "amount",
    "date"
  ],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成"
    },
    "user_id": {
      "bsonType": "objectId",
      "description": "用户的唯一ID，关联用户表"
    },
    "description": {
      "bsonType": "string",
      "description": "交易描述，如'签到奖励'、'兑换商品'"
    },
    "amount": {
      "bsonType": "int",
      "description": "交易金额，正数表示收入，负数表示支出"
    },
    "date": {
      "bsonType": "string",
      "description": "交易日期，格式如 '2020.9.19'"
    }
  }
}
```

Listing 5.6: 交易记录集合 MongoDB Schema

5.6.3 约束条件

- **必填字段**：user_id、description、amount 和 date。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **值范围**：amount 为整数，正数表示收入，负数表示支出。
- **日期格式**：date 字段应遵循'YYYY.MM.DD' 格式，确保日期的一致性和可解析性。
- **权限控制**：仅允许读取和创建，不允许更新和删除，以保护交易记录的完整性。

5.7 dailytasks 集合（dailytasks）

每日任务数据库存放每日任务的相关信息。

5.7.1 字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成
user_id	ObjectId	用户的唯一 ID，用于区分不同用户的任务
name	String	任务名称，如’ 签到打卡’
reward	Int	任务奖励数量
completed	Boolean	任务完成状态，true 表示已完成，false 表示未完成
date	String	任务日期，用于区分每日任务

5.7.2 MongoDB Schema

以下是每日任务集合的 MongoDB Schema 设计：

```
{
  "bsonType": "object",
  "required": [
    "user_id",
    "name",
    "reward",
    "completed",
    "date"
  ],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID, 系统自动生成"
    },
    "user_id": {
      "bsonType": "objectId",
      "description": "用户的唯一ID, 用于区分不同用户的任务"
    },
    "name": {
      "bsonType": "string",
      "description": "任务名称, 如'签到打卡'"
    },
    "reward": {
      "bsonType": "int",
      "description": "任务奖励数量"
    },
```

```
    "completed": {
      "bsonType": "bool",
      "description": "任务完成状态，true表示已完成，false表示未完成"
    },
    "date": {
      "bsonType": "string",
      "description": "任务日期，用于区分每日任务"
    }
  }
}
```

Listing 5.7: 每日任务集合 MongoDB Schema

5.7.3 约束条件

- **必填字段**：`user_id`、`name`、`reward`、`completed` 和 `date`。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **值范围**：
  - `reward` 为整数，表示任务奖励数量。
  - `completed` 为布尔类型，只接受 `true` 或 `false`。
  - `date` 字段应遵循'YYYY.MM.DD' 格式，确保日期的一致性和可解析性。
- **权限控制**：`delete` 操作被禁止，仅允许读取、创建和更新，确保每日任务记录的完整性。

5.7.4 其他数据表设计

后续根据实际代码更新中。。。

第六章　接口设计

6.1　登录接口

6.1.1　6.1.1 调用说明

用于用户登录验证，获取访问令牌。

6.1.2　6.1.2 请求报文

请求方式	POST
请求 URL	/login
请求头	Content-Type: application/json

请求体	<pre>{
	"username": "user123",
	"password": "pass123"
	}</pre>

6.1.3　6.1.3 应答报文

状态码	200 OK
响应头	Content-Type: application/json

响应体	<pre>{
	"token": "jwt-token",
	"user": {
	"id": "user-id",
	"username": "user123"
	}
	}</pre>

6.2 查询所有数据列接口

6.2.1 调用说明

此接口用于查询指定数据表的所有字段信息。它对于前端应用来说尤其有用，可以动态地获取数据库结构，从而在需要时调整用户界面。

6.2.2 请求报文

请求方式	GET
请求 URL	/api/metadata/collection_name
请求头	Authorization: Bearer jwt-token

6.2.3 应答报文

状态码	200 OK
响应头	Content-Type: application/json

响应体	`{`
	`"collection": "collection_name",`
	`"fields": ["field1", "field2", "field3", ...]`
	`}`

6.2.4 接口描述

该接口返回指定数据集合的所有字段信息。此操作通常需要适当的权限，以确保只有验证过的用户或管理员能够访问数据结构。返回的字段信息可以用于构建动态表单或报表。

第七章　系统出错处理设计

7.1　出错信息

系统在发生错误时，返回规范的错误码和信息，方便用户和开发者理解。常见错误信息如下：

错误码	说明
400 Bad Request	请求参数错误，服务器无法理解请求。
401 Unauthorized	未授权访问，需要用户登录。
403 Forbidden	无权限操作，用户权限不足。
404 Not Found	资源不存在，无法找到请求的内容。
500 Internal Server Error	服务器内部错误，无法完成请求。

7.2　补救措施

针对不同错误，系统提供相应的提示和引导：

- **请求参数错误**：提示用户检查输入，确保必填项已填写，格式正确。
- **未授权访问**：引导用户登录或注册，获取访问权限。
- **无权限操作**：提示用户权限不足，如需操作请联系管理员。
- **资源不存在**：提示用户内容已被删除或链接错误，提供返回主页的选项。
- **服务器错误**：提示用户稍后重试，或联系客服获取帮助。

7.3　系统维护设计

系统提供日志记录和监控机制，方便排查问题和维护：

- **日志管理**：记录系统操作日志和错误日志，包括用户操作、接口调用、错误信息等。
- **监控报警**：实时监控系统状态，如 CPU、内存、网络等，出现异常及时报警。
- **备份恢复**：定期备份数据库和重要文件，提供数据恢复方案，防止数据丢失。
- **版本管理**：采用代码版本控制工具，如 Git，便于代码的管理和回溯。

第八章　系统处理规定

8.1　输入输出要求

8.1 输入输出要求

系统应支持多种数据格式的输入和输出，确保数据的完整性和一致性。主要要求如下：

- **输入数据**：支持文本、数字、日期、图片、文件等多种类型的数据输入。
- **输出数据**：以 JSON 格式为主，便于数据的解析和展示；支持图片、文件的下载。
- **数据校验**：对用户输入的数据进行校验，防止错误数据或恶意数据的提交。
- **字符编码**：采用 UTF-8 编码，支持多语言字符集。

8.2　数据管理能力要求

8.2 数据管理能力要求

系统应具备高效的数据存储和检索能力，满足以下要求：

- **数据存储**：采用 uniCloud 云数据库，支持自动扩容和高并发访问。
- **数据检索**：支持复杂查询、分页、排序等功能，提供快速的数据访问。
- **数据安全**：对敏感数据进行加密存储，防止未经授权的访问。
- **数据备份**：定期对数据库进行备份，防止数据丢失。

8.3　故障处理要求

8.3 故障处理要求

系统应具备故障自动切换和恢复能力，保证服务的连续性：

- **故障检测**：实时监控系统状态，及时发现故障。
- **自动切换**：当主服务器发生故障时，能自动切换到备用服务器。
- **故障恢复**：提供快速的数据恢复和服务重启机制，减少故障对用户的影响。
- **应急预案**：制定详细的故障处理和应急响应方案，定期演练。

8.4 其他专门要求

8.4 其他专门要求

- **安全要求**：遵循数据安全和隐私保护法规，使用 HTTPS 加密传输，用户敏感信息加密存储。
- **性能要求**：满足高并发、高可用性的需求，利用云函数的弹性伸缩特性。
- **兼容性要求**：兼容主流操作系统和设备，uni-app 框架支持多端发布。
- **可维护性**：代码结构清晰，注释规范，便于团队协作和后期维护。
- **可扩展性**：采用模块化设计，方便功能的增加和系统的升级。