**珞做TodoList系统**

**软件设计说明书**

团队成员一： 李品正（2020300004081）

团队成员二： 贺镇涛（2021300004071）

团队成员三： 洪志枫（2021300004026）

团队成员四： 刘浩 （2021300004039）

**日期： 2024 年 6 月 9 日**

**文档变更历史记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 变更日期 | 变更人员 | 变更内容详情描述 | 版本 |
| 1 | 2024年6月1日 | 贺镇涛 | 撰写了软件设计规格说明书初稿 | V1.0 |
| 2 |  | XXX | 重新检查，解决实际作品与文档之间的不一致 | V2.0 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1引言 4](#_Toc168867625)

[1.1编写目的 4](#_Toc168867626)

[1.2软件系统概述 4](#_Toc168867627)

[1.3定义 5](#_Toc168867628)

[1.4参考资料 5](#_Toc168867629)

[2软件设计约束 6](#_Toc168867630)

[2.1设计目标和原则 6](#_Toc168867631)

[2.2设计约束 7](#_Toc168867632)

[3软件设计 9](#_Toc168867633)

[3.1体系结构设计 9](#_Toc168867634)

[3.2用户界面设计 10](#_Toc168867635)

[3.3用例设计 10](#_Toc168867636)

[3.4类设计 10](#_Toc168867637)

[3.5数据设计 10](#_Toc168867638)

[3.6部署设计 11](#_Toc168867639)

[4 需求跟踪矩阵 12](#_Toc168867640)

# 1引言

## 1.1编写目的

**编写本《珞做 TodoList 团队任务管理系统设计说明书》的目的是为了详细描述系统的架构、模块设计、数据库设计以及界面设计等方面的内容，以便于开发人员理解系统的内部结构和实现细节，明确各个模块之间的功能交互和数据流动，为后续的生产环境部署、开发维护、演化提供清晰的指导，并可用于产品验收和用户参考。**

**本文的预期读者包括但不限于软件开发人员（包括开发主管、技术主管）、测试人员、质量保证人员以及其他需要深入了解系统设计和实现细节的相关人员，同时可供用户参考。**

## 1.2软件系统概述

**-项目名称：珞做TodoList团队任务管理系统**

**-用户（单位）：公众（一般用户）**

**[委托2021HYSE05-校园二手交易平台项目组作为用户代表]**

**-开发者：珞做TodoList项目组(2021HYSE12组)**

**-项目背景和大致需求：**

日益复杂的工作环境和逐渐加快的生活节奏促使人们持续追求工作效率提升和更有效的生活规划。为此，本小组提出了时间管理工具**珞做TodoList**，旨在帮助个人和**团队**在日益复杂的**多任务、多角色场景下有效规划、分配和管理任务。具体包括：**

1. **基础任务规划：**用户可以创建任务清单，设定截止日期，优先级等。软件提供友好的时间规划辅助工具、跟踪提醒组件。
2. **多团队、多角色场景下的任务派遣和管理：**支持团队协作，允许团队成员互相指派任务、查看任务进度和共享资源；支持团队内的权限管理（普通成员和团队管理者）。同时支持多团队、多重身份应用场景。
3. **完善的权限管理和信息安全保障**支持细粒度多角色的权限分配和控制、支持密文形式存储同步、支持外部临时访客。

## 1.3定义

本文档中使用的术语包括：

1. 用户(普通用户)：任何本系统的使用者都是用户，拥有基本的个人任务管理功能。任何用户均可创建或受邀加入/申请退出团队。团队内的普通用户可以提出协作任务请求。
2. 团队所有者：本系统中团队最高权限的所有者，可以授予/修改/驳回团队内其他成员在本团队的权限，可以解散团队。创建团队的用户默认为团队所有者，也可转让给其他团队成员。
3. 管理员：团队内的普通用户接受团队所有者授予的不同于普通用户的特别权限而成为管理员。特殊权限包括：邀请/移出团队普通成员、邀请/移除访客、受理用户的协作任务请求。
4. 访客：团队外的成员收到群组所有者或管理员的邀请，获得限期内查看部分团队任务列表和接受协作/拒绝协作任务请求的权利。访客成员不能主动提出协作请求。
5. 操作人：指操作任务条目的用户。
6. 参与人：任务条目涉及的协作用户。协作任务的参与人必须在任务时间段内空闲。
7. 任务计划：至少具有任务名称的数据条目，可以包括时间、参与人、提醒方式、反馈方式、执行周期、其他备注。
8. 协作任务与非协作任务：协作任务指多人共同参与，在所有参与人的时间表上都占用时间段。非协作用户为单人任务。
9. 任务的分派：订立好的任务被分派到参与人的设备上，建立提醒。

## 1.4参考资料

[1]毛新军，董威. 软件工程：从理论到实践.北京：高等教育出版社.2022.

# 2软件设计约束

## 2.1设计目标和原则

**2.1.1 设计目标**

**本系统设计目标的定性描述如下，具体需求定义参见《需求规格说明书》。**

**1. 功能实现：系统实现下述功能。**

**a. 基础任务规划：**用户可以创建任务清单，设定截止日期，优先级等。软件提供友好的时间规划辅助工具、跟踪提醒组件。

**b. 多团队、多角色场景下的任务派遣和管理：**支持团队协作，允许团队成员互相指派任务、查看任务进度和共享资源；支持团队内的权限管理（普通成员和团队管理者）。同时支持多团队、多重身份应用场景。

**c. 完善的权限管理和信息安全保障**支持细粒度多角色的权限分配和控制、支持密文形式存储同步、支持外部临时访客。

**2. 性能和稳定性要求：**

**a. 基础性能要求：确保应用体验快速流畅，可处理大量数据和多个同时操作(~100个)，满足用户和团队的需求，尤其在繁忙的工作时段。响应时间尽可能短，确保用户在进行任务添加、编辑、查看等操作时感知到的延迟在可接受范围内（<1s）。**

**b. 可靠性：系统具备高度的可靠性和稳定性，确保用户数据不丢失，系统不会频繁崩溃或出现不可预测的错误。在异常情况下，系统能够自动进行故障恢复，并提供恢复用户数据的机制，需要建立可靠的数据持久措施。**

**c. 可扩展性：系统设计应当使用前后端分离的架构体系具备灵活的架构设计，允许添加新的模块或功能、便于整合进团队现有网络设施，而不会影响现有功能的稳定性和性能。**

**3. 用户体验要求：**

**用户界面设计符合现代设计原则，简洁、直观、易用。提供清晰的操作指引和帮助文档，以便用户快速上手并充分利用各项功能。**

**4. 安全性要求：**

**采取严格的安全措施，保护用户数据的机密性和完整性。用户身份验证采用安全可靠的机制，防止未经授权的访问。**

**2.1.2 基本设计原则**

**本设计遵循以下基本设计原则：**

**1. 模块化：系统模块化设计，各模块之间解耦合，方便维护和扩展。**

**2. 一致性：保持系统内部和外部接口的一致性，提高系统的易用性和可理解性。**

**3. 简洁性：避免过度设计和复杂性，保持系统设计简洁明了。**

**4. 可重用性：重用已有的模块和组件，提高开发效率和代码质量。**

**5. 可测试性：设计易于测试的系统，便于进行单元测试和集成测试，确保系统质量。**

**6. 灵活性：系统架构和设计灵活，能够适应未来的需求变化和技术演进。**

## 2.2设计约束

**2.2.1 环境约束**

**软件环境：**

|  |  |  |
| --- | --- | --- |
|  | **名称** | **版本** |
| 服务器端操作系统 | Ubuntu | Ubuntu Server 22.04.4 LTS |
| 数据库平台 | SQLite | 3.x |
| 应用平台 | 跨平台(Web/Windows/Android,其中Windows和Android通过ReactWebApp支持) | Android >=Android Q(Android 10)/Windows 7以上; |
| 客户端软件 | Chrome/Android/Windows | Chrome >=83 |

**硬件环境：**

|  |  |
| --- | --- |
| **服务器** | **推荐配置** |
| 应用服务器（前端） | 4核8GB内存，100GB硬盘，带宽25Mbps以上（推荐的最低硬件配置） |
| 应用服务器（后端） | 4核8GB内存，100GB硬盘，带宽25Mbps以上（推荐的最低硬件配置） |
| 数据库服务器 | 4核8GB内存，100GB硬盘，带宽25Mbps以上（推荐的最低硬件配置） |

备注：应用服务器（前端）、应用服务器（后端）、数据库服务器可以是同一服务器，也可以是不同的物理服务器/容器。上述硬件环境是推荐的最低硬件配置，本系统保证能够在最低硬件配置上正常运行。

**2.2.2 开发工具和语言**

**使用Python语言开发，后端使用Django框架，前端使用React框架实现。**

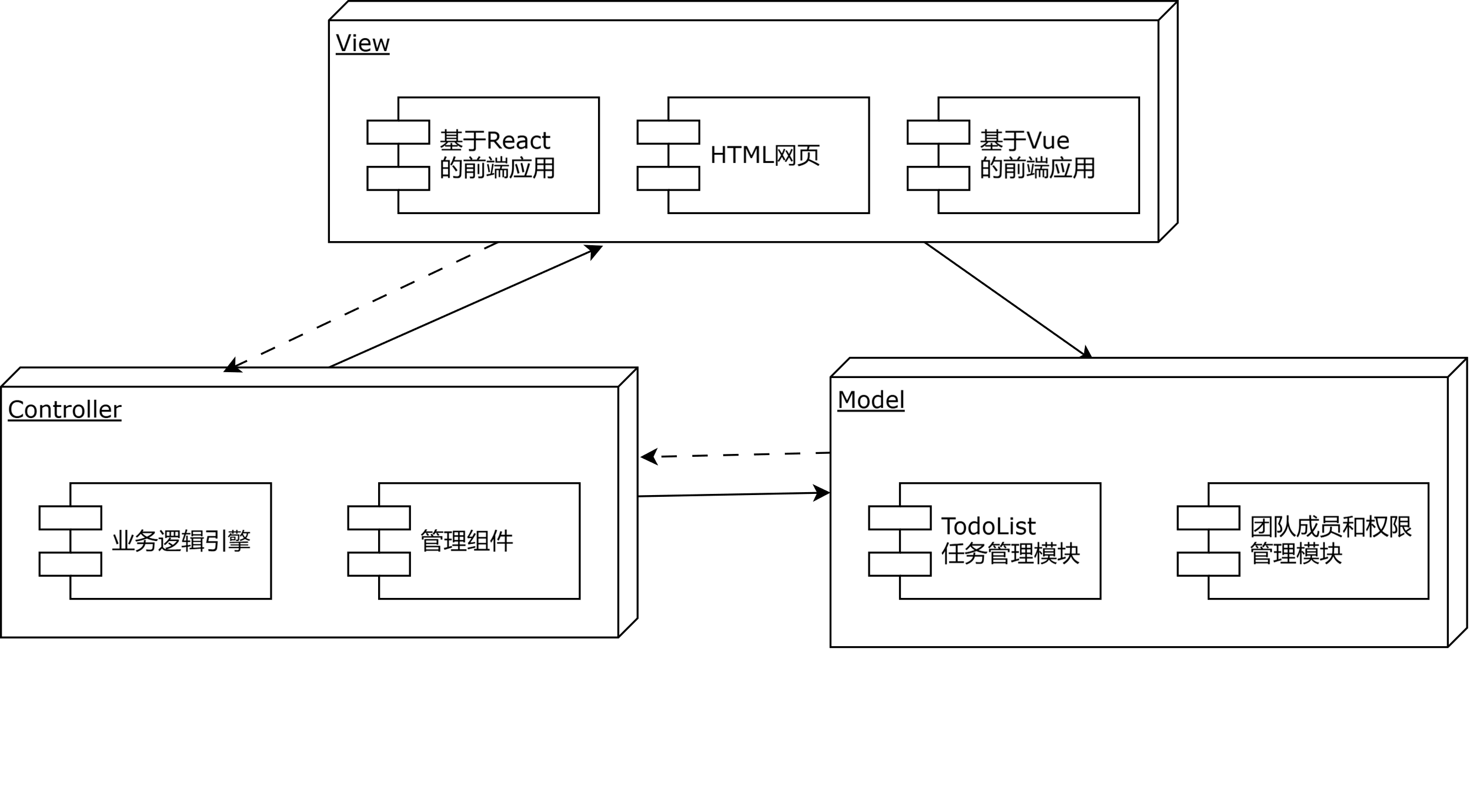
**2.2.3 设计目标约束**

**要求实现《需求规格说明书》中定义的性能和可靠性约束、可扩展性要求、用户体验要求、安全性要求和数据安全要求。具体要求参见《需求规格说明书》3.2节“非功能性需求”。**

# 3软件设计

## 3.1体系结构设计

**软件系统的体系结构设计基于MVC风格，具体包括以下组件：**



* **前端（View）：**

**前端负责用户界面的展示和用户交互。**

**用户通过前端界面与系统进行交互，包括任务添加、编辑、查看等操作。**

**前端组件通过与控制器的通信，更新界面显示并接收用户输入。**

**前端使用React、JavaScript和HTML技术实现。**

* **后端控制器【业务逻辑引擎和管理组件】（Controller）：**

**后端控制器负责接收来自前端的请求，并根据请求调用相应的业务逻辑处理。控制器负责协调前端界面和模型之间的交互，处理用户请求并更新模型状态。控制器将处理结果返回给前端，实现前后端之间的数据传输和交互。**

* **任务管理模块（Model）：**

**任务管理组件包含了与任务相关的业务逻辑和数据操作。**

**它负责管理任务的创建、编辑、删除等操作，以及与任务相关的数据持久化和状态管理。**

**任务管理组件通过接口与控制器进行交互，提供对任务数据的访问和操作功能。**

* **团队管理组件【团队成员管理和权限管理模块】（Model）：**

**团队管理组件包含了与团队协作相关的业务逻辑和数据操作。**

**它负责管理团队成员的邀请、加入、退出等操作，以及团队的任务分配和协作管理。**

**团队管理组件也通过接口与控制器进行交互，提供对团队数据的访问和操作功能。**

**系统的后端基于Django框架实现。**

**上述组件之间的交互关系如下：前端（View）通过向控制器（Controller）发送请求来触发操作，控制器接收请求后调用相应的任务管理组件和团队管理组件提供的接口执行业务逻辑操作，并将处理结果返回给前端。控制器与模型（Model）之间的交互实现了数据传递和业务逻辑处理，同时模型之间通过定义好的接口或事件机制进行交互，确保各个模块之间的解耦合和灵活性。**

## 3.2用户界面设计

**用户界面设计的原型，以及用户界面的设计类图以及页面之间的跳转关系顺序图，并提供必要的文字补充说明。**

**注意：用户界面设计模型和体系结构模型要一致**

## 3.3用例设计

**用例的设计模型，包括描述用例实现的顺序图、用例实现的设计类图等，并提供必要的文字补充说明。**

## 3.4类设计

**给出各个类的实现模型，包括详细描述各个类的可见范围、类的属性和方法，给出精化后的类图，描述类方法的活动图，类对象的状态图等，并提供必要的文字补充说明。**

## 3.5数据设计

**给出软件系统中永久数据的设计模型，包括描述数据库及表的设计类图，描述数据操作的活动图、必须提供必要的文字补充说明。**

## 3.6部署设计

**给出软件系统的部署方式（部署图），必须提供必要的文字补充说明。**

# 4 需求跟踪矩阵

**软件设计与软件需求定义中用户需求、软件规格说明中的软件功能的对应关系说明。**