东莞理工学院实训项目（2024年）RustChatGPT项目

需求规格说明书

学 院： 计算机科学与技术学院

班 级： 软件工程2班

日 期： 2024.06.25

**修订历史记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 日期 | AMD | 修订者 | 说明 |
| V1.0 | 2024.06.25 | A | 詹晓芹 | 新增需求规格说明书 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

（A-添加，M-修改，D-删除）

目录

1. 引言………………………………………………………………………………4

1.1 项目背景………………………………………………………………………4  
1.2 项目目标………………………………………………………………………4  
1.3 项目范围………………………………………………………………………4

1.3.1 系统功能…………………………………………………………………4

1.3.2 用户界面…………………………………………………………………5

1.4角色功能描述…………………………………………………………………5  
1.5 主要业务流程…………………………………………………………………5

2. 功能性需求………………………………………………………………………6

2.1 用户界面 ……………………………………………………………………7

2.1.1 主页………………………………………………………………………7

2.1.2 登录注册界面……………………………………………………………7  
2.1.3 聊天界面…………………………………………………………………8

2.2 提示词接口……………………………………………………………………9

2.2.1 提示词接口描述…………………………………………………………9  
2.2.2 API设计…………………………………………………………………10

2.3 OpenAI API集成………………………………………………………………12

2.3.1 OpenAI API密钥配置……………………………………………………13  
2.3.2 API调用流程……………………………………………………………13

3. 非功能需求……………………………………………………………………14

3.1 性能需求…………………………………………………………………14  
3.2 安全需求…………………………………………………………………14  
3.3 可用性需求………………………………………………………………15

4. 技术栈………………………………………………………………………16

4.1 前端技术栈………………………………………………………………16  
4.2 后端技术栈………………………………………………………………16  
4.3 第三方服务………………………………………………………………17

5. 系统架构……………………………………………………………………18

5.1 前端架构…………………………………………………………………18

5.2 后端架构……………………………………………………………………19

5.3 系统部署架构………………………………………………………………20

6. 测试………………………………………………………………………………20

6.1 单元测试……………………………………………………………………20

6.2 系统测试……………………………………………………………………21

6.3 集成测试……………………………………………………………………21

7. 验收标准………………………………………………………………………22

7.1 功能验收……………………………………………………………………22  
7.2 性能验收……………………………………………………………………22  
7.3 安全验收……………………………………………………………………23

**1. 引言**

**1.1 项目背景**

近年来，随着人工智能和自然语言处理技术的快速发展，基于对话的AI应用越来越受到用户的欢迎。OpenAI的ChatGPT是其中的佼佼者，它能够生成高质量的自然语言文本并进行有效的人机交互。本项目旨在开发一个仿ChatGPT的Web应用程序，以提供类似的用户体验，并通过一个提示词接口供其他Python代码调用，实现与ChatGPT的集成。

**1.2 项目目标**

本项目的主要目标是开发一个Web应用程序，模仿ChatGPT的页面设计，使其能够集成OpenAI API以实现ChatGPT的核心功能。并且提供一个RESTful API接口，使其他Python代码能够传递提示词给ChatGPT，并接收其响应。

**1.3 项目范围**

**1.3.1 系统功能**

用户界面：提供一个仿ChatGPT的Web聊天界面，用户可以通过该界面输入提示词，并查看ChatGPT生成的响应。显示历史对话记录，便于用户查看之前的交互内容。

提示词接口：提供一个RESTful API接口，允许其他Python代码通过该接口传递提示词参数，系统会将该提示词传递给ChatGPT，并返回生成的响应。该接口的设计使其可以被外部应用程序集成，以实现与ChatGPT的交互。

OpenAI API集成：系统集成了OpenAI的API，用于处理用户输入的提示词，并生成相应的回复。通过调用OpenAI的Completion API，系统能够快速生成高质量的自然语言响应。

**1.3.2用户界面**

聊天窗口：显示用户和ChatGPT之间的对话，按照时间顺序排列；用户的输入和ChatGPT的响应以不同的样式展示，便于区分。

输入框：用户在输入框中输入提示词或问题；输入框位于页面底部，方便用户快速输入。

提交按钮：用户点击提交按钮发送提示词；提交按钮位于输入框右侧，确保操作简便。

历史记录：聊天窗口支持滚动查看历史对话，用户可以方便地查看之前的交互内容；每次新的对话会自动添加到历史记录中。

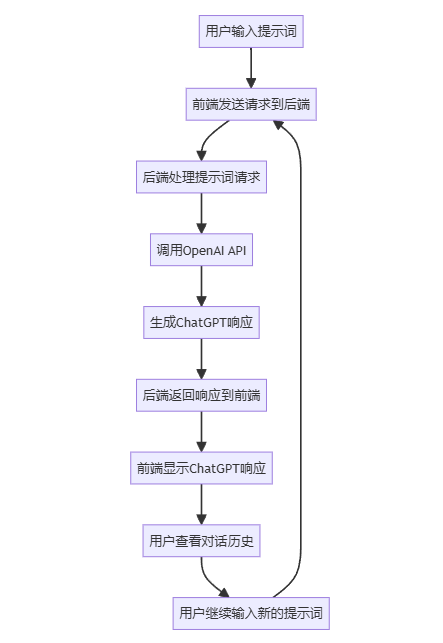
**1.4 角色功能描述**

用户：通过Web界面与ChatGPT进行交互，输入提示词，查看ChatGPT的回复，查看历史对话记录。

开发者：通过提示词接口与ChatGPT集成，调用API传递提示词，接收ChatGPT的响应，用于不同的应用场景，维护和更新系统，确保其稳定性和安全性。

系统管理员：系统管理员在项目中扮演着关键的角色，负责确保系统的稳定性、安全性和性能优化，同时支持团队成员的操作和开发工作，保证系统能够长期有效地运行和服务用户。

**1.5 主要业务流程**



流程描述：

本系统的主要流程从用户输入提示词开始。用户在Web界面的输入框中输入提示词或问题，并点击提交按钮。提交按钮会触发一个AJAX请求，将用户输入的提示词发送到后端API接口 /prompt。接着，后端接收到请求后，会调用OpenAI的Completion API，将提示词发送给ChatGPT。ChatGPT接收到提示词后，会生成相应的响应文本。后端处理完来自ChatGPT的响应后，将其打包并返回给前端。

前端接收到后端返回的响应后，会将ChatGPT的回复显示在聊天窗口中。用户可以在聊天窗口中查看整个对话内容，包括历史记录和最新回复。用户可以继续输入新的提示词，重复上述流程，与ChatGPT进行多轮对话。每轮对话的内容都会保留在聊天窗口中，形成完整的对话记录。这种持续交互的设计，确保了用户体验的连续性和对话的连贯性。

**2. 功能需求**

**2.1 用户界面**

**2.1.1 主页**

**2.1.1.1需求描述：**

主页是用户进入系统的第一个界面，旨在提供应用程序的概述和入口。

**2.1.1.2页面设计：**

**2.1.1.3角色说明：**

|  |  |  |
| --- | --- | --- |
| 用户名称 | 权限 | 职责 |
| 用户 | 访问主页，点击登录或注册按钮进入登录页面，直接在主页进行聊天 | / |

**2.1.1.4逻辑说明：**

用户打开应用程序时，自动显示主页，可直接聊天或者当用户点击登录或注册，跳转登录页面，登录成功开始聊天。

**2.1.1.5数据项描述：**

无需持久化数据，主页仅展示静态信息和用户操作按钮。

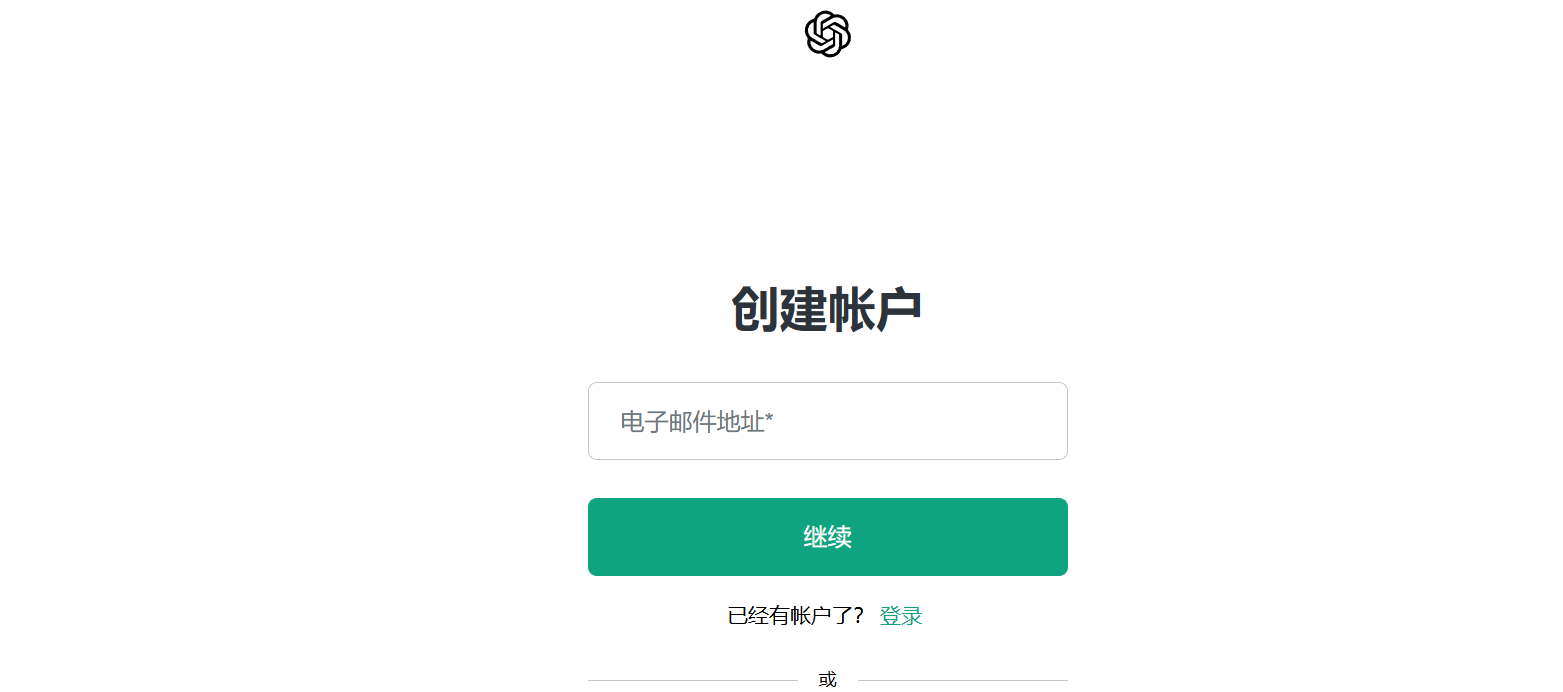
**2.1.2 登录注册界面**

**2.1.2.1需求描述**

用户登录注册，可以获得更多智能的回复，体验更多功能

**2.1.2.2页面设计**

仿chatGPT登录注册页面，用户输入信息点击登录或注册按钮



**2.1.2.3 角色说明**

|  |  |  |
| --- | --- | --- |
| 用户名称 | 权限 | 职责 |
| 用户 | 在登录框输入信息登录或注册 | / |

**2.1.2.4 逻辑说明**

用户输入账号信息，后端检测该账号信息是否已经注册，如果已经注册，检测密码是否正确，密码正确进入聊天页面，密码失败重定向到登录页面。账号不存在则提醒注册账号或者检查账号信息是否正确

**2.1.2.5 数据项描述**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据名称 | 填写方式 | 数据定义 | 必填 | 数据项 |
| id | 自动填入 | Varchar | / | / |
| 用户名 | 自动录入 | Varchar | / | / |
| 密码 | 自动录入 | Varchar | / | / |

**2.1.3 聊天界面**

**2.1.3.1需求描述：**

聊天界面是用户与ChatGPT进行交互的主要界面，支持实时的文本输入和响应展示。

**2.1.3.2页面设计：**

**2.1.3.3角色说明：**

|  |  |  |
| --- | --- | --- |
| 用户名称 | 权限 | 职责 |
| 用户 | 在输入框中输入提示词或问题，查看ChatGPT的响应。 | / |

**2.1.3.4逻辑说明：**

用户在输入框中输入文本后，点击提交按钮，前端发送请求到后端处理。

后端接收到请求后，调用OpenAI API生成响应，返回给前端。

前端将生成的响应显示在聊天窗口中，用户可以实时看到ChatGPT的回复。

**2.1.3.5 数据项描述：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据名称 | 填写方式 | 数据定义 | 必填 | 数据项 |
| 提示词 | 自动录入 | Text | / | / |
| ChatGPT生成的响应 | 自动录入 | Text | / | / |

**2.2 提示词接口**

提示词接口是仿ChatGPT应用程序的核心组件之一，旨在提供其他应用程序或Python代码与ChatGPT模型交互的能力。通过该接口，外部系统可以传递提示词，接收由ChatGPT生成的文本响应，实现跨系统的对话功能集成。

**2.2.1 提示词接口描述**

提示词接口的设计目标是实现与外部系统的无缝集成，使它们能够利用ChatGPT的强大对话能力。这一接口基于RESTful架构风格，确保简单、灵活和易于使用。通过POST请求，外部系统可以发送包含提示词的JSON数据到指定的URL，系统将提示词传递给OpenAI的ChatGPT模型进行处理，并将生成的文本响应返回给调用方。

功能描述：

* 外部集成能力： 提供开放的API接口，使得其他应用程序或Python代码可以方便地调用ChatGPT生成对话文本。这种设计允许广泛的应用场景，如集成到客服系统、智能助手、内容生成工具等。
* 简洁的请求和响应格式： 请求体采用标准的JSON格式，包含一个必要的字段prompt，代表用户输入的提示词。响应体也采用JSON格式，包含ChatGPT生成的回复文本，确保数据传输的简洁和易解析性。
* 错误处理机制： 设计完善的错误处理机制，确保在发生错误时返回适当的HTTP状态码和错误信息，使调用方能够识别并处理异常情况，如无效请求、服务器错误等。

**2.2.2 API设计**

URL: /prompt

方法: POST

请求头: Content-Type: application/json

请求体:

Json：{

"prompt": "Your input prompt here"

}

请求体包含一个字段prompt，该字段为用户输入的提示词，类型为字符串。提示词是ChatGPT生成文本的基础，系统将根据提示词生成相应的文本响应。

响应体:{

"response": "ChatGPT generated response here"

}

响应体包含一个字段response，该字段为ChatGPT生成的文本响应，类型为字符串。系统将ChatGPT生成的文本返回给调用方，以便其进行进一步的处理或展示。

错误处理:

* 400 Bad Request: 当请求体缺少prompt字段或prompt字段为空时，返回此状态码。响应体将包含详细的错误信息，说明请求不符合API规范。
* 401 Unauthorized: 当请求未经授权或认证失败时，返回此状态码。响应体将包含错误信息，提示调用方进行适当的身份验证。
* 500 Internal Server Error: 当服务器内部发生错误或无法处理请求时，返回此状态码。响应体将包含通用的错误信息，提示调用方稍后重试或联系支持团队。

示例请求和响应:

* 请求示例:

POST /prompt HTTP/1.1

Host: api.example.com

Content-Type: application/json

{

"prompt": "Tell me a joke"

}

* 成功响应示例:

HTTP/1.1 200 OK

Content-Type: application/json

{

"response": "Why don't scientists trust atoms? Because they make up everything!"

}

* 错误响应示例:

HTTP/1.1 400 Bad Request

Content-Type: application/json

{

"error": "The 'prompt' field is required."

}

实现细节:

1. 接收请求：后端通过FastAPI框架接收来自前端或其他应用程序的提示词请求。使用POST方法接收请求体中的JSON数据，并验证请求格式是否正确。
2. 构建请求：在确认请求合法后，后端将接收到的提示词构建为OpenAI API的请求参数，包括必要的API密钥和其他配置。
3. 发送请求：后端向OpenAI API发送HTTP请求，包含构建好的请求参数和认证信息，确保请求的安全性和有效性。
4. 处理响应：接收OpenAI API返回的响应数据，提取生成的文本内容，并进行必要的格式化或处理。
5. 返回结果：将生成的文本作为响应返回给前端或调用方，确保响应格式符合预期，同时在响应头中设置适当的HTTP状态码。

**2.3 OpenAI API集成**

**2.3.1 OpenAI API密钥配置**

**2.3.1.1需求描述：**

OpenAI API密钥用于认证和授权系统访问OpenAI服务，确保只有授权的应用程序可以调用ChatGPT模型。

**2.3.1.2密钥存储：**

将API密钥保存在环境变量中，例如OPENAI\_API\_KEY。

**2.3.1.3配置步骤：**

开发者从OpenAI获取API密钥，然后将密钥添加到系统的环境变量中，确保安全地存储和使用。

**2.3.1.4角色说明：**

|  |  |  |
| --- | --- | --- |
| 用户名称 | 权限 | 职责 |
| 系统管理员 | 负责配置和管理系统的环境变量，包括API密钥的存储。 | / |

**2.3.1.5逻辑说明：**

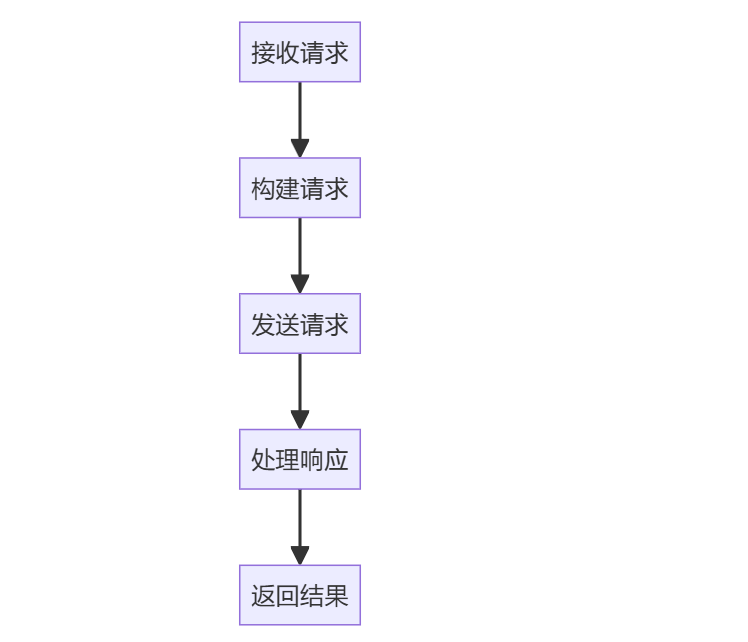
系统在启动时从环境变量中读取OpenAI API密钥。后端在处理提示词请求时，使用该密钥进行OpenAI API的认证和授权。

**2.3.2 API调用流程**

**2.3.2.1流程描述：**

API调用流程描述了后端如何使用OpenAI API处理用户输入的提示词，并生成ChatGPT的响应。

**2.3.2.2流程步骤：**



**2.3.2.3角色说明：**

|  |  |  |
| --- | --- | --- |
| 用户名称 | 权限 | 职责 |
| 开发者 | 使用系统提供的API接口与OpenAI API进行集成，获取ChatGPT生成的文本响应。 | / |

**2.3.2.4逻辑说明：**

后端接收到用户输入的提示词后，构建合适的HTTP请求，将提示词传递给OpenAI API。

OpenAI API生成响应后，后端处理并将结果返回给前端或调用方。

**3 非功能需求**

**3.1 性能需求**

**3.1.1性能需求描述：**

在设计和开发Web应用程序时，性能需求是确保系统能够在各种负载情况下快速响应用户请求和处理数据的关键因素之一。性能需求涉及系统的响应时间、并发处理能力以及资源利用效率。

**3.1.2实时响应**

对于ChatGPT模拟器这样的应用程序，实时响应是关键之一。用户期望能够快速地获取ChatGPT生成的响应，以保持对话的流畅性和真实感。因此，系统设计需要确保在用户提交提示词后，能够在数秒内生成并返回响应。这需要考虑到后端处理逻辑的优化，例如异步处理和缓存机制的使用，以最大限度地减少响应延迟。

**3.1.3并发处理**

另一个重要的性能考量是系统的并发处理能力。系统应设计为能够同时处理多个用户的请求，特别是在高流量或繁忙时段。通过使用适当的并发处理技术（如线程池或异步任务队列），系统能够有效地管理和调度请求，确保每个请求都能及时得到处理而不影响其他用户的体验。

**3.1.4资源利用**

有效利用服务器资源对系统性能至关重要。系统应设计为在处理请求时，能够有效地利用服务器的内存、CPU和存储资源，以提高系统的处理能力和响应速度。这包括优化算法和数据结构的选择，减少不必要的资源消耗，并且定期进行性能测试和调优，以确保系统在各种负载情况下仍能保持稳定和高效。

**3.2 安全需求**

**3.2.1安全需求描述**

在构建Web应用程序时，安全性是至关重要的考量因素。安全需求涵盖了数据传输的加密、身份验证、漏洞管理和数据保护，以确保系统和用户数据的安全性和完整性。

**3.2.2数据加密**

为了保护用户输入的提示词和系统返回的ChatGPT响应，在数据传输过程中应使用HTTPS协议进行加密。HTTPS协议通过SSL/TLS加密机制，确保数据在传输过程中不被恶意第三方截取或篡改，保证用户数据的机密性和完整性。

**3.2.3身份验证**

系统应实施严格的身份验证机制，以确保只有经过授权的用户或应用程序可以访问系统功能和API。这可以通过使用令牌验证、OAuth认证等现代身份验证技术来实现，有效防止未经授权的访问和潜在的安全威胁。

**3.2.4漏洞管理**

定期进行安全漏洞扫描和评估是确保系统安全的重要步骤。系统管理员应及时更新和修补发现的安全漏洞，以防止黑客利用漏洞进行攻击或数据泄露。

**3.2.5合规性和法律要求**

系统应符合适用的法律法规和组织内部的安全政策要求。这包括数据保护法规（如GDPR）、个人隐私保护法律以及其他相关的法律要求，确保系统的运行和数据处理都合法合规。

**3.3 可用性需求**

**3.3.1可用性需求描述**

系统的可用性是衡量系统可靠性和用户满意度的重要指标。可用性需求包括系统的可靠性、备份与恢复能力、监控与警报机制，以确保系统能够随时提供服务并及时响应用户需求。

**3.3.2可靠性**

系统应设计为具有高可靠性，即能够持续运行并处理用户请求，减少由于系统故障或服务中断而导致的停机时间。通过实施容错机制、负载均衡和故障恢复策略，确保系统在面对异常情况时仍能保持稳定运行。

**3.3.3备份与恢复**

为了应对数据丢失或系统故障的风险，系统应实施有效的数据备份和恢复策略。定期备份系统数据，并确保备份数据存储在安全可靠的位置，以便在需要时能够快速恢复系统和用户数据。

**3.3.4监控与警报**

系统应设置监控系统，实时监测系统的运行状态、性能指标和用户活动。通过实时警报和通知机制，及时发现和响应系统异常、性能下降或安全事件，以保证及时的故障排除和问题解决。

**4 技术栈**

**4.1 前端技术栈**

**4.1.1前端技术栈描述：**

前端技术栈是指用于开发和构建Web应用程序用户界面的技术和工具集合。在开发ChatGPT模拟器的前端界面时，需要选择和使用适合的前端技术栈，以实现良好的用户体验和界面交互。

**4.1.2主要技术选型：**

* 框架：React.js
  + React.js是一个流行的JavaScript库，用于构建用户界面。它提供了组件化开发的能力，使得界面的复杂性得以管理和扩展。
* 状态管理：Redux
  + Redux是用于管理应用程序状态的JavaScript库。在ChatGPT模拟器中，可以使用Redux来管理用户输入、系统状态以及与后端的通信状态。
* 样式处理：CSS Modules 或 Styled Components
  + CSS Modules或Styled Components可以帮助管理界面的样式，使得样式与组件关联，并提供模块化和可重用的样式定义。
* 构建工具：Webpack
  + Webpack是一个现代的JavaScript应用程序静态模块打包工具。它可以将多个模块打包成一个或多个文件，以优化加载时间。
* 其他工具和库：
  + Axios：用于HTTP请求。
  + Jest和Enzyme：用于前端单元测试。
  + ESLint和Prettier：用于代码规范和格式化。

**4.2 后端技术栈**

**4.2.1后端技术栈描述：**

后端技术栈是指用于实现服务器端应用程序逻辑和数据处理的技术和工具。在开发ChatGPT模拟器的后端服务时，需要选择和使用适合的后端技术栈，以支持接收用户输入、调用OpenAI API并返回响应等功能。

**4.2.2主要技术选型：**

* 框架：FastAPI
  + FastAPI是一个现代的Python Web框架，基于标准的Python类型提示来实现高性能的API开发。
* 异步处理：AsyncIO
  + AsyncIO是Python的异步I/O框架，能够支持异步请求处理，提高系统的并发能力和响应速度。
* OpenAI API调用：Requests 或 HTTPX
  + 使用Requests或HTTPX库来发起HTTP请求，调用OpenAI API生成ChatGPT的响应。
* 数据存储：SQLite（轻量级数据库）
  + 对于简单的数据存储需求，可以选择SQLite作为后端数据库，用于存储用户的对话历史或其他必要的信息。
* 安全：OAuth2
  + 使用OAuth2协议来实现用户身份验证和授权，保护API端点不被未授权的访问所利用。
* 部署：Docker 和 Docker Compose
  + 使用Docker容器化应用程序，并通过Docker Compose来管理多个容器的部署和运行。

**4.3 第三方服务**

**4.3.1第三方服务描述：**

在开发和部署ChatGPT模拟器时，可能会依赖一些第三方服务来增强系统的功能或提供必要的基础设施支持。

**4.3.2主要第三方服务选型：**

* OpenAI API：OpenAI API用于调用GPT模型，生成对用户输入的响应文本。
* GitHub Actions 或 Jenkins：GitHub Actions或Jenkins用于自动化CI/CD流程，例如自动运行测试、构建和部署应用程序。
* Sentry：Sentry用于应用程序的错误和异常监控，帮助实时捕获和诊断生产环境中的问题。
* AWS S3：AWS S3用于存储静态文件和备份数据，提供高可用性和安全的对象存储服务。
* Google Analytics：Google Analytics用于监控和分析用户访问和行为数据，帮助优化用户体验和营销策略

**5 系统架构**

**5.1 前端架构**

前端架构的设计旨在实现用户友好的界面，能够与后端快速交互，并展示ChatGPT生成的响应文本。

**5.1.1前端技术选型和架构设计：**

**技术选型：**

React.js：作为主要的前端框架，支持组件化开发和虚拟DOM技术，提高页面渲染效率和用户交互体验。

Redux：用于管理前端应用的状态，包括用户输入的提示词和系统状态的管理。

Axios：处理前端与后端的HTTP请求和响应，实现与后端接口的数据交换。

CSS Modules 或 Styled Components：管理界面的样式，确保样式模块化、可重用，并避免全局样式污染。

**架构设计：**

组件化结构：将界面拆分为多个独立的组件，每个组件专注于特定的功能或视图。例如，输入提示词的表单组件、显示ChatGPT生成响应的文本区域组件等。

状态管理：使用Redux管理全局状态，包括用户输入状态、系统加载状态和错误状态。通过Redux的单一数据源原则，统一管理应用状态，实现数据的一致性和可预测性。

异步处理：利用Redux中间件处理异步操作，如发送提示词到后端并等待响应。这确保了应用程序的流畅性和性能优化，避免阻塞主线程。

通信和数据流：

用户输入处理：用户输入提示词后，前端将数据通过Redux状态管理传递给后端接口。

后端响应处理：接收后端返回的ChatGPT生成的文本响应，更新前端界面展示给用户。

**5.2 后端架构**

后端架构设计旨在支持快速、安全地处理用户请求，并与OpenAI API集成以生成文本响应。

**5.2.1后端技术选型和架构设计：**

**技术选型：**

FASTAPI：作为Python的现代Web框架，支持异步处理和快速API开发。它提供了基于标准Python类型提示的自动文档生成功能，便于API的维护和扩展。

Python：作为主要编程语言，用于业务逻辑的处理和与OpenAI API的集成。

AsyncIO：支持异步请求处理，提高系统的并发能力和响应速度。

Requests 或 HTTPX：用于向OpenAI API发送HTTP请求，并处理返回的生成文本响应。

**架构设计：**

RESTful API：采用RESTful设计风格，定义清晰的API端点，包括接收提示词请求和返回ChatGPT生成响应的API接口。

模块化和分层设计：将应用程序分为不同的模块或服务，例如处理提示词请求、调用OpenAI API生成文本响应等。这样的设计有助于降低耦合度，提高代码的可维护性和可扩展性。

安全性考量：实施OAuth2认证机制或API密钥验证，确保只有经过授权的应用程序可以访问API端点和服务。

**数据流和处理：**

接收和处理请求：后端接收来自前端的提示词请求，验证和解析请求参数。

与OpenAI API集成：构建OpenAI API的请求参数，并通过HTTP请求发送给OpenAI服务端。处理返回的生成文本响应，格式化后返回给前端。

异常处理和日志记录：捕获和处理异常情况，记录日志以便进行故障排除和性能优化。

**5.3 系统部署架构**

系统部署架构设计考虑到应用程序的可扩展性、高可用性和安全性，使用容器化技术简化部署和管理。

**5.3.1系统部署技术选型和架构设计：**

**技术选型：**

Docker 和 Docker Compose：将前端和后端应用程序容器化，统一开发、测试和生产环境的部署。

Nginx：作为反向代理服务器，处理前端静态文件的服务和负载均衡。

AWS ECS 或 Kubernetes：用于管理和编排容器集群，提供高可用性和自动扩展能力。

CI/CD工具（如GitHub Actions 或 Jenkins）：自动化构建、测试和部署流程，确保代码的持续集成和交付。

**架构设计：**

容器化部署：每个服务（前端和后端）打包为一个独立的Docker容器，包含应用程序和运行时依赖。

集群管理：使用AWS ECS或Kubernetes管理和调度容器，根据负载自动扩展容器实例，确保系统在高流量时保持稳定性。

安全配置：配置网络安全组和访问控制策略，限制对容器和服务的访问，并实施HTTPS传输加密。

**监控和日志：**

集成监控系统：使用AWS CloudWatch或Prometheus等监控工具，实时监测应用程序的性能指标和运行状态。

日志管理：集中管理应用程序的日志，便于故障排除和系统优化。

**6 测试**

**6.1 单元测试**

**前端单元测试：**

使用测试框架如Jest或React Testing Library对React组件进行测试。

测试用户输入提示词的表单组件，确保输入验证和状态管理的正确性。

测试展示ChatGPT生成响应的文本区域组件，验证文本显示和格式化是否正确。

**后端单元测试：**

使用pytest等测试框架对FASTAPI应用的API端点进行单元测试。

测试提示词接口是否能够正确解析和处理请求参数。

编写测试用例模拟调用OpenAI API，验证生成文本响应的处理和格式化。

**异步代码和边界条件测试：**

测试异步处理代码，确保在并发请求和处理异常时的稳定性。

测试边界条件，如空输入、非法输入等，确认系统能够正确处理异常情况。

**6.2 系统测试**

系统测试验证整个应用程序在集成后的功能和性能表现，以及系统与外部系统的交互。

**前端系统测试：**

模拟用户交互场景，通过界面输入提示词并验证ChatGPT生成的文本响应。

确认界面的响应速度和用户体验是否符合预期。

**后端系统测试：**

执行端到端的API测试，包括发送提示词请求和接收生成文本响应。

确保后端服务能够正常响应和处理预期的请求量和负载。

**性能测试：**

使用工具如Apache JMeter或Locust进行性能测试，评估系统在不同负载下的响应时间和吞吐量。确保系统在高并发情况下依然能够稳定运行，并且资源利用率合理。

**6.3 集成测试**

集成测试验证不同组件、模块或服务之间的集成是否正确，确保整体系统的功能和数据流的正确性。

**前后端集成测试：**

测试前端与后端之间的数据传输和交互是否顺畅和正确。

确认前端发送的提示词能够正确被后端接收和处理，后端能够正确调用OpenAI API生成响应。

**外部服务集成测试：**

集成测试OpenAI API的调用，确保请求参数和响应数据的正确格式和处理。

模拟不同的生成文本请求，验证系统能够正确解析和展示生成的文本响应。

**数据一致性和错误处理：**

测试数据在系统内部流动的一致性，确保从输入到输出的数据处理流程正确无误。验证系统在异常情况下的错误处理机制，如网络超时、服务不可用等情况下的表现。

**7 验收标准**

**7.1 功能验收**

功能验收确保仿ChatGPT应用程序按照需求规格书中定义的功能正常工作，并且用户能够顺利使用系统的各项功能。

具体的功能验收标准包括但不限于：

* **用户界面验证**：
  + 确认用户能够输入提示词并发送请求。
  + 验证ChatGPT生成的文本响应能够正确显示和格式化。
  + 确保界面交互流畅，用户操作符合预期反馈。
* **提示词接口**：
  + 测试提示词接口能够接收并解析来自前端或其他应用程序的请求参数。
  + 确认接口能够有效地将参数传递给ChatGPT，并接收生成的文本响应。
* **OpenAI API集成**：
  + 验证系统能够正确配置和调用OpenAI API，包括认证和请求参数的正确构建。
  + 确认从OpenAI API接收的生成文本响应与预期的格式和内容一致。
* **异常处理**：
  + 测试系统在异常情况下的表现，如网络超时、服务不可用等，确保能够适当地返回错误信息或进行恢复。

**7.2 性能验收**

性能验收评估ChatGPT模拟器应用程序在各种负载条件下的性能表现，包括响应时间、并发处理能力和系统资源利用率。

具体的性能验收标准包括但不限于：

* **响应时间**：
  + 测试系统对用户请求的响应时间，确保在正常负载下能够在合理的时间内返回结果。
  + 确认生成文本的响应时间不超过预定的阈值。
* **并发能力**：
  + 使用负载测试工具对系统进行压力测试，评估在高并发情况下系统的稳定性和吞吐量。
  + 确保系统能够在高负载下维持良好的性能表现，不出现性能急剧下降或宕机的情况。
* **资源利用率**：
  + 监控和评估系统在运行时的内存和CPU使用情况，确保资源利用率在合理范围内。
  + 确保系统在长时间运行和持续高负载下能够有效管理和优化资源使用。

**7.3 安全验收**

安全验收确保ChatGPT模拟器应用程序在设计和实现上符合安全标准，保护用户数据和系统免受恶意攻击和数据泄露的风险。

具体的安全验收标准包括但不限于：

* **数据传输安全**：
  + 确保用户与系统之间的数据传输采用加密协议，如HTTPS，防止数据在传输过程中被窃取或篡改。
* **访问控制**：
  + 实施适当的身份验证和授权机制，确保只有经过认证的用户可以访问系统的敏感功能和数据。
* **数据隐私保护**：
  + 确保用户输入的提示词和ChatGPT生成的文本响应数据得到适当的保护，防止未经授权的访问和泄露。
* **安全漏洞管理**：
  + 定期进行安全漏洞扫描和评估，及时修复发现的安全问题，并建立响应措施应对新的安全威胁。