Assignment 3 of course 42036702 System Analysis and Design at SSE, Tongji University

绿植助手

——一站式绿植养护家园

小组成员

2051973 韩嘉睿、2154296 傅佳恒

2151974 赵明泽、2153051 刘　杰

目 录

一、引言与概述

1.1 文档使用

本文档涵盖了项目第三部分的要求，即提供一个与平台相关的详细设计模型，对架构进行优化和设计模型进行增强。文档描述了为系统分析和设计创建的 UML 模型和其他相关文档，简明扼要地指出了项目规格和设计的变化和添加，包括了更新的用例模型和架构细化部分，展示了平台相关的架构和细化的整体结构，提供了子系统和接口的清单，并详细演示了系统与外部系统之间的接口规范。

1.2 项目简介

在这里是引言与概述内容。本项目的目标是搭建一站式的绿植养护平台，为用户提供科普信息管理、园艺社交、植物医疗和养护提醒等多个功能模块，旨在解决用户在绿植养护过程中遇到的问题和烦恼。

通过丰富的科普知识和实用技巧，用户可以更好地了解和管理自己的绿植，从栽培技巧到环境要求，为养护提供全面指导。

园艺社交功能将架设一个互动平台，让用户与绿植爱好者分享经验、交流心得，通过互相启发和鼓励，共同成长。

植物医疗模块将提供专业的健康诊断和治疗建议，用户可通过上传照片或描述植物状况，获得准确的病虫害识别与应对方案，确保绿植健康成长。

此外，个性化的养护提醒服务将根据用户所拥有的绿植种类和需求，及时通知浇水、施肥、修剪等养护操作，帮助用户建立养护习惯，守护绿植的生命力和美观度。

通过实现以上目标，该软件系统将成为用户在绿植养护过程中的终身伴侣，为用户提供养护知识学习、与绿植爱好者交流、解决难题的全方位支持，并通过个性化的养护提醒让用户能轻松愉快地享受绿植养护的乐趣，同时提升绿植的存活率和美丽程度。

1.3 项目背景

• 绿植不仅可以净化空气、美化环境，而且侍养绿植还能缓解压力、愉悦身心。随着人们生活水平的提高，越来越多的人开始将养花种草当作一种新的消遣。家庭绿植、室内绿植市场规模不断扩大。

• 在当今这个快节奏的时代，相比在线下平台采买和养护绿植，人们会更加青睐线上平台提供的便捷服务。

• 一方面，线上平台选择多，功能全，人们能够获得更加丰富的体验；另一方面，线上平台也更能节省人们的精力和时间。

1.4 项目 UML 模型简介

在本项目的系设计阶段，我们使用了一系列 UML（Unified Modeling Language）模型来支持不同用例的实现和系统功能的描述。完成过程中，我们使用了用例图、活动图、类图及时序图对用例的实现进行了详细阐述。以下是我们为每个用例创建的 UML 模型的详细说明：

用例图：用例图是一种图形化工具，用于展示系统的功能和参与者之间的交互。在第一次作业任务中，需求分析阶段，我们为每个用例绘制了用例图，明确了系统的主要功能和参与者角色。例如，我们在用例图中准确地标识了登录注册、植物医疗、养护提醒、社交圈、科普平台、举报平台、商城和个人信息管理等八个主要用例，并显示了它们之间的关系和依赖。

活动图：活动图用于描述系统中的流程和业务逻辑。在需求分析阶段，我们使用活动图来展示用户在各个用例中的操作步骤，以及系统对用户请求的处理过程。活动图通过图形化的方式展示了系统中的业务流程，帮助我们理解系统的工作流程和各个步骤之间的依赖关系。

类图：类图描述了系统中各个类的属性、方法和它们之间的关系。在系统设计阶段阶段，我们使用类图详细说明了系统中的各个类，包括植物对象、用户对象、消息对象、商店对象等。在类图中，我们明确了每个类的属性和方法，以及它们之间的关联、继承和依赖关系，从而帮助我们理解系统的静态结构。

时序图：序列图用于展示系统中对象之间的交互和消息传递顺序。在系统设

计阶段阶段，我们使用序列图来描述每个用例的执行过程，以强调对象之间的交互和消息传递。通过序列图，我们能够清楚地了解用户与系统的交互方式，并触发相应的操作和响应。

通过这些 UML 模型，我们能够更全面地理解系统的结构和功能，并捕捉到用户的需求。这些模型在项目的系统分析和设计阶段起到了关键作用，帮助我们确保系统的一致性、可靠性和可扩展性。通过清晰地描述系统的结构、功能和交互，我们能够更好地进行系统设计和实现，并为项目的开发奠定良好的基础。

1.5 项目进展

在项目的第三阶段，我们通过合理的设计决策，对系统的架构进行了进一步的优化和完善，同时应用了一些设计原则和模式，以确保系统的可扩展性和可维护性。

在介绍与概述部分，详细描述了我们进行系统分析和设计过程中创建的 UML 模型和其他相关文档。这份介绍文件对项目规范和设计的变化与新增进行了简明扼要的概述，为读者提供了对整个项目的概念和背景理解。

同时，我们更新了部分用例模型，通过进一步细化和优化系统的整体架构，提供了一个与特定平台相关的详细架构设计。我们还提供了一个子系统和接口的清单，以明确各个子系统之间的关系和通信方式。为了更加具体地说明系统与外部系统之间的接口规范，我们展示了一些示例，例如与第三方消息系统和地图服务等外部系统的接口。

设计模型部分包括了五个详细的用例实现示例。我们通过应用各种设计机制和优化的架构，设计了这些用例的具体实现方式。

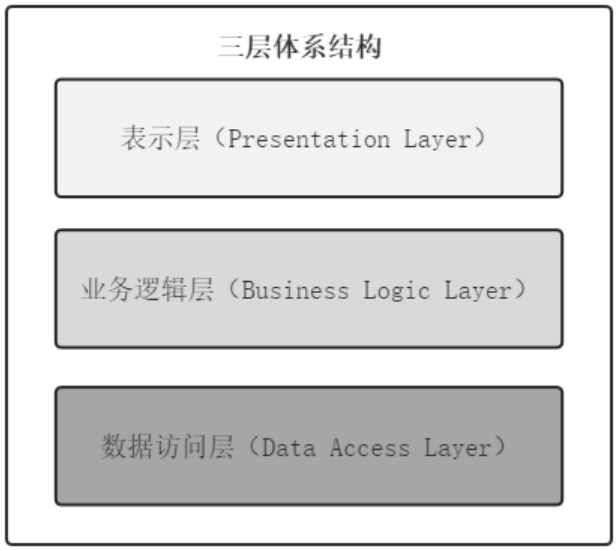
在解决方案中，我们明确了所采用的架构风格和关键设计决策。这些决策是基于项目需求和系统设计的特点而做出的，旨在提高系统的性能、可靠性和可维护性。同时，我们也充分考虑了团队成员的贡献，每个成员在项目中扮演了不同的角色和职责，共同推动了项目的进展和成功。

通过第三阶段的工作，我们取得了显著的进展，项目的设计模型得到了进一步细化和完善，确保了项目的顺利完成并达到预期的目标。

二、架构细化

2.1 平台相关的架构设计

在之前的整体架构设计中，我们考虑到了传统三层分层架构的多种优势，包括其清晰的分层结构，各个层次的职责明确，易于理解与维护等优势，选取了使用这种结构来实现我们的项目。



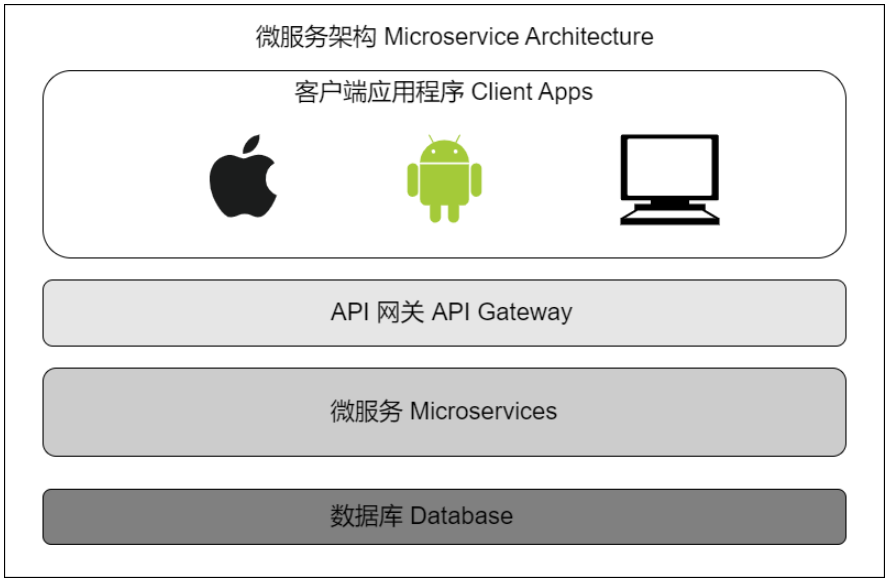
然而，随着对绿植助手项目的深入分析和架构设计的细化，我们意识到传统的三层体系结构存在的性能瓶颈、部署复杂性以及高耦合度等巨大的缺陷可能对我们项目开发造成不利的影响，决定将架构设计从传统的三层分层架构转变为微服务架构。这个决策是基于对两种架构的优缺点进行的全面的评估和权衡，决定采用微服务架构的主要原因如下：

1. 平台独立性和灵活性：我们的绿植助手计划支持IOS、Android和Web端，微服务架构提供了更好的平台独立性，每个平台可以独立开发和部署相应的微服务。这使得我们能够更好地满足不同平台的需求，并在未来根据需要进行更灵活的扩展和演进。

2. 强大的可伸缩性：随着用户数量和业务规模的增长，我们需要一个可伸缩的架构来处理不断增加的负载。微服务架构的独立部署和水平扩展的特性使我们能够针对具体的微服务进行灵活的扩展，确保在高负载时仍能保持良好的性能

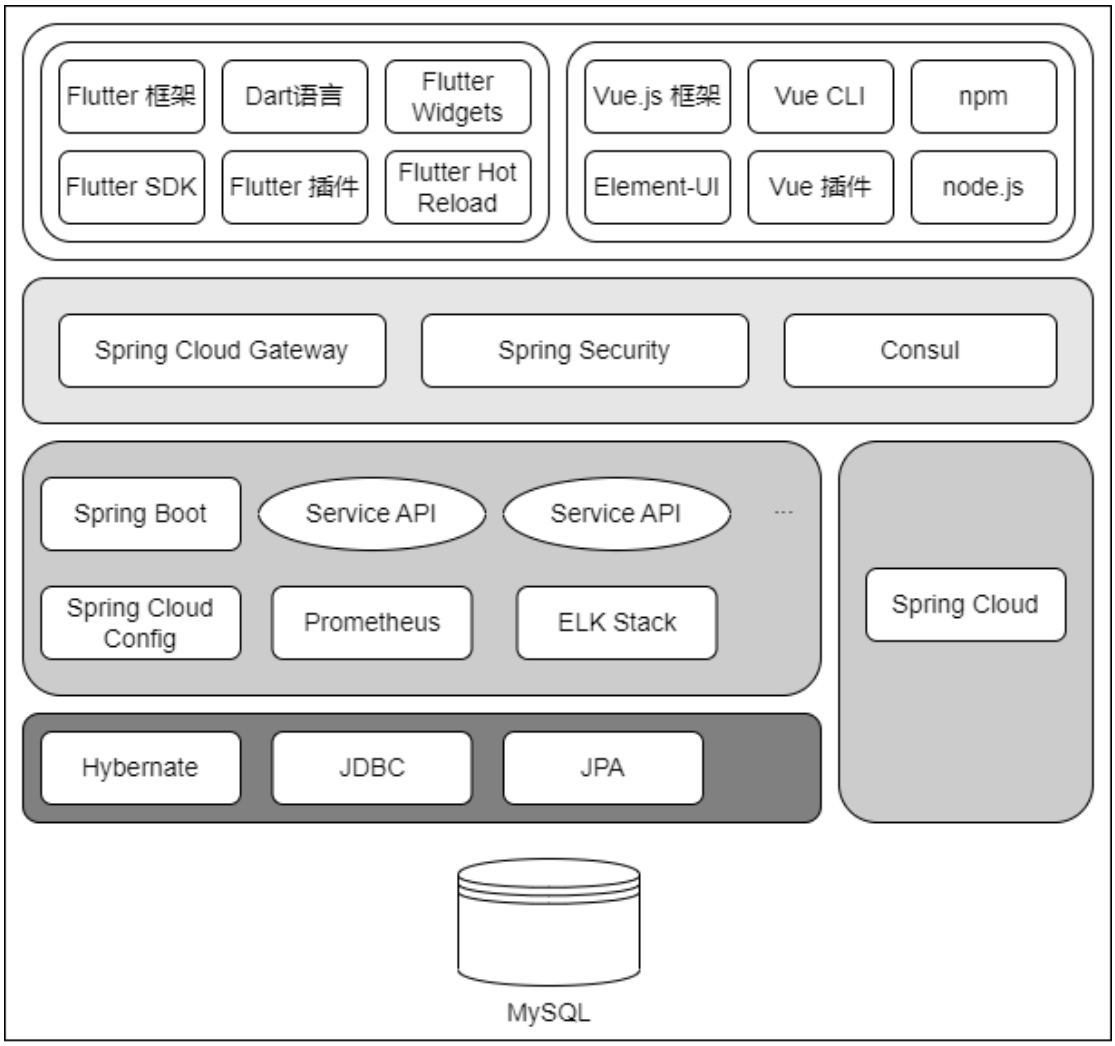
和可用性。

3. 高度松耦合的服务：微服务架构将系统拆分为一系列相对独立的服务，这些服务通过轻量级通信协议进行交互。这种松耦合的架构使得每个微服务可以独立开发、测试、部署和维护，从而提高了团队的开发效率和系统的可维护性。



尽管微服务架构需要更多的管理和运维工作，并且在设计和实现时需要考虑到分布式系统的复杂性，但我们相信这个决策能够带来更多长期的优势。通过采用微服务架构，我们能够实现更强的平台独立性、高可伸缩性和松耦合性，为绿植助手平台提供更好的灵活性、可扩展性和可维护性，以适应未来的需求变化和业务增长。

我们还对微服务架构进行了进一步的技术选型。在移动端，我们选择了采用 Flutter 框架。Flutter 是一个由 Google 开源的 UI 工具包，使用 Dart 语言编写，具有跨平台的能力。它提供了丰富的组件和快速的渲染能力，可以高效构建流畅、美观的原生移动应用。通过一套代码同时覆盖 iOS 和 Android 平台，我们能够更高效地开发、测试和部署移动应用，减少开发周期和维护成本。而在 Web 端，我们选用 Vue.js 框架。Vue.js 是一款轻量级的 JavaScript 框架，具有简洁的语法和响应式的数据绑定能力。它的设计理念注重可组合性和可测试性，使开发者能够快速构建交互性强的单页面应用。Vue.js 的生态系统丰富，提供了许多插件和工具，以便更高效地开发和维护 Web 应用。



除了移动端和 Web 端的框架选型，我们还设计了一系列的微服务技术栈来支持整个系统的开发和部署。为实现微服务架构的搭建和管理，我们采用了以下关键技术选型：

- 服务注册与发现：选择了 Consul 作为服务注册与发现工具，它提供了可靠的服务注册和发现机制，使得微服务能够动态地进行服务之间的通信和发现。同时，我们采用 Spring Cloud 作为微服务的开发框架，它提供了丰富的功能和工具，可以帮助我们构建可靠和弹性的分布式系统。

- 服务网关：为了实现统一的入口和路由管理，我们引入了 Spring Cloud Gateway 作为服务网关。它提供了灵活的路由和过滤功能，同时与 Spring Cloud 框架紧密集成，使得我们能够轻松地管理服务之间的请求流量和访问控制。

- 分布式配置中心：为实现配置的集中管理和动态刷新，我们选择了 Spring Cloud Config 作为分布式配置中心。它提供了集中式的配置存储和管理，支持动态刷新配置，使得我们能够方便地对微服务的配置进行管理和调整。

- 监控和日志系统：为了确保系统的可观察性和故障排查能力，我们引入了 Prometheus 和 ELK Stack 作为监控和日志系统。Prometheus 用于收集和存储指标数据，提供实时的监控和报警功能。ELK Stack（Elasticsearch、Logstash、Kibana）用于日志的收集、处理和可视化，帮助我们实时追踪和分析系统的日志信息。

除了上述选型，我们还采用了 Spring Security 作为安全框架，用于实现身份验证和访问控制。同时，我们使用 Hibernate 作为 ORM 框架，用于简化数据库操作和对象关系映射。

这些技术选型基于我们对各个组件功能、性能和可靠性的综合考虑，同时考虑与选择框架（如 Spring Boot 和 Spring Cloud）的兼容性和协同能力。这样，我们能够构建出一个稳定、可扩展和易于管理的微服务架构，满足整个系统的要求并为用户提供优质的体验。

2.2 子系统与接口设计

本项目设计的子系统及接口设计列表如下：

1. 登录注册子系统

子系统名称：LoginRegistrationSubsystem

接口类名称：LoginRegistrationCheck

登录注册子系统相关接口函数	
登录	logIn(string userID,string password):bool
登出	logOut():void
获取用户 ID	getUserID():userID
获取用户密码	getUserPassword():userPassword
检查用户登录状态	checkInfo(string userID):bool
打开注册界面	openRegister():void
创建新的用户	createUser(string userID,string password):userID
返回登录界面	returnToLogin():void

2. 植物医疗子系统

子系统名称：PlantMedicalSubsystem

接口类名称：PlantMedicalInterface

植物医疗子系统相关接口函数	
创建医疗需求	createMedicalService(string userID):bool
获取院区信息	getHospitalInformation(string hospitalID) : string
获取园丁信息	getDoctorInformation(string doctorID) : string
推荐园丁信息	recommendDoctor() :string
预约园丁	book(string userID,string doctorID) : bool
创建咨询空间	createChatRoom(string userID, string doctorID) : bool
发送信息	sendInfo(string userID, string doctorID, string Infomation): bool
查看信息	viewInfo(string userID, string doctorID, string Infomation): bool

3. 养护提醒子系统

子系统名称：MaintenanceReminderSubsystem

接口类名称：MaintenanceReminderInterface

养护提醒子系统相关接口函数	
返回所有的提醒信息	showAllReminder():string
返回指定提醒信息	getReminderInfo(string reminderID):string
创建新提醒	createNewReminder(string userID, string reminderID,string reminderName, string reminderContent ,string reminderTime):bool
更改提醒	setReminder(string userID, string reminderID,string reminderName, string reminderContent ,string reminderTime):bool
更改当前提醒开启状态，返回当前提醒状态	setStatus(string reminderID, bool status):bool

删除提醒	deleteReminder(string reminderID):bool

4. 社交圈子系统

子系统名称：SocialCircleSubsystem

接口类名称：SocialCircleInterface

社交圈子系统相关接口函数	
展示所有的帖子	showAllPost() :bool
创建新帖子并返回帖子 ID	AddPost(text newPost):string postID
更改帖子信息	setPost(string userID, text newpost): bool
审核帖子	auditPost(text newpost):bool
删除帖子	deletePost(string postID):bool
对帖子进行评论	commentPost(string userID, string postID, text comment):bool

5. 科普平台子系统

子系统名称：SciencePlatformSubsystem

接口类名称：SciencePlatformInterface

科普平台子系统相关接口函数	
创建新科普信息并返回创建 ID	createScienceInfo(string userID , text newInfo):string InfoID
展示所有科普信息	showAllScienceInfo() :bool
删除科普信息	deleteScienceInfo(string InfoID):bool
设置信息的科普状态	setScienceStatus(string userID, string InfoID, bool status):bool
设置信息的敏感状态	setSensitiveStatus(string userID, string InfoID, bool status):bool
对科普信息进行评论	commentScienceInfo(string userID, string InfoID, text comment): bool

6. 举报子系统

子系统名称：ReportingSubsystem
接口类名称：ReportingInterface

举报子系统相关接口函数	
创建一个举报信息，返回举报信息的 ID	createReport(string sendUserID, string ReportInfo): string reportID
撤销举报信息	revokeReport(string reportID):bool
查看对应举报信息	viewReport(string reportID):string
对举报信息进行审核并设置结果	auditReport(string reportID,bool state):bool
对举报信息设置反馈	createFeedback(string reportID, string FeedbackInfo)：bool
查看反馈信息	viewFeedback(string reportID):string

7. 商城管理系统

子系统名称：MallSystem
接口类名称：MallInterface

商城管理系统相关接口函数	
展示所有商品信息	showAllProduct():string
获取对应商品信息	getProductInformation(string prductID) : string
创建新商品	createProduct(string productID, string productName, string sellerID, string type, int price, string result) : bool
更改商品信息	changeProductInformation(string productID, string changeData) : bool
查看商品信息	reviewProductInformation(string productID, string storeID) : bool
开始购买	startPurchase(string orderID, string sellerID, string productID, string buyerID) : void
预定订单	startOrder(string productID, string sellerID, string buyerID) : bool

删除订单	deleteOrder(string orderID) : bool

8. 个人信息管理系统

子系统名称：PersonalInformationSystem

接口类名称：PersonalInformationInterface

商城管理系统相关接口函数	
添加用户信息	addUser(string userID,user information):bool
更改用户信息	updateUser(string userID,user information):bool
删除用户	deleteUser(string userID):bool

2.3 接口规范

以植物医疗系统与高德地图的接口为例，在设计植物医疗系统时，需要使用标记了植物医生位置的地图服务。为了实现这一功能，可以选择使用高德地图提供的接口。当用户进入植物医疗系统并查看植物医生位置时，系统就会自动调用接口，展示用户周边地图。本系统中，我们决定采用当下较为完善的高德地图所提供的接口。高德地图 JS API 是一套 JavaScript 语言开发的的地图应用编程接口，移动端、PC 端一体化设计，一套 API 兼容众多系统平台。 JS API 提供了 2D、3D 地图模式，满足绝大多数开发者对地图展示、地图自定义、图层加载、点标记添加、矢量图形绘制的需求，同时也提供了 POI 搜索、路线规划、地理编码、行政区查询、定位等众多开放服务接口。高德地图 API 的设计风格包括：

• RESTful 风格：高德地图 API 遵循 RESTful 架构风格，使用统一的 URL 和 HTTP 方法进行资源的访问和操作。例如，使用 GET 方法获取地图数据、使用 POST 方法提交数据等。

• 前后端分离：高德地图 API 的设计支持前后端分离开发模式，前端通过 HTTP 请求获取数据，后端提供数据接口进行响应。这种设计方式使得前端开发者能够更加方便地使用 API 提供的功能和数据。

• 请求参数：高德地图 API 的请求参数通常采用 URL 查询字符串的形式

进行传递。例如，在获取地图数据时，可以通过参数指定地图的中心点坐标、缩放级别等信息。

• JSON 格式：高德地图 API 的响应数据一般以 JSON 格式返回，这种格式易于解析和处理。开发者可以通过解析 JSON 数据获取地图、路线、位置等相关信息。

• 授权认证：高德地图 API 使用授权认证机制来保护数据的安全性。开发者需要在请求中提供有效的 API 密钥或签名等信息，以验证其身份和权限。

根据上述风格，高德地图 API 贴合绿植助手项目的合适选择。它将提供一种统一和易于使用的接口，以满足不同平台和功能模块的需求，并支持扩展和互操作性。查阅相关文档，可以发现文档中 API 所包含的参数及方法众多。其中创建地图接口参数值均可设为固定值，而用于标记植物医生位置的接口部分参数可如下表所示：

Name	Description
opts.map Map	要显示该 marker 的地图对象。
opts.position (Vector2|LngLat)	点标记在地图上显示的位置。
opts.icon (Icon|string)	在点标记中显示的图标。可以传一个图标地址，也可以传 Icon 对象。有合法的 content 内容设置时，此属性无效。
opts.content(string|HTMLElement)	点标记显示内容。可以是 HTML 要素字符串或者 HTML DOM 对象。content 有效时，icon 属性将被覆盖。
opts.title string	鼠标滑过点标记时的文字提示。不设置则鼠标滑过无文字提示。
opts.visible boolean	点标记是否可见，默认值：true。

对此，设计接口规范如下：

接口名称：PlantDoctorMapInterface

功能：连接植物医疗系统与高德地图服务，获取标记了植物医生位置的地图信息

接口类型：HTTP GET

传入数据：

Name	Description
opts.position（植物医生位置）	表示植物医生的地理位置坐标。
opts.content（植物医生基本信息）	包含植物医生的基本信息，用于展示在地图上。
Key（高德地图 API 密钥）	用于授权认证。
Location（用户位置）	当前位置的经纬度坐标，例如："latitude,longitude"。
radius	搜索半径，单位为米。
Output（响应数据的格式）	"json"。

除特定参数外，其他参数采用满足要求的固定值。

返回数据：返回一个 JSON 数据类型，表示地图窗口，用于展示标记了植物医生位置的地理信息。

示例响应：

```
Content-Type: application/json
{
  "status": "success",
  "userLocation": {
    "latitude": 40.712776,
    "longitude": -74.005974
  },
  "plantDoctors": [
    {
      "name": "植物医生 1",
      "location": {
        "latitude": 40.712345,
        "longitude": -74.123456
      },
      "address": "地址 1",
      "phone": "电话 1"
    },
    {
      "name": "植物医生 2",
      "location": {
        "latitude": 40.654321,
        "longitude": -74.543210
      },
      "address": "地址 2",
      "phone": "电话 2"
    }
```

```
  ],
  "mapURL": "https://maps.amap.com/show/map?id=123456"
}
```

根据返回的 URL，植物医疗系统能够展示高德电子地图，用于查看植物医生的位置信息。

2.4 接口规范举例详述

1. 植物医生子系统接口规范：

子系统名称：PlantMedicalSubsystem

接口类：PlantMedicalInterface

接口函数：

- createMedicalService(string userID): bool

  功能：创建医疗需求，用于用户创建植物医疗服务的需求

  参数：userID: 用户 ID

  返回值：创建医疗需求成功返回 true，否则返回 false

- getHospitalInformation(string hospitalID): string

  功能：获取院区信息，用于获取指定院区的相关信息

  参数：hospitalID: 院区 ID

  返回值：院区信息的字符串表示

- getDoctorInformation(string doctorID): string

  功能：获取园丁信息，用于获取指定园丁的相关信息

  参数：doctorID: 园丁 ID

  返回值：园丁信息的字符串表示

- recommendDoctor(): string

  功能：推荐园丁信息，用于向用户推荐合适的园丁

  返回值：推荐的园丁信息的字符串表示

- book(string userID, string doctorID): bool

  功能：预约园丁，用于用户预约指定的园丁提供的服务

  参数：userID: 用户 ID

  doctorID: 园丁 ID

返回值：预约成功返回 true，否则返回 false

- createChatRoom(string userID, string doctorID): bool

  功能：创建咨询空间，用于用户与园丁之间创建专属的咨询空间

  参数：userID: 用户 ID

  doctorID: 园丁 ID

  返回值：创建咨询空间成功返回 true，否则返回 false

- sendInfo(string userID, string doctorID, string information)

  功能：发送信息，用于在咨询空间中用户向园丁发送信息

  参数：userID: 用户 ID

  doctorID: 园丁 ID

  information: 要发送的信息

  返回值：发送信息成功返回 true，否则返回 false

- viewInfo(string userID, string doctorID, string information)

  功能：查看信息，用于在咨询空间中用户查看园丁发送的信息

  参数：userID: 用户 ID

  doctorID: 园丁 ID

  information: 要查看的信息

  返回值：查看信息成功返回 true，否则返回 false

说明：通过这些接口,用户可以与植物医生子系统进行交互,包括创建医疗需求、获取院区和园丁信息、预约园丁提供的服务、创建咨询空间以及在咨询空间中发送和查看信息。这些功能可以帮助用户获取植物医疗服务，并与园丁进行有效的沟通和咨询。

2. 养护提醒子系统接口规范：

子系统名称：MaintenanceReminderSubsystem

接口类：MaintenanceReminderInterface

接口函数：

- showAllReminder(): string

  功能：返回所有的提醒信息，用于展示用户所有的养护提醒

返回值：包含所有提醒信息的字符串表示

- getReminderInfo(string reminderID): string

功能：返回指定提醒信息，根据提供的提醒 ID 获取对应的养护提醒的详细信息

参数：reminderID: 提醒 ID

返回值：指定提醒信息的字符串表示

- createNewReminder(string userID, string reminderID, string reminderName, string reminderContent, string reminderTime): bool

功能：创建新提醒，用户可以根据提供的信息创建新的养护提醒，包括提醒 ID、名称、内容和时间

参数：userID: 用户 ID

reminderID: 提醒 ID

reminderName: 提醒名称

reminderContent: 提醒内容

reminderTime: 提醒时间

返回值：创建新提醒成功返回 true，否则返回 false

- setReminder(string userID, string reminderID, string reminderName, string reminderContent, string reminderTime): bool

功能：更改提醒，允许用户对现有的养护提醒进行修改，包括提醒 ID、名称、内容和时间

参数：userID: 用户 ID

reminderID: 提醒 ID

reminderName: 提醒名称

reminderContent: 提醒内容

reminderTime: 提醒时间

返回值：更改提醒成功返回 true，否则返回 false

- setStatus(string reminderID, bool status): bool

功能：更改当前提醒的开启状态，用户可以通过提供提醒 ID 和状态来开启或关闭特定的养护提醒

参数：reminderID: 提醒 ID

status: 提醒的开启状态，true 表示开启，false 表示关闭

返回值：更改后的当前提醒状态

- deleteReminder(string reminderID)：bool

  功能：删除提醒，根据提供的提醒 ID 删除指定的养护提醒

  参数：reminderID: 提醒 ID

  返回值：删除成功返回 true，否则返回 false

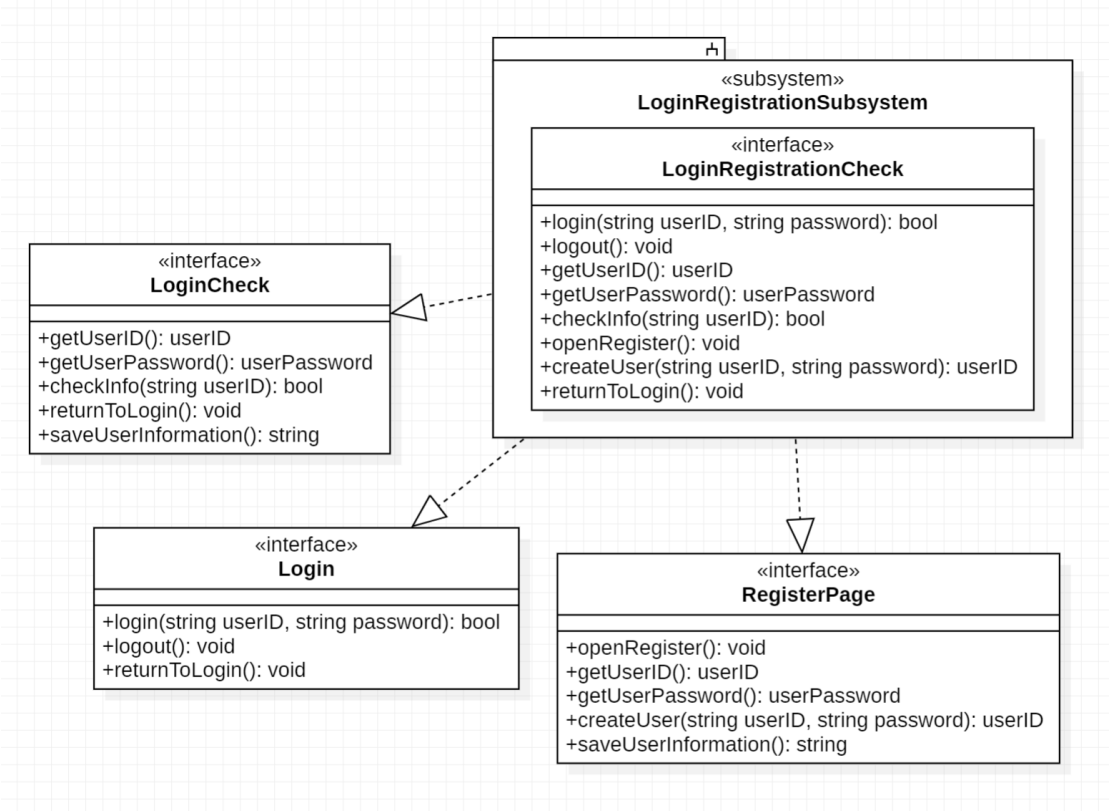
说明：通过这些接口，用户可以管理养护提醒，包括查看所有提醒、获取指定提醒的详细信息、创建新提醒、修改提醒内容和时间、更改提醒的开启状态以及删除提醒。这些功能帮助用户有效地管理植物的养护提醒，确保按时进行相关的养护操作。

三、设计模型

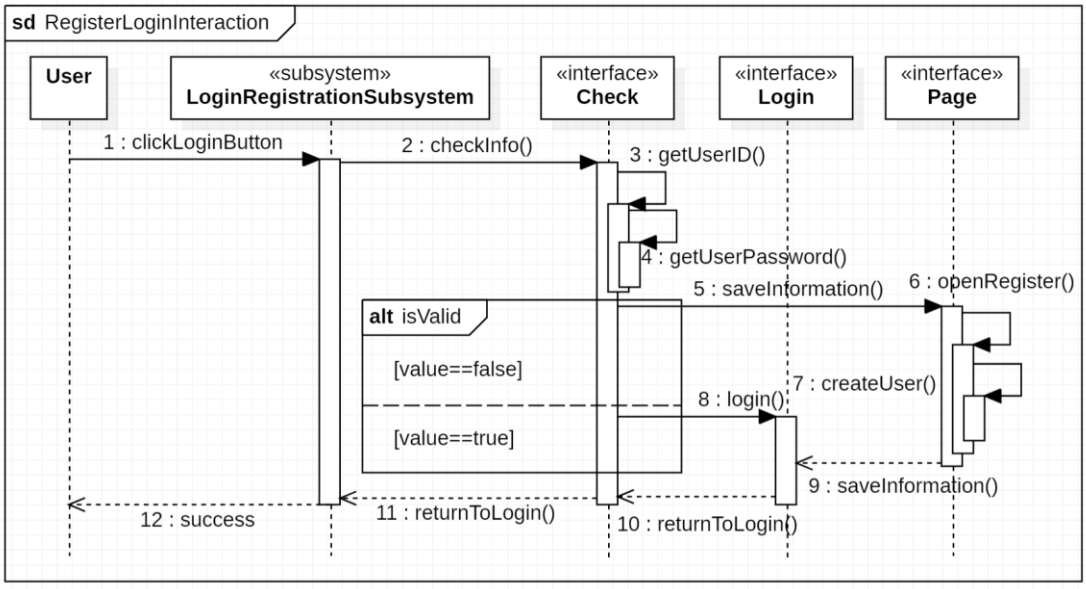
3.1 用例实现之登录与注册

用户点击进入登录注册页面时，系统会检查注册登录列表并获取更新该用户的具体注册信息；系统判断某一特定用户是否注册后将展示该用户所有已注册信息及具体信息的可选项；除此之外，系统还会根据算法判断已注册用户的注册信息的内容是否合法，将所判断好的合法信息设置提交给系统后系统将生成新的登录注册页面并发送给登录注册管理系统。

登录与注册类图：



登录与注册的时序图：

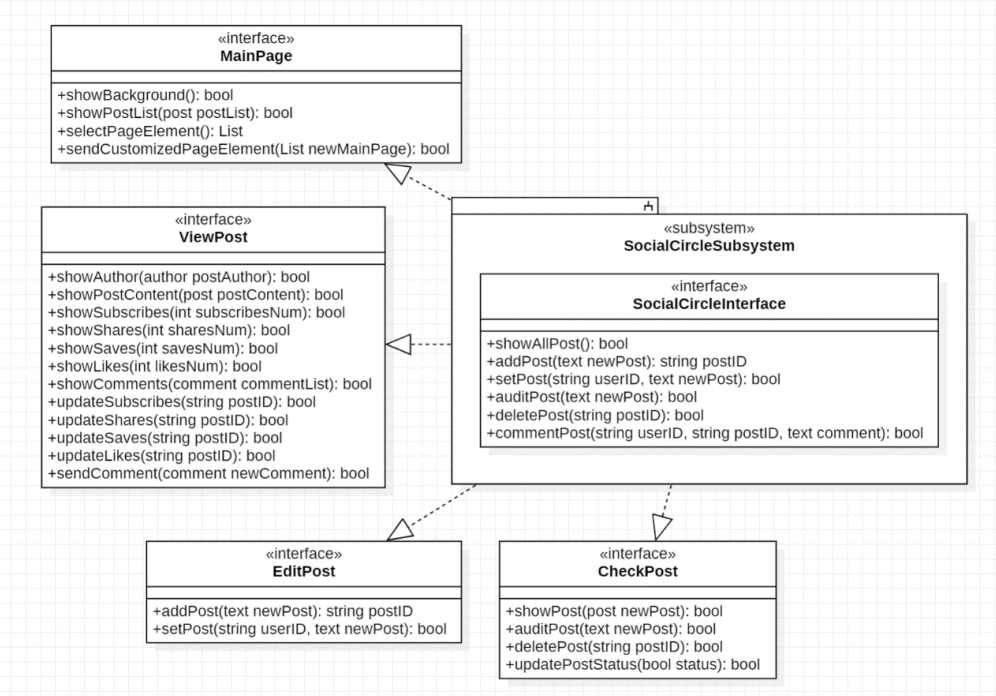


3.2 用例实现之社交圈

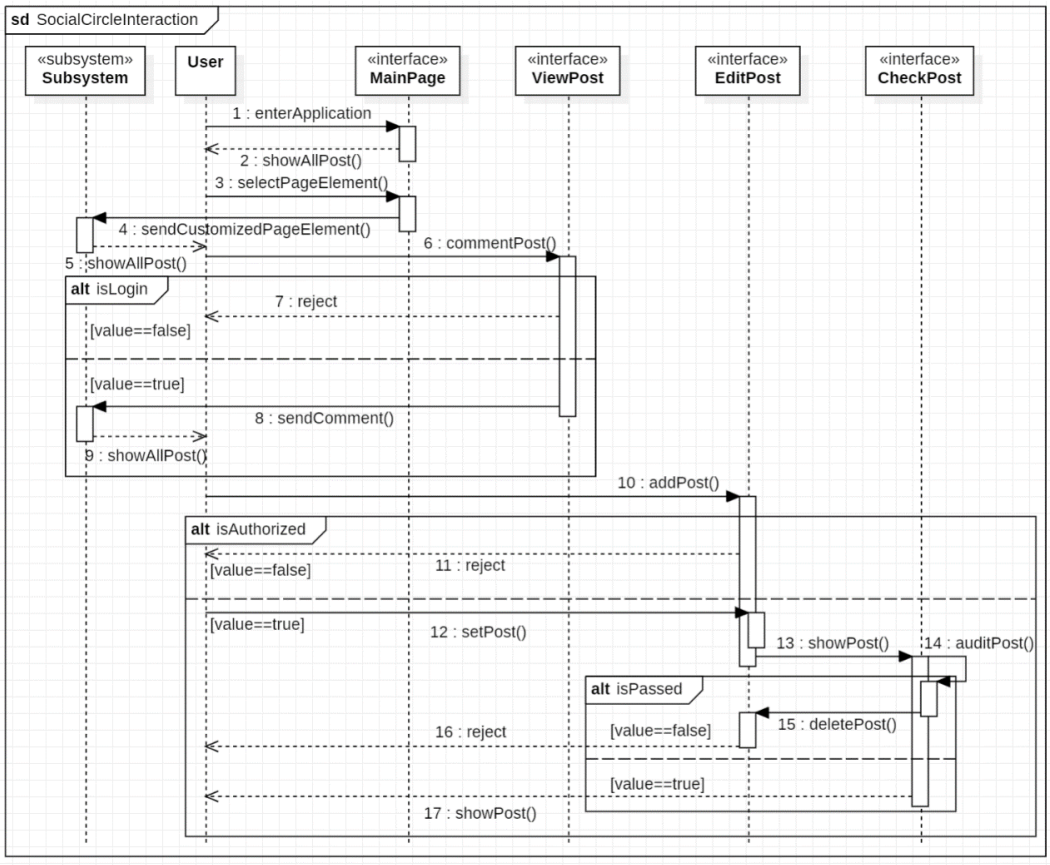
用户点击进入社交圈页面时，系统会读取帖子列表并获取更新各帖子的具体内容；用户选择某一特定帖子后系统将展示该帖子内所有点赞转发收藏关注信息

及具体信息的评论；除此之外，用户还可根据自己喜好选择主页面的内容展示，将所选偏好设置提交给系统后系统将生成新的主页面并发送给社交圈管理系统。

社交圈的类图：



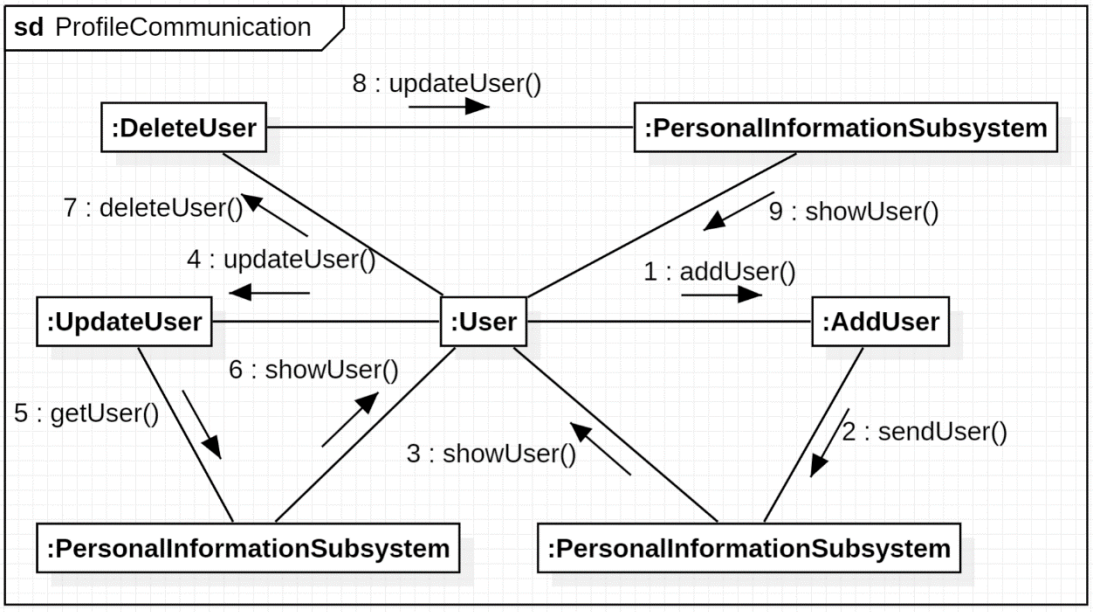
社交圈时序图：



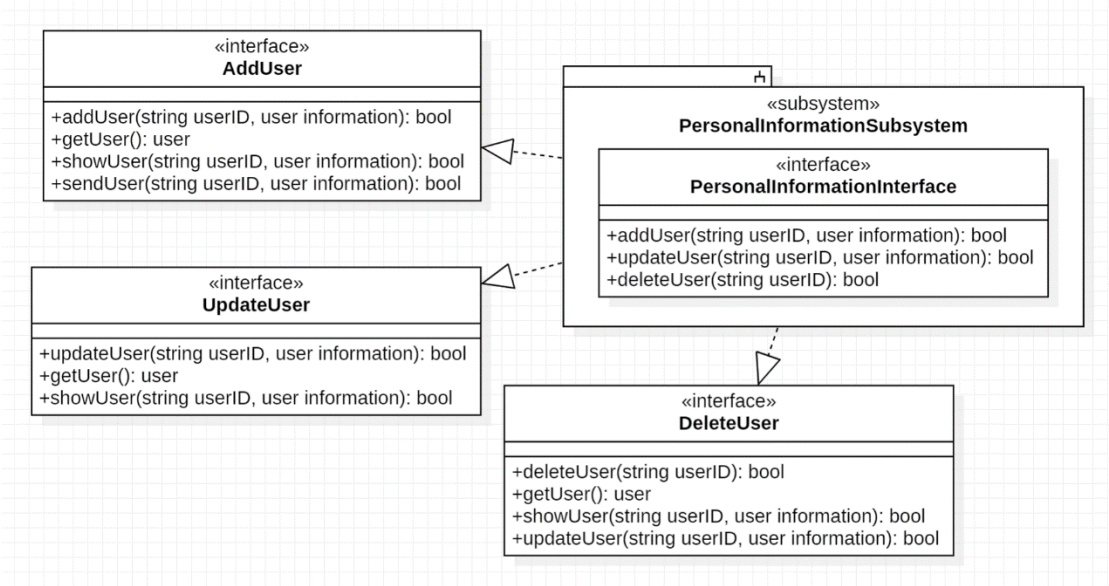
3.3 用例实现之个人信息管理

用户点击进入个人信息管理页面时，系统会读取个人信息列表并获取更新个人信息的具体内容；用户选择某一特定信息标签后系统将展示该信息标签内所有个人信息及具体信息的内容；除此之外，用户还可根据自己意愿修改个人信息页面的内容，将所修改的个人信息设置提交给系统后系统将生成新的个人信息主页面并发送给个人信息管理系统。

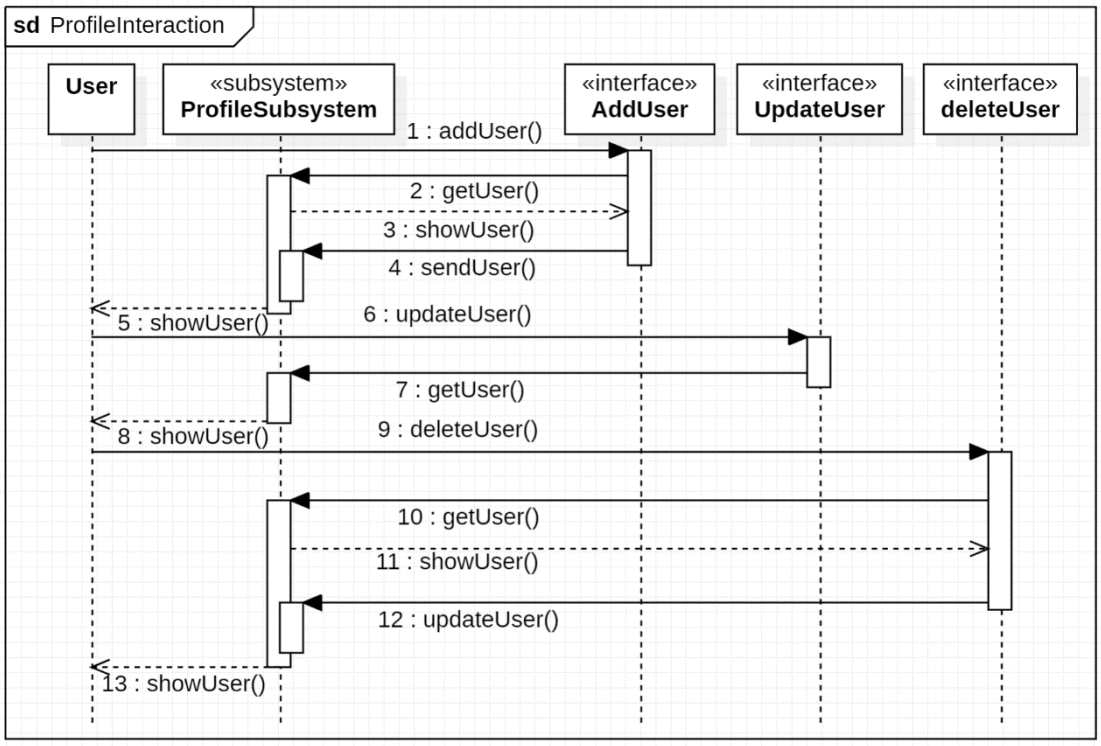
个人信息管理交流图：



个人信息管理类图：



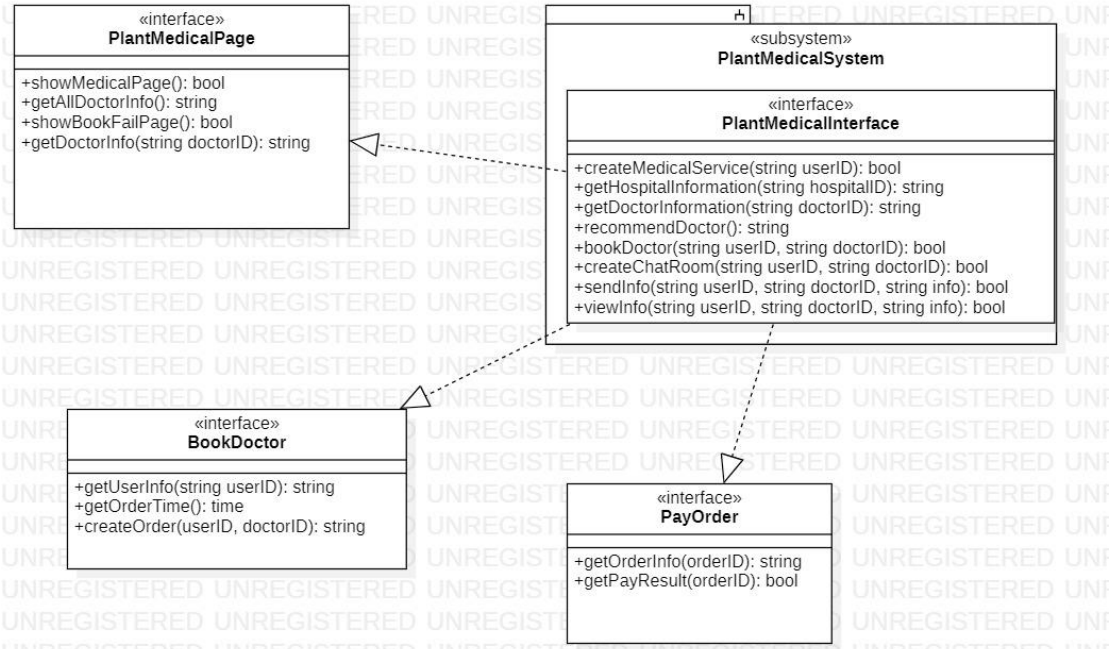
个人信息管理时序图：



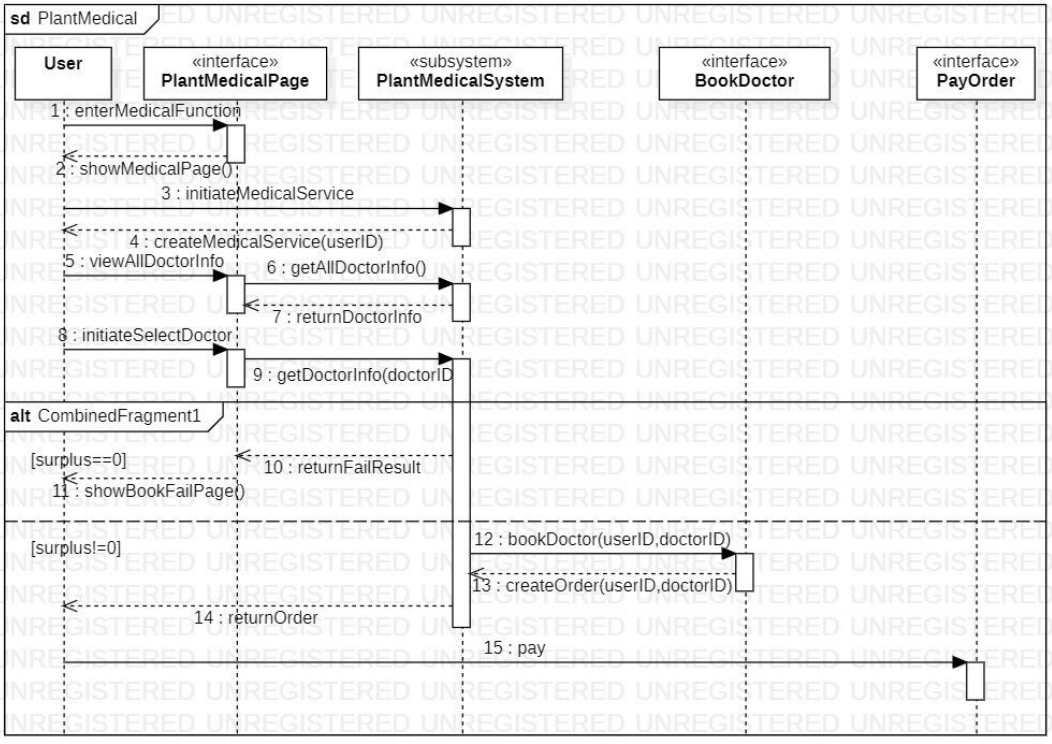
3.4 用例实现之个人信息管理

用户点击按钮进入植物医疗模块，系统展示植物医疗主页；然后，用户发起医疗服务请求，系统接收请求后创建医疗服务；之后，用户浏览所有医生信息，系统返回医生信息以供用户选择；用户挑选好心仪的医生后发起选择医生请求，系统获取该医生信息并检查该医生是否还有预约名额，若有，系统便预约该医生，同时创建订单并返回给用户进行支付；若没有，则展示预约失败页面。

植物医疗类图：



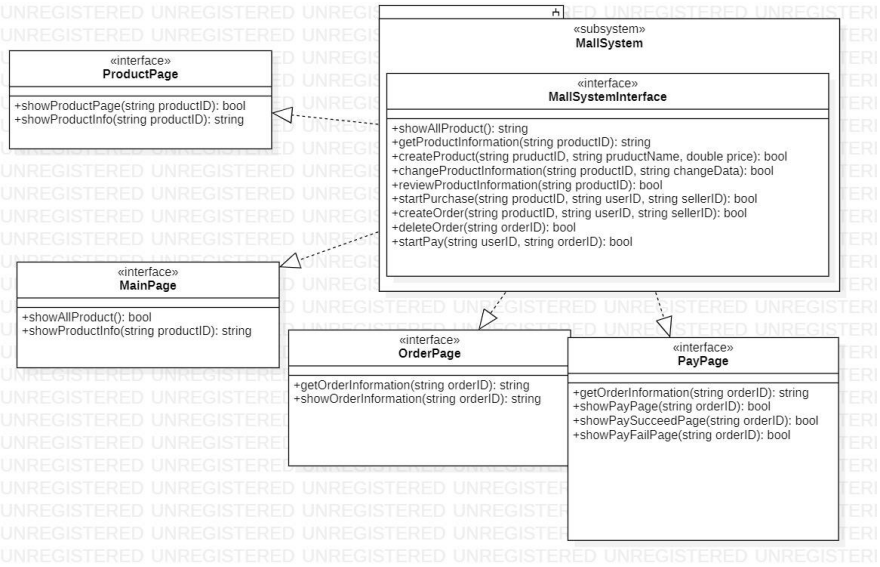
植物医疗时序图：



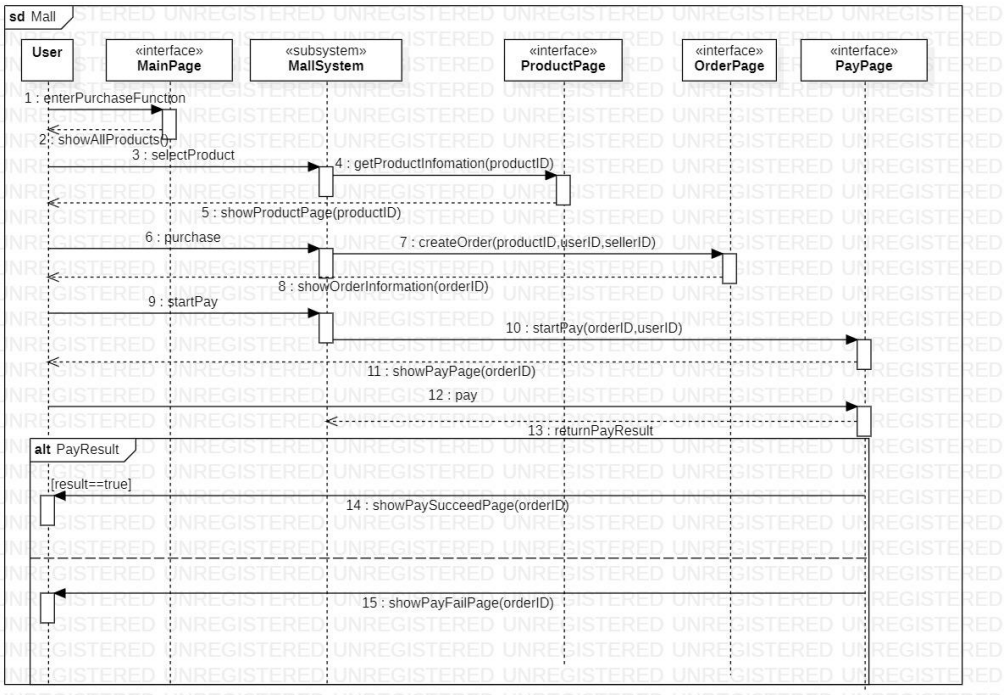
3.5 用例实现之植物商城

用户进入商城购买功能，系统返回商城主页，并为用户展示所有商品信息；用户通过浏览商品信息选择心仪的商品，系统根据商品 ID 返回该商品的详细介绍页面；用户准备购买商品，系统根据商品 ID、用户 ID、卖家 ID 创建订单，并为用户展示订单信息；用户支付后，系统根据支付成功或失败返回相应的页面。

商城购买类图：



商城购买时序图：

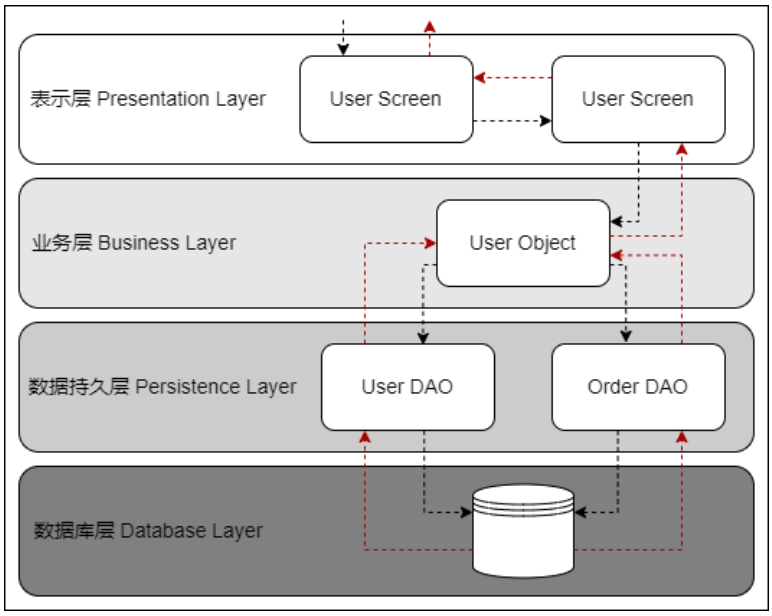


四、架构风格与关键决策

4.1 架构风格

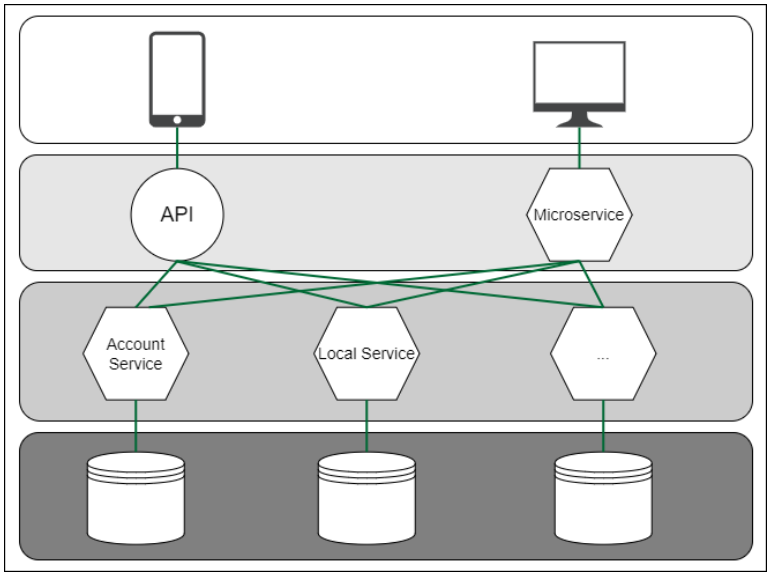
软件体系架构风格在软件开发中起到重要的作用，它们提供了一种组织和设计系统的方法。合适的架构风格有助于实现系统的可维护性、可扩展性、可重用性和性能等目标。以下是一些常见的软件体系架构风格以及它们的特点：

1. 分层体系结构（Layered Architecture）：这种风格将系统划分为多个层次，每个层次都有特定的功能和责任。它提供了良好的模块化和分离关注点的能力，使得系统更易于维护和理解。



2. 客户端-服务器体系结构（Client-Server Architecture）：将系统划分为客户端和服务器端，客户端发送请求并接收响应，服务器端处理请求并提供服务。这种架构风格适用于分布式系统，提供了可伸缩性和灵活性。
3. 事件驱动体系结构（Event-Driven Architecture）：系统中的组件通过事件进行通信和协作。当事件发生时，订阅该事件的组件将被通知并做出相应的响应。这种架构风格提供了松耦合和异步通信的能力。
4. 微服务体系结构（Microservices Architecture）：将系统拆分为一组小型、自治的服务，每个服务都专注于一个特定的业务功能。微服务架构具有高度可扩展性、独立部署和灵活性的优势，但也增加了分布式系统的复

杂性。



5. 领域驱动设计（Domain-Driven Design，DDD）：该方法注重领域模型的设计和实现，以解决复杂业务问题。它强调将领域专业知识嵌入到系统设计中，通过划分子域和定义聚合根等概念来组织系统。

4.1.1 微服务架构风格

根据第二部分的讨论，我们的项目采用了微服务体系结构。其特点包括系统的拆分和自治、微服务的单一责任性、松耦合等。微服务体系结构，在系统的开发过程中具有下面的优势：

1. 高可扩展性：由于每个服务都是独立的，可以根据需要对具有高负载的服务进行水平扩展，而无需扩展整个系统。
2. 独立部署和快速迭代：每个服务都可以独立地进行开发、测试和部署，使得团队可以更快地交付新功能和修复问题。
3. 灵活的技术栈：不同的服务可以使用适合其需求的不同技术栈，从而提供了更多的选择和灵活性。
4. 易于团队协作：微服务使得团队可以根据服务的职责进行划分，并独立地开发和管理服务。这种组织方式促进了团队间的协作和沟通。
5. 容错和隔离性：每个微服务都是独立的，因此当一个服务出现故障时，其他服务仍然可以正常运行。这提高了系统的容错性和可用性。

通过采用微服务体系结构，我们的项目能够充分利用服务的自治性、独立性

和松耦合性，从而提高开发速度、系统的可扩展性和容错性，实现高效的团队协作和持续交付。

4.1.2 数据抽象与面向对象体系结构

为了构建一个可扩展、可维护和安全的系统，我们决定采用数据抽象和面向对象的体系结构。这是一种广泛应用于软件开发的设计风格，它将数据和相关操作封装在对象中，并通过定义类、对象和继承关系来组织代码。我们选择这一体系结构风格的原因主要有以下几点：

1. 模块化和可维护性：数据抽象和面向对象体系结构提供了一种模块化的方式来组织代码。我们可以将相关的数据和操作封装在类中，使得代码更易于理解、扩展和维护。每个类都具有明确定义的职责，这样可以减少代码之间的耦合性，并提高系统的可维护性。
2. 代码重用和可扩展性：面向对象体系结构鼓励代码的重用和组件的可扩展性。我们可以通过定义基类和派生类的继承关系来实现代码的重用，并通过多态性来扩展和定制类的行为。这种灵活性使得我们可以根据系统需求进行快速的功能扩展和定制，同时保持代码的一致性和可维护性。
3. 数据安全性和封装性：通过数据抽象和封装，我们可以将数据的内部实现细节与对外接口分离。这样可以提高数据的安全性和完整性，并限制对数据的直接访问。通过定义适当的访问修饰符，我们可以控制对数据的访问权限，从而保护数据的机密性和一致性。

4.1.3 API 接口设计风格

API 接口设计风格对于构建可靠、可扩展和易于使用的应用程序至关重要。一方面，API 接口设计风格提供了一种统一的标准和规范，使得不同的应用程序能够以一致的方式进行通信和交互。这样可以降低学习成本，提高开发效率，并促进不同系统之间的集成。良好的 API 接口设计风格使得 API 易于理解和使用，通过清晰的命名、一致的结构和文档化的约定，提供了直观和友好的接口，降低了使用 API 的复杂性和困难度。另一方面，API 接口采用的设计风格应该支持系统的演进和扩展。它应该具有模块化的结构，允许添加新的功能和服务，同时保

持现有功能的稳定性。这样可以减少对现有代码的影响，并提高系统的灵活性和可维护性。此外，API 接口设计还应该充分考虑安全性和权限控制。它应该提供一种机制来验证请求的合法性，并限制对敏感数据和操作的访问，以保护系统免受未经授权的访问和攻击。目前常见的 API 接口设计风格有以下几种：

1. RESTful API：这是一种基于 Representational State Transfer（REST）原则的设计风格。它使用 HTTP 协议中的 GET、POST、PUT、DELETE 等方法来对资源进行操作，并使用 URL 来标识和访问资源。RESTful API 具有简单、可扩展、松耦合和易于缓存的特点。
2. GraphQL API：一种由 Facebook 开发的 API 查询语言和运行时环境。GraphQL 允许客户端定义所需的数据结构和字段，从而避免了过度获取或不足获取数据的问题。它具有灵活性、高度可定制和减少网络传输量的特点。
3. gRPC API：一种基于 Google 开发的开源高性能远程过程调用（RPC）框架的 API 设计风格。gRPC 使用 Protocol Buffers 作为接口定义语言（IDL），并使用 HTTP/2 作为传输协议。它具有高效、强类型和多语言支持的特点。
4. SOAP API：基于 Simple Object Access Protocol（SOAP）的 API 设计风格。它使用 XML 格式进行消息传输，并使用 Web 服务描述语言（WSDL）定义接口。SOAP 具有丰富的功能、安全性和可靠性，但相对于其他风格来说较为复杂。

每种 API 接口设计风格都有其特点和适用场景。对于本项目，考虑到其需要支持植物科普、信息管理、园艺社交、植物医疗和养护提醒等多个功能模块，以及跨不同平台（iOS、Android 和 Web）的支持，我们采用了 RESTful 的 API 设计风格。

RESTful API 是一种常见的 API 接口设计风格，其主要特点包括：

1. 基于 HTTP：RESTful API 使用 HTTP 协议作为通信协议，利用 HTTP 的方法（GET、POST、PUT、DELETE 等）对资源进行操作。
2. 资源导向：RESTful API 将系统中的数据和功能抽象为资源，并通过 URL（统一资源定位符）来标识和访问这些资源。
3. 状态转移：RESTful API 通过在请求中传递状态信息来实现资源的操作

和状态转移。这些状态信息可以使用 HTTP 的状态码来表示。

4. 无状态性：RESTful API 是无状态的，即每个请求都是独立的，服务器不需要保留客户端的状态信息。每个请求应该包含所有必要的信息。

RESTful API 具有广泛的支持和普及度,它符合标准的 HTTP 方法和状态码，具有直观的结构和语义，易于理解和调试。它的简单性和松耦合性使得不同的客户端可以灵活地调用和集成,同时支持缓存和扩展性。它支持 HTTP 的缓存机制，能够减少网络流量和服务器负载，提高系统的性能和可伸缩性。此外，RESTful API 也具有平台无关性，适合同时进行移动端和 Web 端的开发，因为它可以通过 URL 进行资源访问，并可以使用 JSON 等常见的数据格式进行数据交换。

4.2 关键决策

在系统设计过程中，我们进行的关键决策主要包括：

- 最初，我们考虑采用传统的三层分层架构风格。然而，在考虑到传统的三层体系结构存在的性能瓶颈、部署复杂性以及高耦合度等巨大的缺陷可能对我们项目开发造成不利的影响后,我们决定将架构设计风格转变为微服务架构。
- 考虑到移动端和 Web 端不同的开发生态特点及需求，我们对 UI 设计的框架进行了技术选型。在移动端，考虑到其丰富的组件与快速渲染能力，以及跨平台能力，我们选择了采用 Flutter 框架；在 Web 端，我们考虑到其快速构建强交互性单页面应用的能力，以及强大的生态系统，选用了 Vue.js 框架。
- 在微服务开发与支持方面，我们选用了 Spring Boot 快速开发框架，它能够提供自动配置和约定优于配置的原则,使得开发者能够快速搭建和启动应用程序，使得开发更加简单和高效。此外，我们采用 Spring Cloud 作为我们的微服务支持，它基于 Spring Boot，能够帮助开发人员轻松实现服务注册与发现、负载均衡、服务容错、分布式配置等关键微服务功能，使得微服务架构的开发和管理更加便捷。同时，它们还提供了容错和故障恢复机制，例如断路器模式和服务降级，以保障系统的稳定运行，有利于保证系统的稳定性与可靠性。

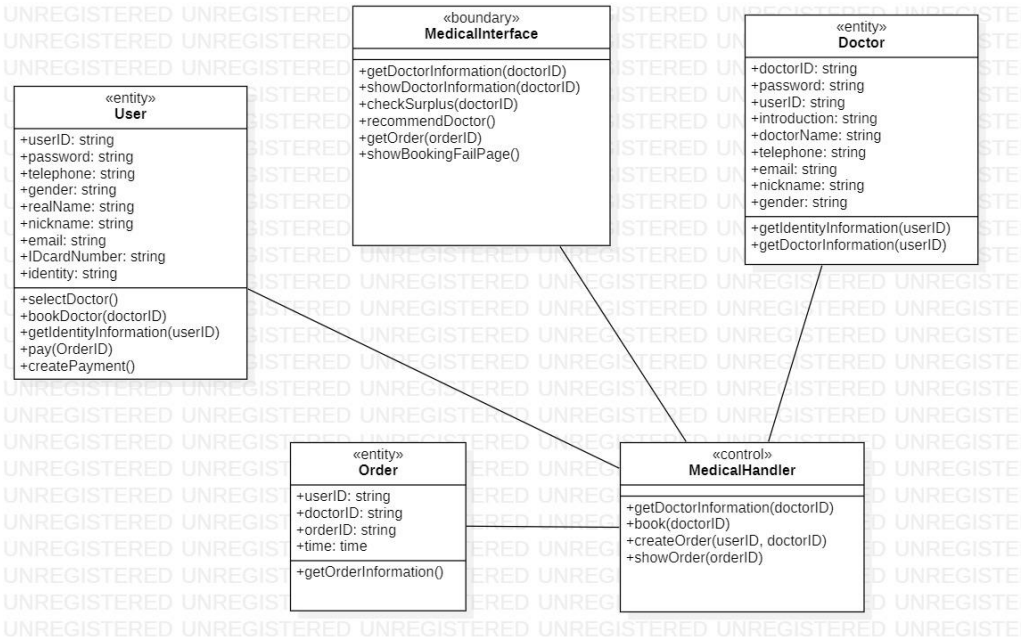
- 在安全支持方面，考虑到 Spring Security 强大的综合安全功能，包括身份验证、授权、密码存储和加密、会话管理、访问控制等，以及 Spring Security 与 Spring 框架的紧密集成及丰富的可配置选项和可扩展性，我们选用 Spring Security 来为应用程序提供安全功能。
- 在监控系统方面，我们决定采用 Prometheus，具有良好的可扩展性，能够提供强大的查询语言（PromQL），可以轻松处理大规模的监控需求，满足对高吞吐量数据收集和查询的需求。
- 在日志管理方面，我们选用了 ELK Stack，其由 Elasticsearch、Logstash 和 Kibana 组成，可以提供强大的监控和日志管理功能，帮助开发者实时了解系统的运行状态、性能指标和日志信息。此外，ELK Stack 还具有良好的可扩展性、灵活的查询和分析能力，以及丰富的可视化和报表功能，是构建高效监控和日志管理系统的理想选择。
- 我们选用了 Hibernate 作为对象关系映射框架，这是考虑到它可以带来对象关系映射、数据库无关性、自动化的数据库操作、缓存机制、事务管理以及映射和关联关系上的优势。Hibernate 是一种强大且成熟的持久化框架，可以提高开发效率、简化数据库操作，并提供良好的性能和可维护性。
- 为了实现良好的易用性、用户友好性及美观性，我们根据用户体验进行了页面设计和调研。根据调研结果，我们更新了用户交互设计并调整了特定功能，以提升用户体验。

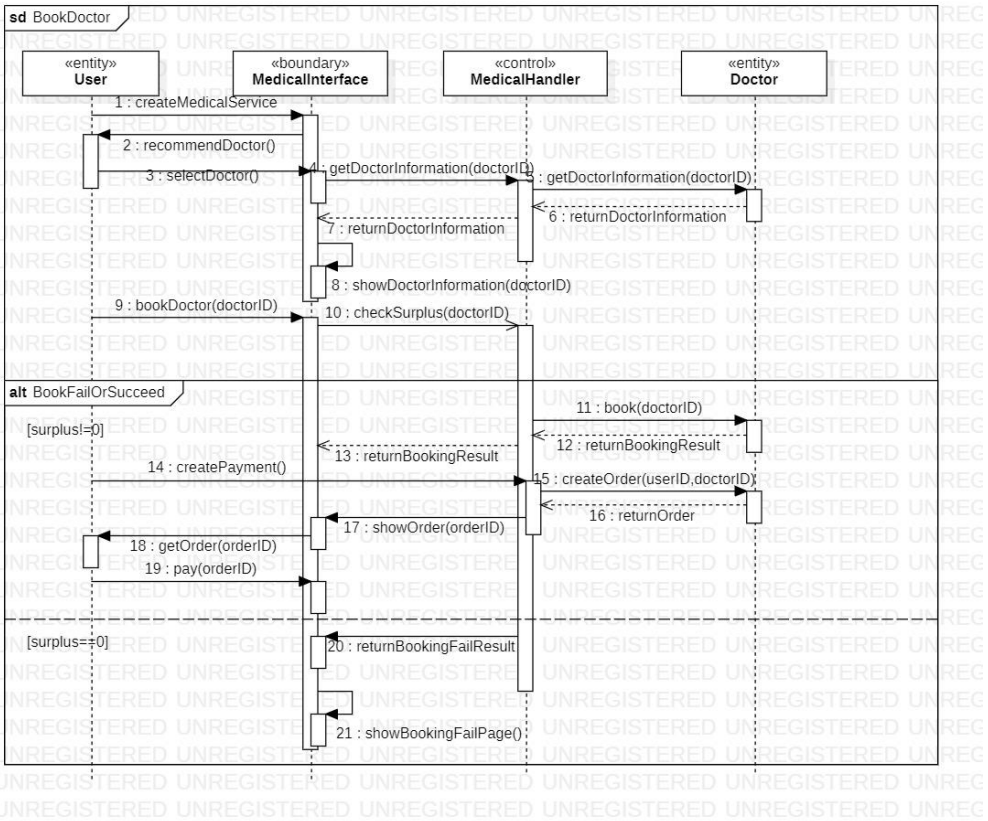
通过以上的一系列关键设计决策，我们能够保障用户数据的安全性和隐私性，提供高性能和良好的用户体验，并对系统的开发和管理进行有效的控制和保护。

五、更新的用例模型

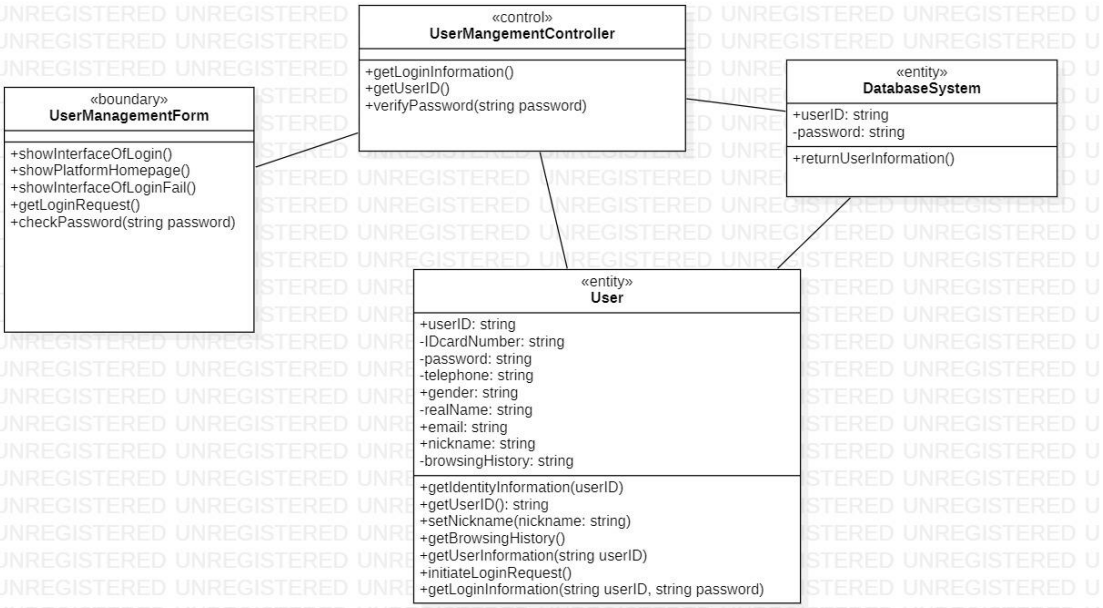
我们针对上一次的用例模型进行了更加详细地逻辑修改和细化，更新了一部分用例模型类图和时序图如下：

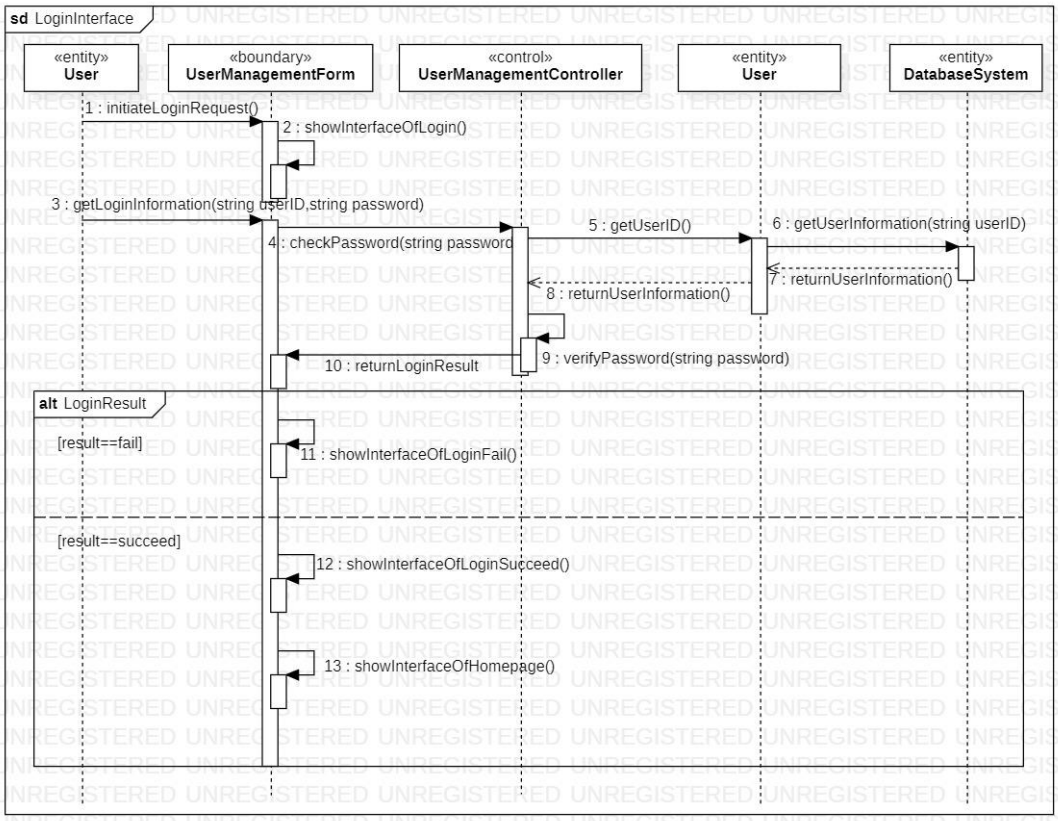
1. 植物医生预约功能


«boundary»
MedicalInterface
+getDoctorInformation(doctorID)
+showDoctorInformation(doctorID)
+checkSurplus(doctorID)
+recommendDoctor()
+getOrder(orderID)
+showBookingFailPage()
«entity»
Doctor
+doctorID: string
+password: string
+userID: string
+introduction: string
+doctorName: string
+telephone: string
+email: string
+nickname: string
+gender: string
+getIdentityInformation(userID)
+getDoctorInformation(userID)
«entity»
User
+userID: string
+password: string
+telephone: string
+gender: string
+realName: string
+nickname: string
+email: string
+IDcardNumber: string
+identity: string
+selectDoctor()
+bookDoctor(doctorID)
+getIdentityInformation(userID)
+pay(OrderID)
+createPayment()
«entity»
Order
+userID: string
+doctorID: string
+orderID: string
+time: time
+getOrderInformation()
«control»
MedicalHandler
+getDoctorInformation(doctorID)
+book(doctorID)
+createOrder(userID, doctorID)
+showOrder(orderID)

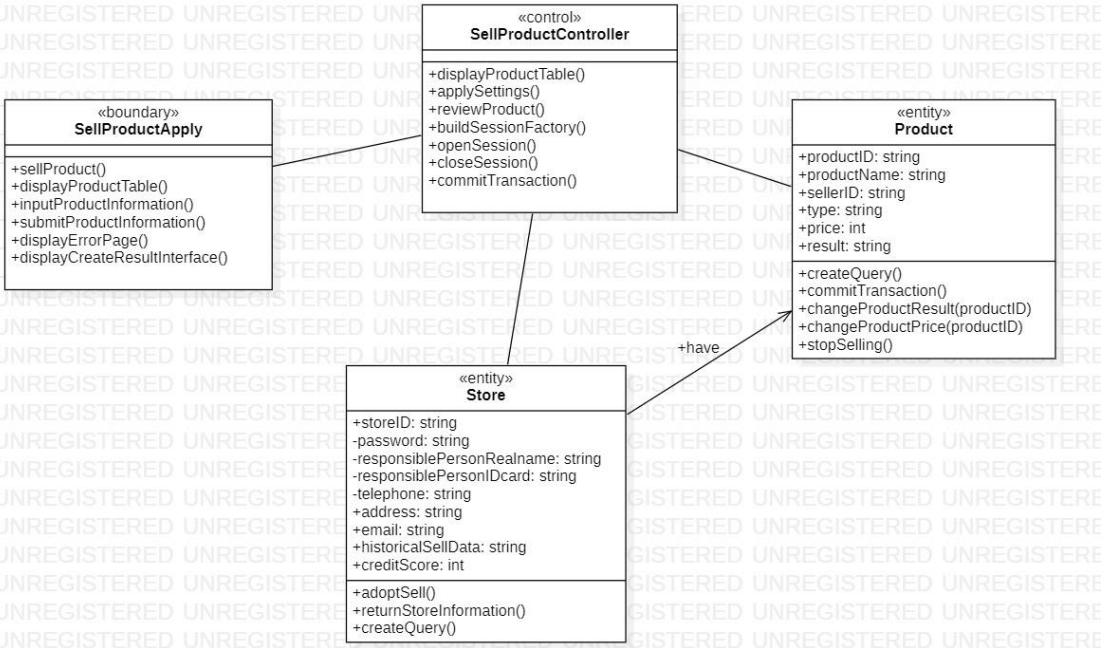

sd BookDoctor
«entity»
User
«boundary»
MedicalInterface
«control»
MedicalHandler
«entity»
Doctor
1 : createMedicalService
2 : recommendDoctor()
3 : selectDoctor()
4 : getDoctorInformation(doctorID)
5 : getDoctorInformation(doctorID)
6 : returnDoctorInformation
7 : returnDoctorInformation
8 : showDoctorInformation(doctorID)
9 : bookDoctor(doctorID)
10 : checkSurplus(doctorID)
alt BookFailOrSucceed
[surplus!=0]
11 : book(doctorID)
12 : returnBookingResult
13 : returnBookingResult
14 : createPayment()
15 : createOrder(userID,doctorID)
16 : returnOrder
17 : showOrder(orderID)
18 : getOrder(orderID)
19 : pay(orderID)
[surplus==0]
20 : returnBookingFailResult
21 : showBookingFailPage()

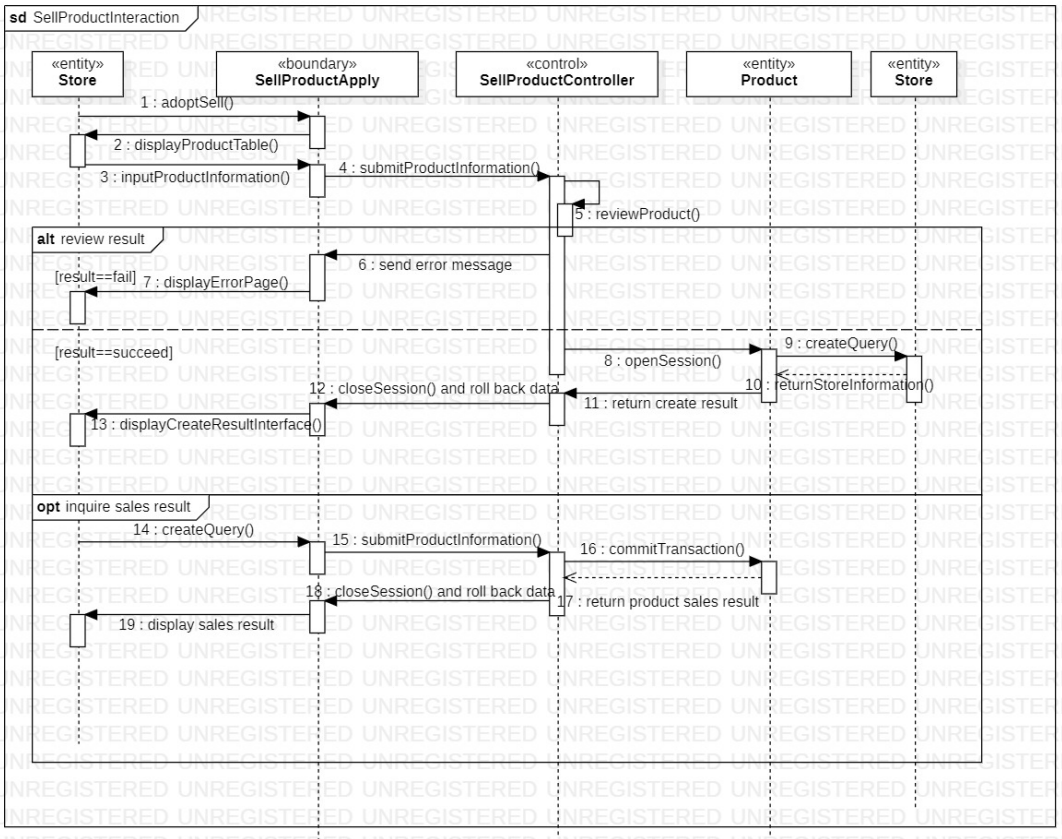
2. 注册登录功能


«control»
UserMangementController
+getLoginInformation()
+getUserID()
+verifyPassword(string password)
«boundary»
UserManagementForm
+showInterfaceOfLogin()
+showPlatformHomepage()
+showInterfaceOfLoginFail()
+getLoginRequest()
+checkPassword(string password)
«entity»
DatabaseSystem
+userID: string
-password: string
+returnUserInformation()
«entity»
User
+userID: string
-IDcardNumber: string
-password: string
-telephone: string
+gender: string
-realName: string
+email: string
+nickname: string
-browsingHistory: string
+getIdentityInformation(userID)
+getUserID(): string
+setNickname(nickname: string)
+getBrowsingHistory()
+getUserInformation(string userID)
+initiateLoginRequest()
+getLoginInformation(string userID, string password)


sd LoginInterface
«entity»
User
«boundary»
UserManagementForm
«control»
UserManagementController
«entity»
User
«entity»
DatabaseSystem
1 : initiateLoginRequest()
2 : showInterfaceOfLogin()
3 : getLoginInformation(string userID,string password)
4 : checkPassword(string password)
5 : getUserID()
6 : getUserInformation(string userID)
7 : returnUserInformation()
8 : returnUserInformation()
9 : verifyPassword(string password)
10 : returnLoginResult
alt LoginResult
[result==fail]
11 : showInterfaceOfLoginFail()
[result==succeed]
12 : showInterfaceOfLoginSucceed()
13 : showInterfaceOfHomepage()

3. 商城购买功能


«control»
SellProductController
+displayProductTable()
+applySettings()
+reviewProduct()
+buildSessionFactory()
+openSession()
+closeSession()
+commitTransaction()
«boundary»
SellProductApply
+sellProduct()
+displayProductTable()
+inputProductInformation()
+submitProductInformation()
+displayErrorPage()
+displayCreateResultInterface()
«entity»
Product
+productID: string
+productName: string
+sellerID: string
+type: string
+price: int
+result: string
+createQuery()
+commitTransaction()
+changeProductResult(productID)
+changeProductPrice(productID)
+stopSelling()
+have
«entity»
Store
+storeID: string
-password: string
-responsiblePersonRealname: string
-responsiblePersonIDcard: string
-telephone: string
+address: string
+email: string
+historicalSellData: string
+creditScore: int
+adoptSell()
+returnStoreInformation()
+createQuery()


sd SellProductInteraction
«entity» Store
«boundary» SellProductApply
«control» SellProductController
«entity» Product
«entity» Store
1 : adoptSell()
2 : displayProductTable()
3 : inputProductInformation()
4 : submitProductInformation()
5 : reviewProduct()
alt review result
6 : send error message
[result==fail]
7 : displayErrorPage()
[result==succeed]
8 : openSession()
9 : createQuery()
10 : returnStoreInformation()
11 : return create result
12 : closeSession() and roll back data
13 : displayCreateResultInterface()
opt inquire sales result
14 : createQuery()
15 : submitProductInformation()
16 : commitTransaction()
17 : return product sales result
18 : closeSession() and roll back data
19 : display sales result

六、项目成员分工贡献

在完成本阶段工作的过程中，所有团队成员都积极参与了小组讨论，并认真完成了各自的任务。团队成员合作融洽，及时沟通并解决了所有的问题。成员具体分工与贡献如下表所示：

姓名	用例模型更新	平台相关架构	子系统接口设计	详细的用例实现	详细的类设计	架构风格及关键决策	文档写作
韩嘉睿		✔	✔	✔		✔	✔
傅佳恒		✔	✔	✔		✔	✔
刘杰	✔		✔	✔	✔		✔
赵明泽	✔		✔	✔	✔		✔