

北京理工大学计算机学院

《Java 程序设计》课程设计模板

班级____07112304____学号____1120233608____姓名____

1 程序的运行环境、安装步骤

项目仓库: <https://github.com/oosquare/neocourse>

运行环境: JRE 21

程序的组成部份: 生产环境下仅需要已打包好的 jar, 以及可选的配置文件和已有的 SQLite 数据库文件。

安装和运行步骤:

- 安装 JRE 21。
- 将程序 jar 文件、配置文件、数据库（可选）复制到计算机上同一目录下。
- 修改配置文件中的默认管理员相关选项, 或者通过环境变量指定（见下文）。
- 打开 PowerShell, 更改工作目录, 输入以下命令: `java -jar neocourse-0.1.0.jar`
- 在浏览器中访问网站 `http://<ip>:8080`, 若为本机, 则<ip>为 127.0.0.1

2 程序开发平台

代码行数: 约 13000 行

开发环境: NixOS 24.11/Windows 11, IntelliJ IDEA 2024.3 CE, JDK 21.0.2, SQLite 3.47.0

技术栈: Spring Framework, Spring Boot, Spring Security, JPA, Vaadin Flow

3 程序功能说明:

3.1 用例总览

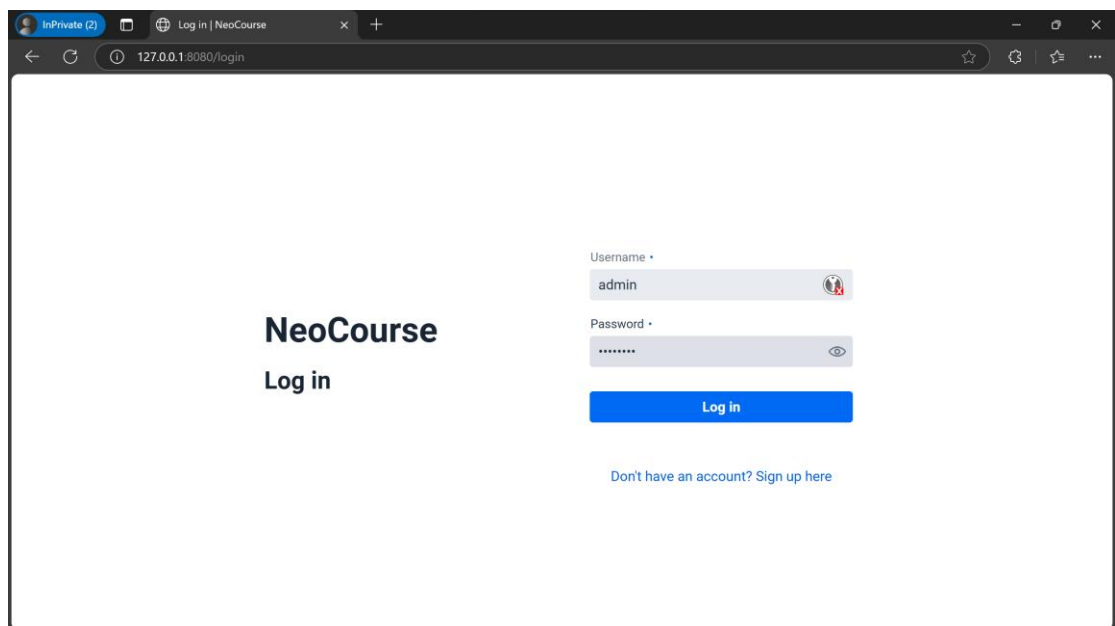
本程序名为 NeoCourse, 其为一个选课网站, 支持添加课程、选择课程、课程评分、考勤管理等多种功能。其设计目的为替换已有的物理选课网站, 提供更高的性能和更友好的移动端使用体验。

本系统共有三种用户 User——学生 Student、老师 Teacher、管理员 Administrator。这三种用户 User 并非完全独立, 一个用户 User 可以同时拥有老师 Teacher 和管理员 Administrator 两者类型, 所以用户 User 也对应着职责 Role。职责 Role 中, 学生 Student 与老师 Teacher、管理员 Administrator 互斥。每种用户 User 及其用例如下图所示。

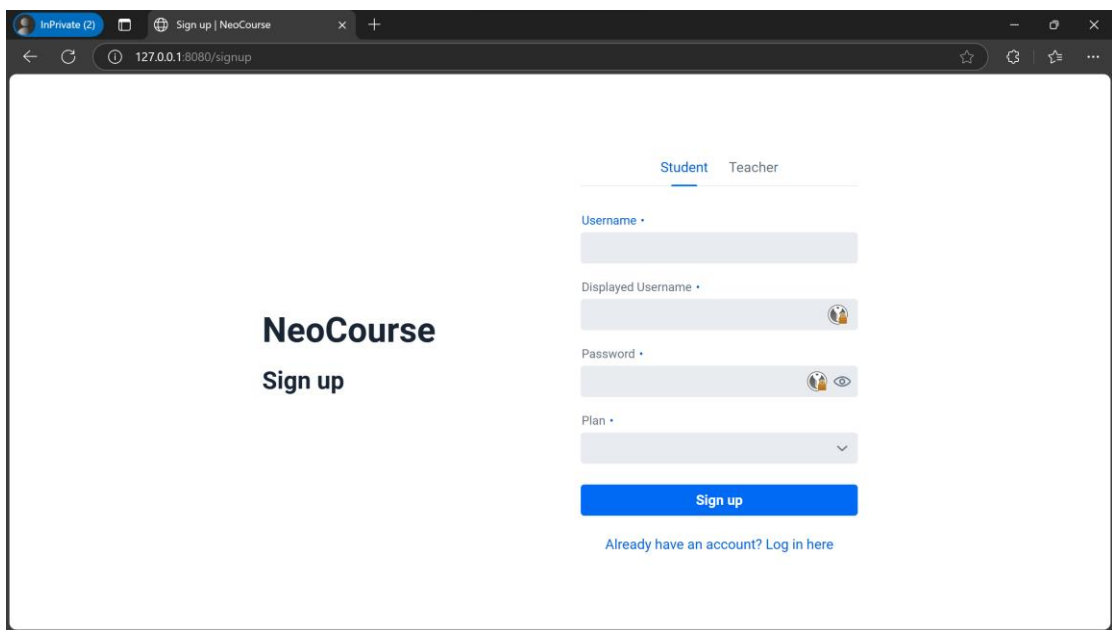


3.2 主要功能展示

- 登录页面:



- 注册页面：



Student Teacher

Username *

Displayed Username *

Password *

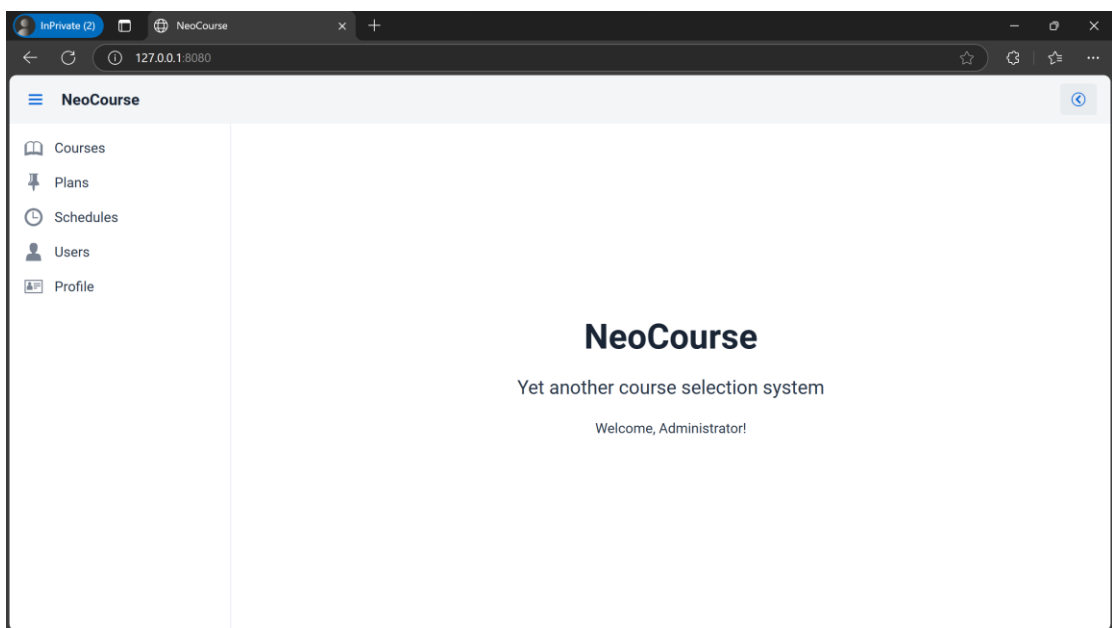
Plan *

Sign up

Already have an account? Log in here

学生 Student 注册需要填写用户名、显示用户名、密码，并选择一个计划 Plan，其中计划 Plan 是可选择的课程的集合，并包括一些评估考核指标。

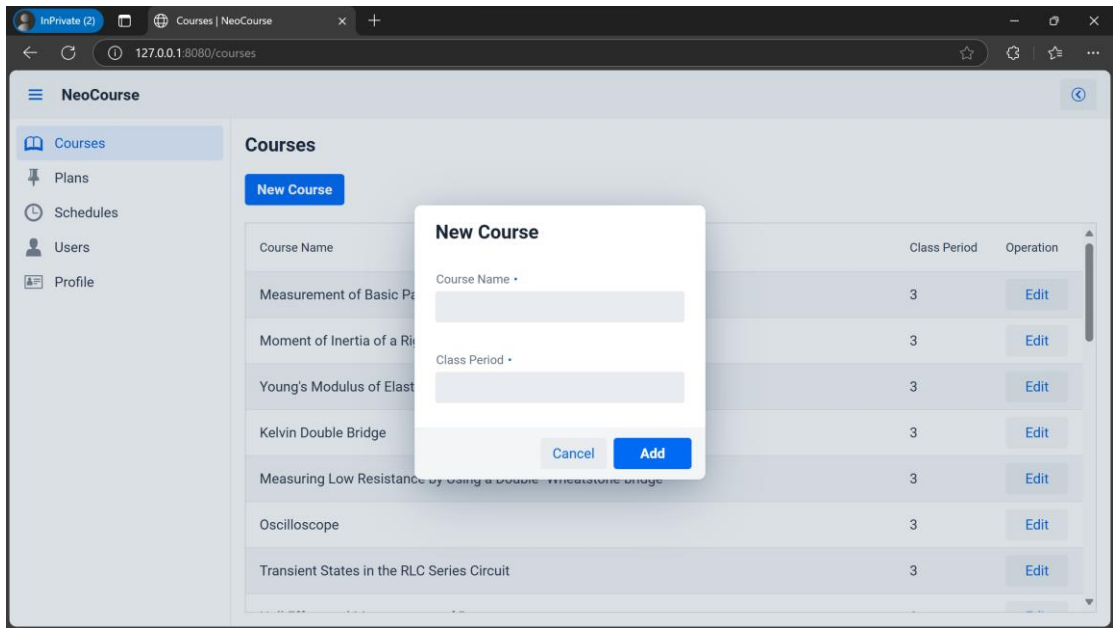
- 网站主页：



以上是管理员 Administrator 的主页，包括课程查看、计划查看、安排查看、用户列表和管理、用户信息。

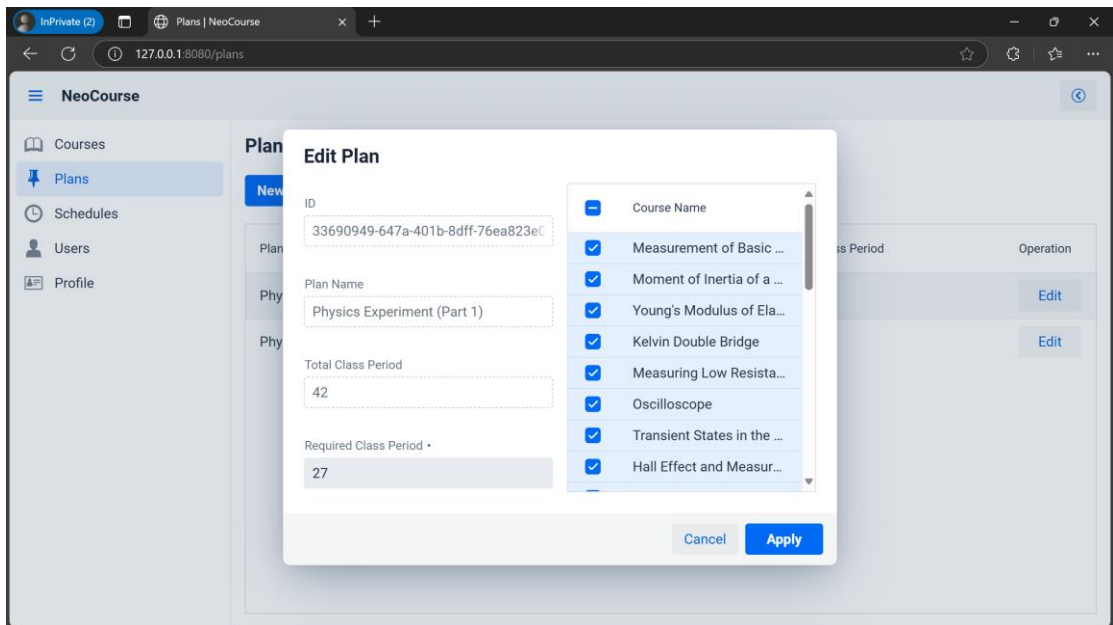
老师 Teacher 的主页类似，学生 Student 的主页则不同，包括选课、已选课查看、成绩查看、用户信息。

- 课程 Course 的创建：



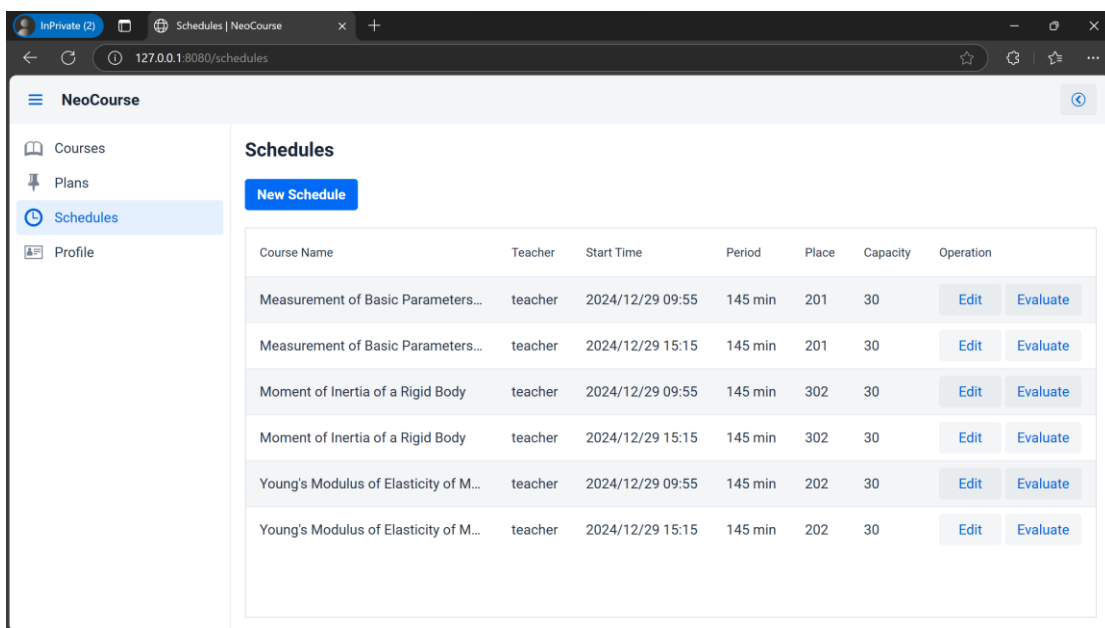
系统中具有老师 Teacher 职责的用户可以创建和删除课程 Course。

- 计划 Plan 的编辑：



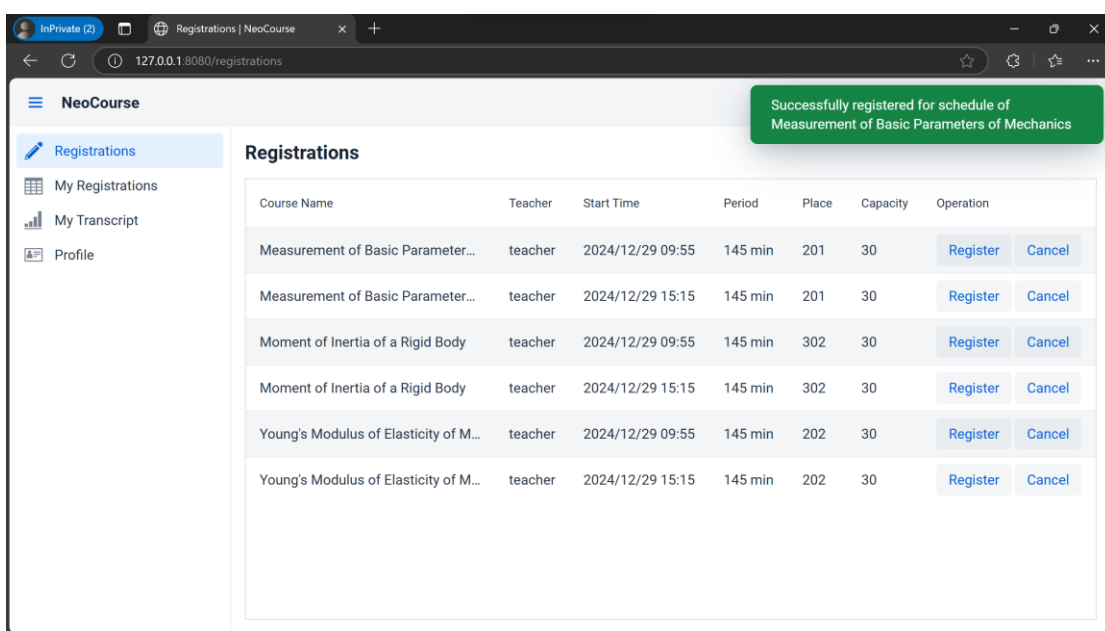
系统中具有老师 Teacher 职责的用户可以创建和删除计划 Plan。

- 安排 Schedule 管理：



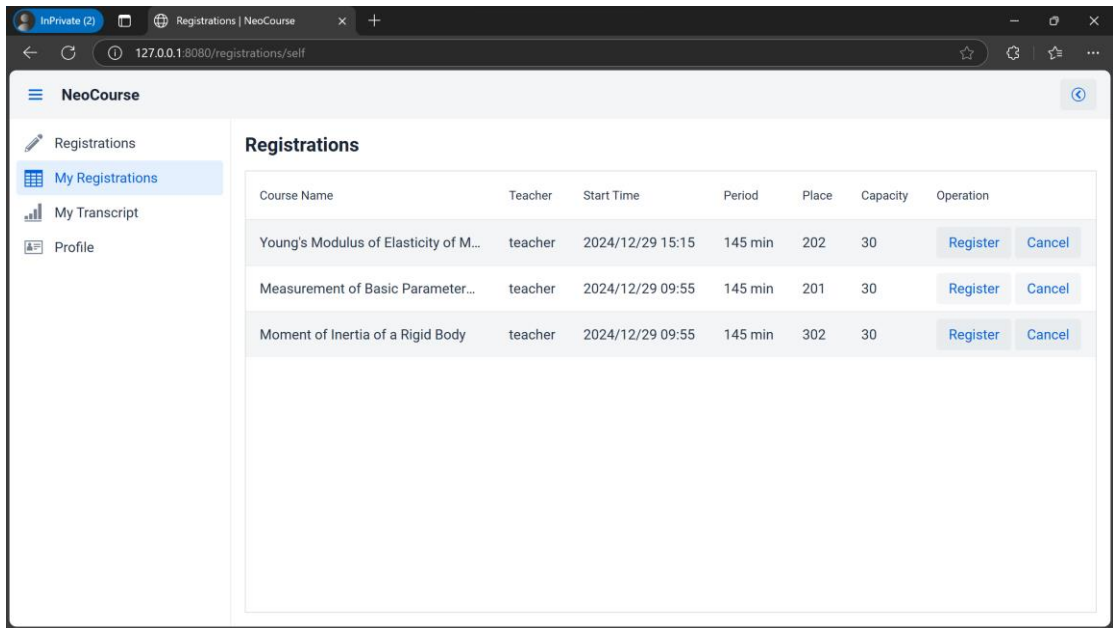
系统中具有老师 Teacher 职责的用户可以创建和删除安排 Schedule、查看指定安排 Schedule 的已选学生 Student 列表并评分。

- 学生 Student 选课页面：

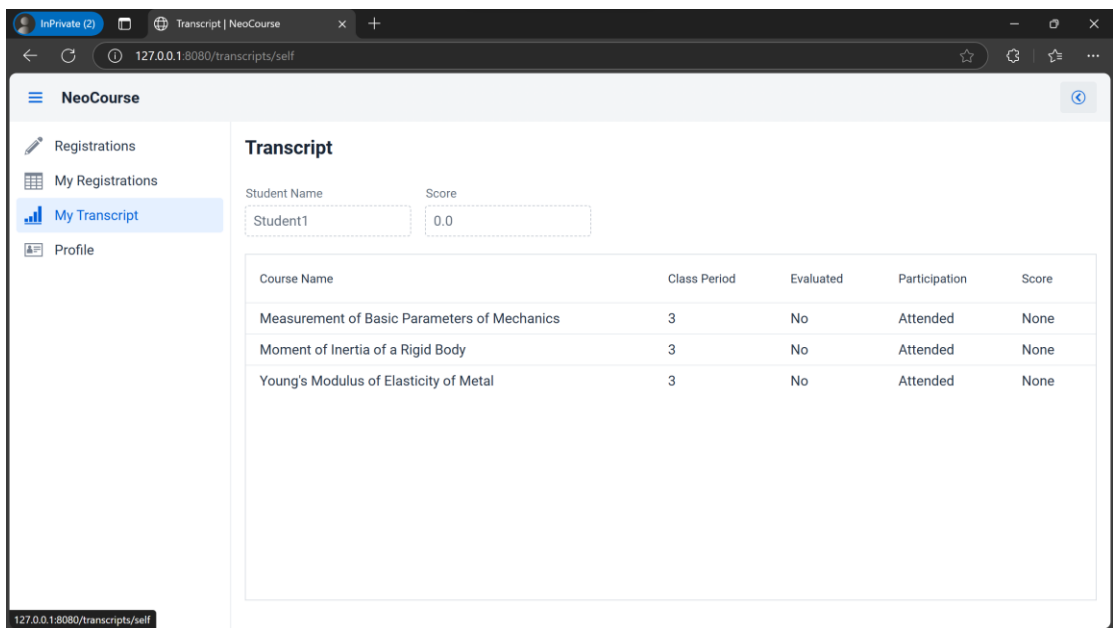


系统中具有学生 Student 的用户可以在所有安排列表中选择注册一些安排 Schedule，要求所有选择的安排 Schedule 对应的课程 Course 不重复其都在学生 Student 的 Plan 中，且不超过上限。

- 学生 Student 已选安排 Schedule 查看：



- 学生 Student 查看成绩单 Transcript:



- 老师 Teacher 评分:

NeoCourse

Evaluation

ID: 74a70498-ba31-487b-9e... Course Name: Measurement of Basic F

Student: Student1 Participation Status: ☒ Attended ☐ Absent Score: 90

[Update](#)

Student: Student2 Participation Status: ☐ Attended ☒ Absent Score: 0.0

[Update](#)

[Return](#)

老师 Teacher 可以为每个学生 Student 的此次安排 Schedule 的情况评分，包括分数评定和考勤评定。

- 用户 User 修改密码：

NeoCourse

User

Username: teacher

Displayed Username: teacher

Role: teacher

[Change Password](#)

Change Password

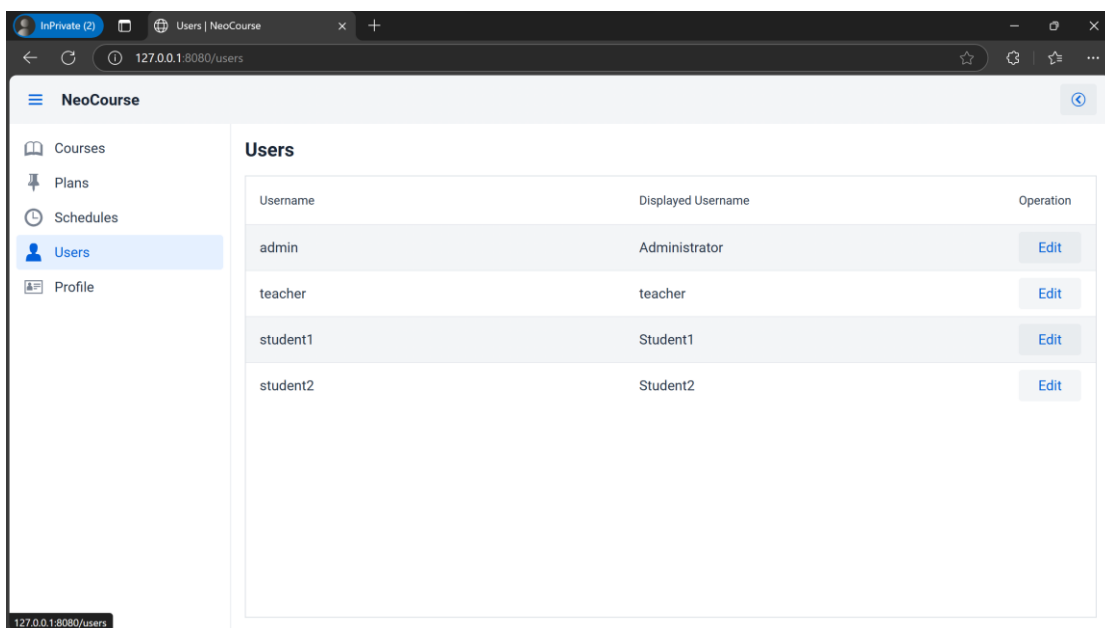
Old Password *

New Password *

Confirm Password *

[Cancel](#) [Apply](#)

- 管理员 Administrator 管理用户 User：



- 管理员 Administrator 的授权

具有管理员 Administrator 职责的用户可以选择将某个具有老师 Teacher 职责的用户升级为管理员 Administrator，实为为其添加管理员 Administrator 职责。

系统可以通过三个环境变量初始化系统默认管理员的相关信息：

- NEOCOURSE_ADMINISTRATOR_USERNAME：管理员的用户名
- NEOCOURSE_ADMINISTRATOR_DISPLAYED_USERNAME：管理员的显示用户名
- NEOCOURSE_ADMINISTRATOR_PASSWORD：管理员的密码（明文）

3.3 重要业务规则简述

- 计划 Plan 包括课程 Course 和必修学时，其必须不大于所有已包括课程 Course 的学时总和。
- 学生 Student 选择安排 Schedule 时，应满足安排 Schedule 对应的课程 Course 包括于自己的计划 Plan 中，且没有已经选择过此课程 Course。
- 安排 Schedule 创建时，应满足在同一地点不于其他已有 Schedule 冲突。
- 删除 Course 时，应满足当前不存在对应的安排 Schedule。
- 删除安排 Schedule 时，应当满足没有学生 Student 注册此安排 Schedule，并且当前时间早于安排 Schedule 的开始时间。
- 学生 Student 选择取消安排 Schedule 时，应当满足当前时间早于安排 Schedule 的开始时间，否则无法取消。
- 老师 Teacher 为学生 Student 在某次安排 Schedule 中评分时，应当满足当前时间晚于安排 Schedule 的开始时间，即不可以上课之前就评分。

4 程序算法说明及面向对象实现技术方案

4.1 层次结构

程序基于领域驱动设计思想进行开发，程序分为四个层级——接口层、应用层、领域层、基础设施层，分别位于项目的四个包中。

领域层是程序的核心，包含各种模型和领域服务，用于表示网站的核心业务逻辑。

应用层是对领域层业务逻辑的简单编排，对外部接口提供一个简单统一的门面。应用

层中主要包含各种应用服务，对应系统的各个用例。应用服务采用 CQRS 进行设计，对于各种增加、修改、删除等写操作，调用领域层模型进行完成，而对于各种查询操作，则直接从基础设施层中读取，分离读写两条链路，保证领域层的简洁和稳定。

基础设施层是对程序基础功能的支持，包括对数据库访问的包装。基础设施层实现领域层的接口从而实现基础功能，同时不与其产生强耦合，具有较高扩展性。

接口层是用户访问系统的入口，在本系统中为 Web 前端。接口层处理用户交互和数据的转换，调用应用层。

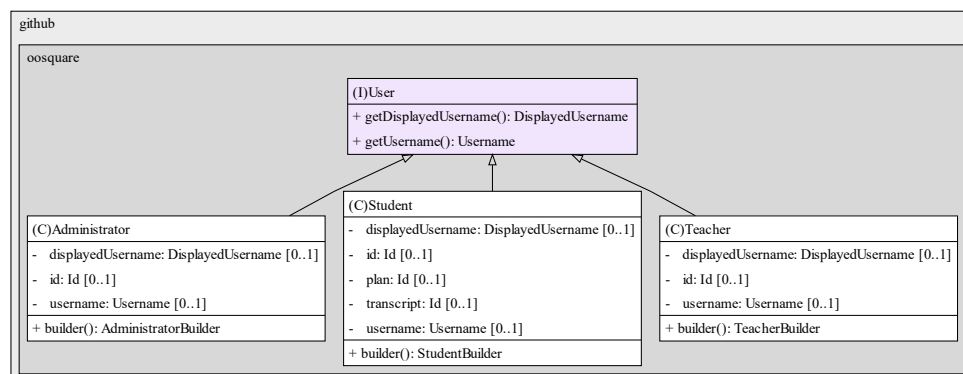
4.2 领域模型设计

领域层中有多个领域实体和相关的值对象。

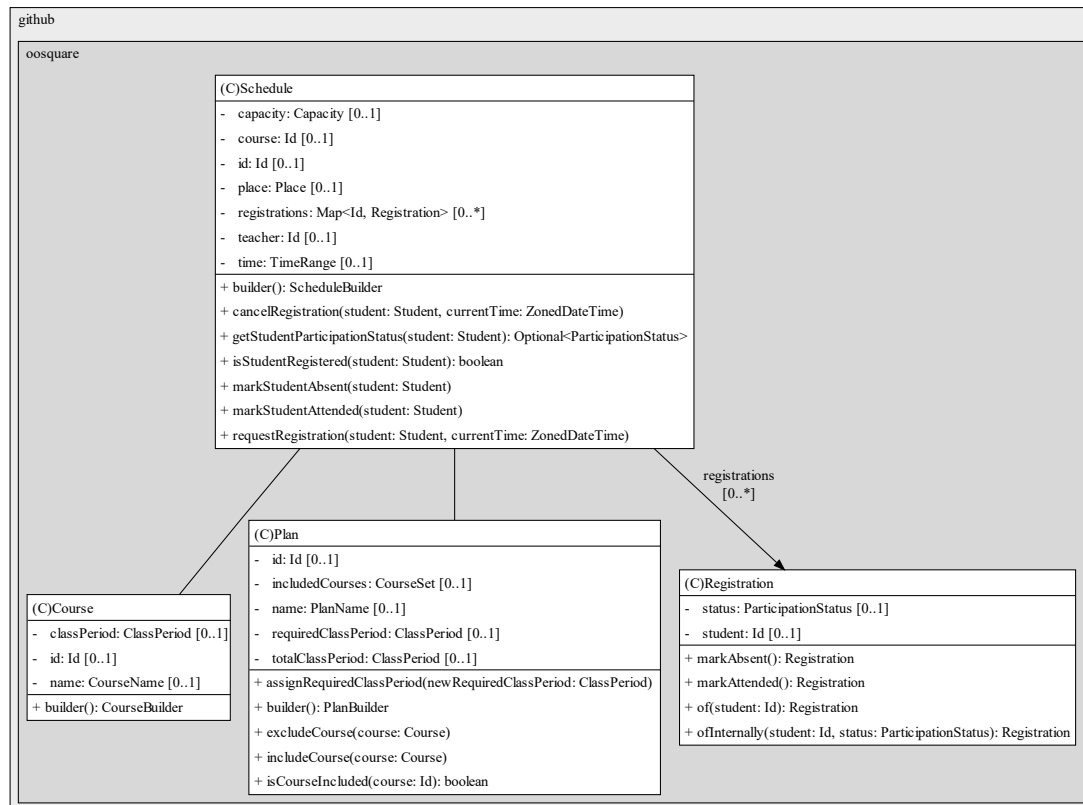
重要的领域实体有：

- **Account**：系统中的用户用于认证、授权的基础信息，并与各职责关联。
- **Administrator**：代表管理员职责的相关领域概念的集合。
- **Student**：代表学生职责的相关领域概念的集合。
- **Teacher**：代表老师职责的相关领域概念的集合。
- **Course**：一门可供老师 Teacher 教授的课程。
- **Plan**：学生 Student 应选择学习的课程 Course 的集合，并包含相关评估要求。
- **Schedule**：课程 Course 的具体安排，并管理与此次安排有关的各种信息。
- **Transcript**：学生 Student 的成绩单信息。

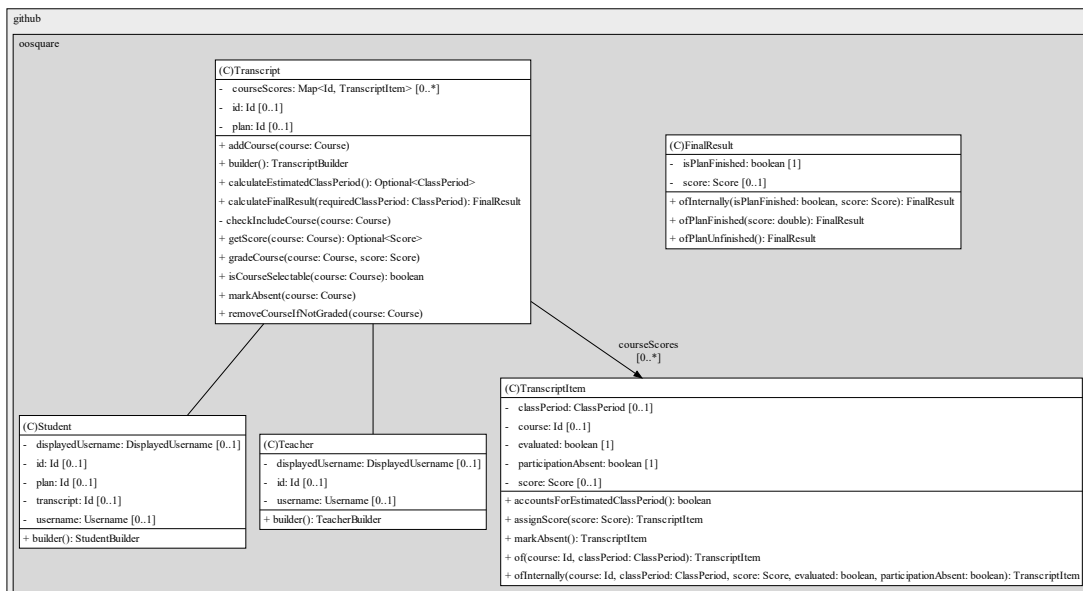
每个实体都具有较多的值对象，分别表示一些最基础的概念，在此不列出。



用户 User 是一个接口，代表抽象的用户，可以通过其获取所有 User 的通用信息。Administrator、Student、Teacher 的含义在上文以及介绍。



上图中，每一个安排 **Schedule** 都与课程 **Course** 关联，学生 **Student** 可以通过选择一个具体的安排 **Schedule** 来选课程 **Course**，安排 **Schedule** 中会记录学生 **Student** 的报名情况 **Registration**。学生 **Student** 需要在安排 **Schedule** 中指定的时间和地点参加。



上图中，成绩单 **Transcript** 与学生 **Student** 一一对应，老师 **Teacher** 则可以通过成绩单 **Transcript** 为学生 **Student** 打分。成绩单 **Transcript** 有成绩单项 **TranscriptItem**，分别记录课程的参与情况与分数。

对于每一个领域实体，纵观其在系统中的生命周期，可以分为创建、复原、修改、持久化、删除等多个过程。

对于创建过程，一个 **Entity** 实体则对应 **EntityFactory**，其接受各种必要的创建参数，进

行相关的逻辑校验，最终构建出一个完整的对象。

对于持久化、复原、删除过程，一个是聚合根的 `Entity` 实体则对应 `EntityRepository` 接口，完成聚合根内实体及值对象的持久化和复原。在基础设施层中，使用对应的实现类，完成实际工作。

对于与业务逻辑有关的修改过程，通过使用领域模型自身的方法完成，或者通过一些特殊的跨实体领域服务完成，本系统中典型的领域服务有：

- `ScheduleService`：完成安排 `Schedule` 的创建和删除的全过程。
- `RegistrationService`：完成学生 `Student` 注册和取消注册安排 `Schedule`。
- `EvaluationService`：完成老师 `Teacher` 对学生 `Student` 的评分。

5 技术亮点、关键点及其解决方案

- 本程序的亮点：
 - 简洁、现代的 UI，并且实现完备的功能。
 - 具有完全的移动端支持，网站完全使用响应式设计，兼容各种设备。
 - 友好和完整的错误提示，对各种异常状况进行了充分的考虑和准备。
- 本程序的技术关键点：
 - 整体设计完全基于 `DDD`，领域模型作为项目的核心部分，对整个选课系统的设计进行了丰富饱满的建模，而非简单的 `CRUD`。
 - 层次清晰，对象间低耦合，在跨层调用中，使用 `DTO` 和对应的转化对象，实现层之间的隔离。
 - 将所有领域相关的概念设计为单独的实体或值对象，并加上相关的业务逻辑，提高内聚性，使得代码清晰整洁。
 - 在开发过程中运用测试驱动开发，编写完善的测试支持，对领域层进行全面的单元测试，对应用层和基础设施层进行关键运行路径的集成测试，并使用 `GitHub Action` 进行持续集成。
 - 使用 `JPA` 简化数据访问，同时实现跨 `SQL` 数据库的支持，目前在测试时使用 `H2` 数据库，生产环境中使用 `SQLite` 数据库。
 - 使用读写分离，避免修改和读取两种不同用途的代码混杂，既保证修改操作的严谨和正确性，又简化读取操作、提高性能。
 - 使用 `Vaadin Flow` 作为 `Web` 前端开发工具，实现用 `Java` 代码制作前端，简化开发。
- 遇到的技术难点及对应的解决方案：
 - 集成测试时使用 `@DataJpaTest` 无法正常运行
问题描述：为集成测试类加上 `@DataJpaTest` 注解，但报错找不到 `bean`
最终的解决方案：`@DataJpaTest` 的测试中默认只实例化 `Spring Data` 提供的 `Repository` 接口的实现类。由于本项目不使用 `Spring Data` 的接口，所以 `Spring` 不会在此时扫描自行编写的相关组件。通过 `@Import` 注解包括相关类即可解决。
 - 抛出 `LazyInitializationException` 异常
问题描述：使用 `JPA`，可能抛出 `LazyInitializationException` 异常
最终的解决方案：`LazyInitializationException` 是由于 `JPA` 读取的实体具有懒加载机制，若超出事务边界，则会抛出此异常。可以通过指定立即加载或编写特定的投影对象直接用于查询避免多次加载，本项目主要采用后者。

6 简要开发过程

2024-10-27 init: initialize repository
2024-10-27 feat(utility): add ID and ID generator
2024-10-27 feat(domain): add Student domain model
2024-10-28 feat(domain): add validation for value objects' constructors
2024-10-28 feat(domain): add Teacher domain model
2024-10-28 refactor(domain): change directory structure
2024-10-28 refactor(domain): change value objects' field names
2024-10-28 feat(domain): add Course domain model
2024-10-28 feat(domain): add Plan domain model
2024-10-29 refactor(domain): use Guava's immutable collections in VOs
2024-10-29 refactor(domain): replace entities' IDs with refs in parameters
2024-10-29 refactor(domain): add base exception class for domain
2024-10-30 feat(domain): add Schedule domain model
2024-10-30 feat(domain): add score calculation for Student
2024-10-30 refactor(domain): make all entities implement Entity
2024-10-30 feat(domain): add base interface of repositories
2024-10-31 feat(domain): add factory methods for usage in repositories
2024-10-31 feat(domain): add factory methods for usage in repositories
2024-10-31 feat(domain): add factory and repository for Course
2024-10-31 refactor(utility): replace constructor with factory method
2024-10-31 refactor(domain): refine VOs' toString() format
2024-10-31 feat(domain): add factory and repository for Plan
2024-10-31 refactor(domain): extract Transcript aggregate
2024-10-31 feat(domain): add factory and repository for Student
2024-10-31 feat(domain): add factory and repository for Teacher and Transcript
2024-11-01 refactor(utility): simplify ID generating utility
2024-11-01 refactor(domain): extract TimeRange VO
2024-11-01 feat(domain): add factory and repository for Schedule
2024-11-01 ci: add automatic testing workflow
2024-11-01 chore: add gradle-wrapper.jar
2024-11-01 ci: split workflow
2024-11-02 feat(domain): add Administrator aggregate root
2024-11-02 feat(domain): add schedule registration service
2024-11-03 refactor(domain): extract TranscriptItem
2024-11-03 feat(domain): add check to prevent duplicated registrations
2024-11-03 refactor(domain): refine course evaluation and registraion
2024-11-04 refactor(domain): improve Teacher managing Schedule
2024-11-04 feat(domain): add evaluation service
2024-11-04 refactor(domain): remove redundant check in Schedule
2024-11-04 feat(domain): add schedule add/remove service
2024-11-04 refactor(domain): simplify Plan repository
2024-11-05 feat(domain): add course add/remove service

2024-11-06 feat(infrastructure): add Administrator repository
2024-11-07 refactor(domain): replace factory methods with builders
2024-11-07 feat(infrastructure): add Student repository
2024-11-07 feat(infrastructure): add Teacher repository
2024-11-07 feat(infrastructure): add Course repository
2024-11-08 feat(infrastructure): add Plan repository
2024-11-08 refactor(infrastructure): remove DO's toString(), equals() & hashCode()
2024-11-08 refactor(infrastructure): improve handling single result queries
2024-11-08 refactor(infrastructure): change database table names
2024-11-08 feat(infrastructure): add Schedule repository
2024-11-09 feat(infrastructure): add Transcript repository
2024-11-14 feat(domain): add Account and its repository
2024-11-14 feat(domain): add AccountFactory
2024-11-14 feat(infrastructure): add Account repository
2024-11-19 feat(application): add Course application services
2024-11-19 chore: add vaadin dependencies
2024-11-19 refactor(utility, domain): add base and validation exception
2024-11-19 refactor(utility, domain): add exception for unreachable code
2024-11-19 refactor(utility, domain): add exception for entity not found
2024-11-20 refactor(utility, domain): add exception for entity's field duplication
2024-11-20 refactor(utility, domain): partially use business rule exception
2024-11-20 refactor(domain): simplify Teacher managing Course
2024-11-20 refactor(utility, domain): use business rule exception
2024-11-21 refactor(application): change CourseCommandService's parameters
2024-11-21 refactor(utility, domain): use business rule exception
2024-11-21 feat(application): add Plan application services
2024-11-21 feat(*): add requiredClassPeriod field for Plan
2024-11-21 feat(application): add account parameter for query services
2024-11-21 feat(ui): add course UI prototype
2024-11-21 feat(ui): add UI layout
2024-11-22 feat(application): add Schedule command service
2024-11-22 refactor(infrastructure): make RegistrationData an entity
2024-11-22 feat(application): add Schedule query service
2024-11-22 feat(domain): add totalClassPeriod for Plan
2024-11-22 feat(application): add assignRequireClassPeriod command
2024-11-24 feat(ui): add plan view prototype
2024-11-25 refactor(ui): use dialogs to manage courses
2024-11-25 refactor(application): change plan queries' returning DTOs
2024-11-25 refactor(application): simplify query services' parameters
2024-11-25 refactor(ui): use dialogs in plan management
2024-11-25 feat(ui): finish plan view
2024-11-25 feat(ui): finish schedule view
2024-11-26 feat(ui): add index view
2024-11-26 feat(application): add registration command service

2024-11-26 feat(application): add query by student for schedules
2024-11-26 feat(ui): add registration view
2024-11-26 feat(application): add evaluation command service
2024-11-26 refactor(ui): extract dialog super class
2024-11-26 refactor(infrastructure): make TranscriptItemData an entity
2024-11-26 feat(application): change schedule's query representation
2024-11-26 chore: set up integration test task
2024-11-27 feat(application): add ITs for course and plan command services
2024-11-27 refactor(domain): make some all-args constructors private
2024-11-27 feat(application): add ITs for schedule command service
2024-11-28 feat(application): add ITs for registration and evaluation command services
2024-11-28 fix(infrastructure): dynamic instantiation in sub-query
2024-11-28 feat(ui): add evaluation view
2024-11-28 feat(*): add secure functionality
2024-11-28 feat(application): add user command service
2024-11-29 feat(ui): add signing up functionality
2024-11-29 feat(*): add default user initialization in development mode
2024-11-29 feat(ui, application): read user info from security context
2024-11-29 feat(ui): add log out capability
2024-11-29 feat(ui): enable auto navigation to login after signing up
2024-11-29 feat(domain): make schedule not evaluable before it starts
2024-11-30 feat(ui): show current user and adjust log in & sign up view layout
2024-11-30 fix(application): integration test failure
2024-11-30 refactor(domain): make user management role-based
2024-11-30 refactor(infrastructure): change account schema
2024-11-30 refactor(application): make Teacher manage Plan and Course
2024-11-30 feat(ui): hide navigation items if no permission
2024-11-30 refactor(application, infrastructure): rename queries
2024-11-30 feat(ui): query registrations of one account
2024-12-01 refactor(ui): change welcome message placement
2024-12-01 feat(ui): add transcript view
2024-12-01 refactor(domain): rename user service
2024-12-01 feat(ui): add page titles
2024-12-01 fix(domain): wrong transcript result if no class selected
2024-12-05 feat(domain): support add multiple roles to an account
2024-12-12 feat(command): add upgrading account to Administrator
2024-12-12 feat(application, infrastructure): add Account queries
2024-12-12 feat(ui): add users and user edit view
2024-12-12 feat(ui): add upgrading user to administrator
2024-12-13 feat(application): add changing password
2024-12-13 feat(ui): add changing password dialog
2024-12-14 refactor(application): remove unit tests
2024-12-15 chore: add SQLite config
2024-12-28 refactor(ui): improve plan view

2024-12-28 chore: lock dependencies

2024-12-28 feat: add readme and license

2024-12-28 fix: no exec permission for gradlew

2024-12-28 docs: add dev docs

2024-12-28 chore: change version to 0.1.0 and publish release

7 个人小结

经过一个学期的学习，我对 **Java** 与其丰富的生态有了进一步的了解，学习软件工程的知识和思想，并设计尝试一个具有一定规模的可维护的软件架构和系统。本项目经过实际近 5 周的快速迭代开发，不断进行测试、重构，从简单的原型不断向一个完整的系统演化，在此过程中，我对软件开发的各方面的理解有了较大的进步，逐渐形成一个整体图像。