

Team Report

成员：朱柯奇、朱育辰、梁煜、林易成、孙杨洋

Metrics (项目指标)

我们对项目的代码结构与复杂度进行了详细统计与评估, 主要包括以下几个方面:

代码总行数: 反映了项目的整体规模与开发工作量。

源文件数量: 展示了项目的模块化与工程结构。

圈复杂度 (Cyclomatic Complexity): 衡量了代码的逻辑复杂度, 有助于评估代码的可维护性与潜在的错误风险。

依赖数量: 评估了项目中各个模块之间的依赖关系, 越少的依赖有助于提高模块的独立性与可重用性。

所有这些数据的详细统计信息已记录在以下日志文件中, 可供进一步分析参考:

<https://github.com/sustech-cs304/team-project-25spring-15/blob/main/evaluate.log>

Documentation (项目文档)

用户文档

我们为普通用户准备了一份详细的使用文档, 内容包括:

项目的开发背景与目标

系统的基本功能与使用流程

系统界面与操作指南

通过该文档，用户可以快速了解并上手使用我们的系统。

用户文档链接：

开发者文档

为方便后续开发与社区贡献，我们还编写了 API 文档，内容包括：

每个接口的 URL、请求方法、请求参数与响应格式开发者可通过该文档快速了解后端 API 的设计，进行前后端联调或继续开发。

开发者文档链接：

<https://github.com/sustech-cs304/team-project-25spring-15/blob/main/%E6%99%BA%E8%83%BD%E8%AF%BE%E7%A8%8B%E6%84%9F%E7%9F%A5IDE%20-%20%E5%BC%80%E5%8F%91%E8%80%85%E6%96%87%E6%A1%A3.md>

<http://47.117.144.50:8000/swagger>

Test（测试体系）

我们采用 GoFrame 框架内置的 `gtest` 模块实现自动化测试。`gtest` 是一个轻量级但功能强大的 Go 测试工具，能够与控制器、服务层及数据库模块深度集成，适用于单元测试与集成测试场景。

测试代码结构与实现

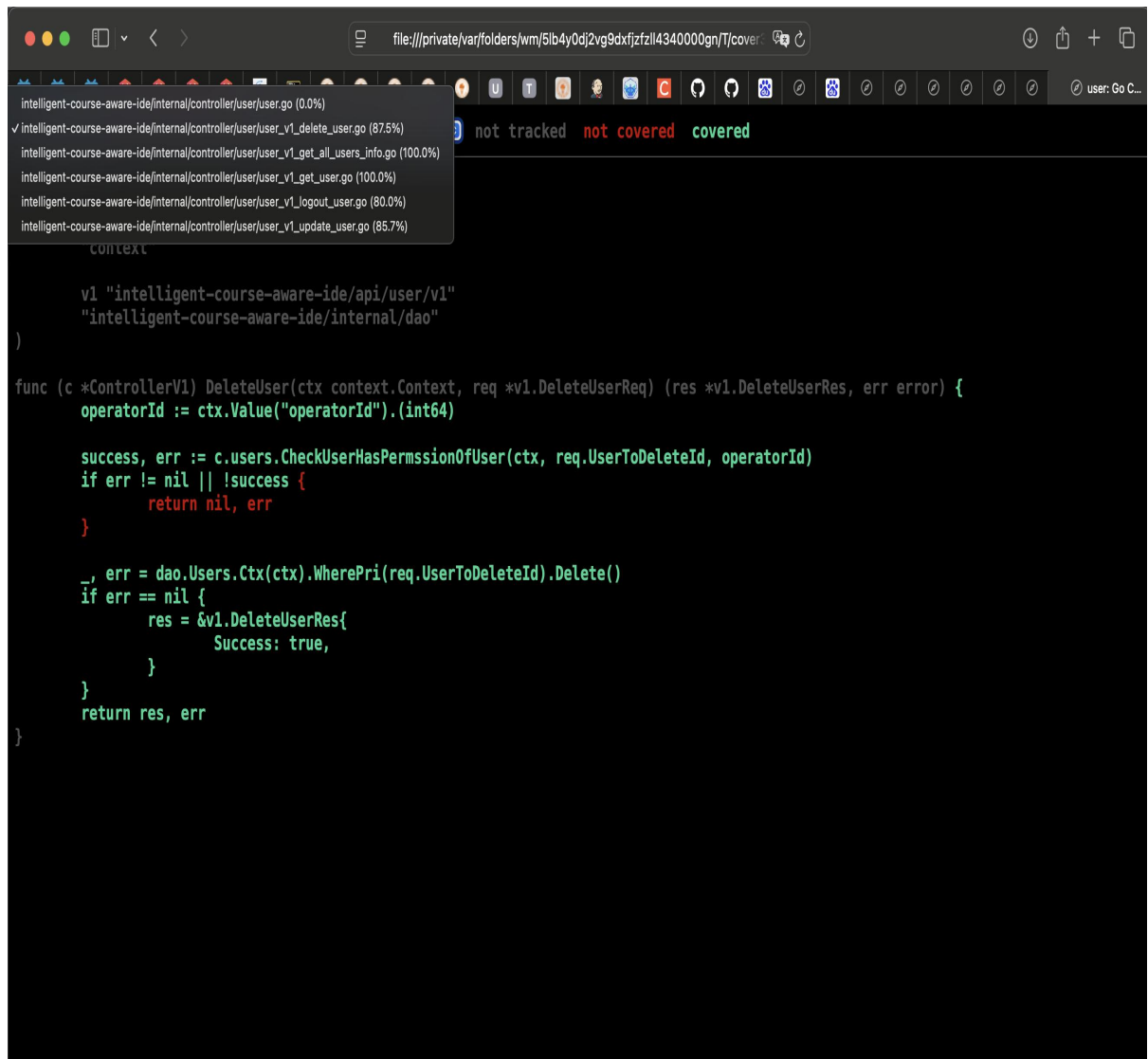
所有测试代码按照模块划分，分别放置于 `controller` 层每个 API 所在目录中。

测试用例以 `gtest.C` 结构体进行组织，并结合 `gtest.Assert`, `gtest.AssertEQ`, `gtest.AssertNE` 等断言函数完成逻辑验证。

测试源码地址:

<https://github.com/sustech-cs304/team-project-25spring-15/tree/main/Backend/intelligent-course-aware-ide/internal/controller>

我们对核心业务逻辑模块实现了超过 70% 的代码覆盖率, 测试统计使用 Go 自带的 `go test -cover` 工具完成。



The screenshot shows a Go IDE with a file explorer on the left displaying a list of files and their coverage percentages. The main editor shows a Go function `DeleteUser` with its implementation. The status bar at the bottom indicates the current line is 'not tracked', 'not covered', and 'covered'.

```
intelligent-course-aware-ide/internal/controller/user/user.go (0.0%)
✓ intelligent-course-aware-ide/internal/controller/user/user_v1_delete_user.go (87.5%)
intelligent-course-aware-ide/internal/controller/user/user_v1_get_all_users_info.go (100.0%)
intelligent-course-aware-ide/internal/controller/user/user_v1_get_user.go (100.0%)
intelligent-course-aware-ide/internal/controller/user/user_v1_logout_user.go (80.0%)
intelligent-course-aware-ide/internal/controller/user/user_v1_update_user.go (86.7%)

CONTEXT

v1 "intelligent-course-aware-ide/api/user/v1"
"intelligent-course-aware-ide/internal/dao"
)

func (c *ControllerV1) DeleteUser(ctx context.Context, req *v1.DeleteUserReq) (res *v1.DeleteUserRes, err error) {
    operatorId := ctx.Value("operatorId").(int64)

    success, err := c.users.CheckUserHasPermssionOfUser(ctx, req.UserToDeleteId, operatorId)
    if err != nil || !success {
        return nil, err
    }

    _, err = dao.Users.Ctx(ctx).WherePri(req.UserToDeleteId).Delete()
    if err == nil {
        res = &v1.DeleteUserRes{
            Success: true,
        }
    }
    return res, err
}
```

测试重点包括:

用户注册、登录及权限控制等核心认证流程

课程管理、讲座发布与报名、练习创建与提交等主要业务模块

系统在异常与边界条件下的稳定性保障

通过系统化测试，我们有效保障了项目在多种场景下的正确性和健壮性。

Build (构建流程)

为了实现项目的自动化构建，我们使用了 **GNU Make** 工具编写了构建脚本

(**Makefile**)，自动完成代码编译、依赖安装、复杂度分析等任务，提升了项目开发和部署的效率与规范性。

使用的构建工具：

GNU Make

Go Modules (用于依赖管理)

Python venv (用于创建分析环境)

构建过程中执行的任务包括：

启动后端开发容器环境；

创建并配置 **Python** 虚拟环境；

安装项目依赖工具（如 **cloc**、**gocyclo**、**lizard** 等）；

后端和前端的代码行数统计、圈复杂度分析、依赖数量计算等静态分析任务。

构建产物：

后端编译生成的可执行文件 **main**

自动分析生成的结果输出（如代码复杂度、文件数量等）

构建脚本文件：

主构建脚本为根目录下的 **Makefile**，包括开发环境启动、分析环境搭建和分析任务执行等部分。

通过该自动化构建流程，我们能够快速、稳定地完成代码构建及质量分析，确保项目具备良好的可维护性与可部署性。

Deployment (部署流程)

为了使我们的软件系统能够顺利对外提供服务，我们采用了现代化的容器化部署方式，使用了 **Docker** 技术对系统进行了封装和管理。整个项目被拆分为三个主要服务进行容器化：后端服务（**app**）、代码运行服务（**runner**）以及数据库服务（**database**）。我们为每个服务分别编写了对应的 **Dockerfile**，并通过 **Docker Compose** 和 **Docker Swarm** 实现编排与部署。

使用的技术/工具:

Docker

Docker Compose

Docker Swarm

Golang 1.23

MySQL 8.0

Python 3 (用于 runner 服务)

容器化相关文件:

各服务的 **Dockerfile** 位于 **Dockerfiles/** 目录下。

链接:

<https://github.com/sustech-cs304/team-project-25spring-15/tree/main/Backend/Dockerfiles>

使用 **docker-compose.yml** 文件管理开发环境下的服务运行。

链接:

<https://github.com/sustech-cs304/team-project-25spring-15/blob/main/Backend/docker-compose.dev.yml>

容器化成功的证明:

所有容器均可通过 `docker-compose up` 和 `docker stack deploy` 命令成功构建并运行。

数据和日志成功通过挂载卷持久化。