**“教师教学科研登记系统”**

系统设计与实现报告

**姓名：杨映川**

**学号：PB22111599**

**计算机科学与技术学院**

**中国科学技术大学**

2025年6月

**目 录**

1 需求分析 1

1.1 系统目标 1

1.2 需求说明 1

2 总体设计 1

2.1 系统模块结构 1

2.2 系统工作流程 1

2.3 数据库设计 1

3 详细设计【可选】 1

3.1 主界面模块 2

3.2 教师登录模块 2

3.3 教师管理模块 2

3.4 论文管理模块 2

3.5 课程管理模块 2

3.6 项目管理模块 2

3.7 查询统计模块 2

3.8 导出报告模块 2

4 实现与测试 2

4.1 实现结果 2

4.2 测试结果 2

4.3 实现中的难点问题及解决【可选】 2

5 总结与讨论 2

# 

# 1 需求分析

## 1.1 系统目标

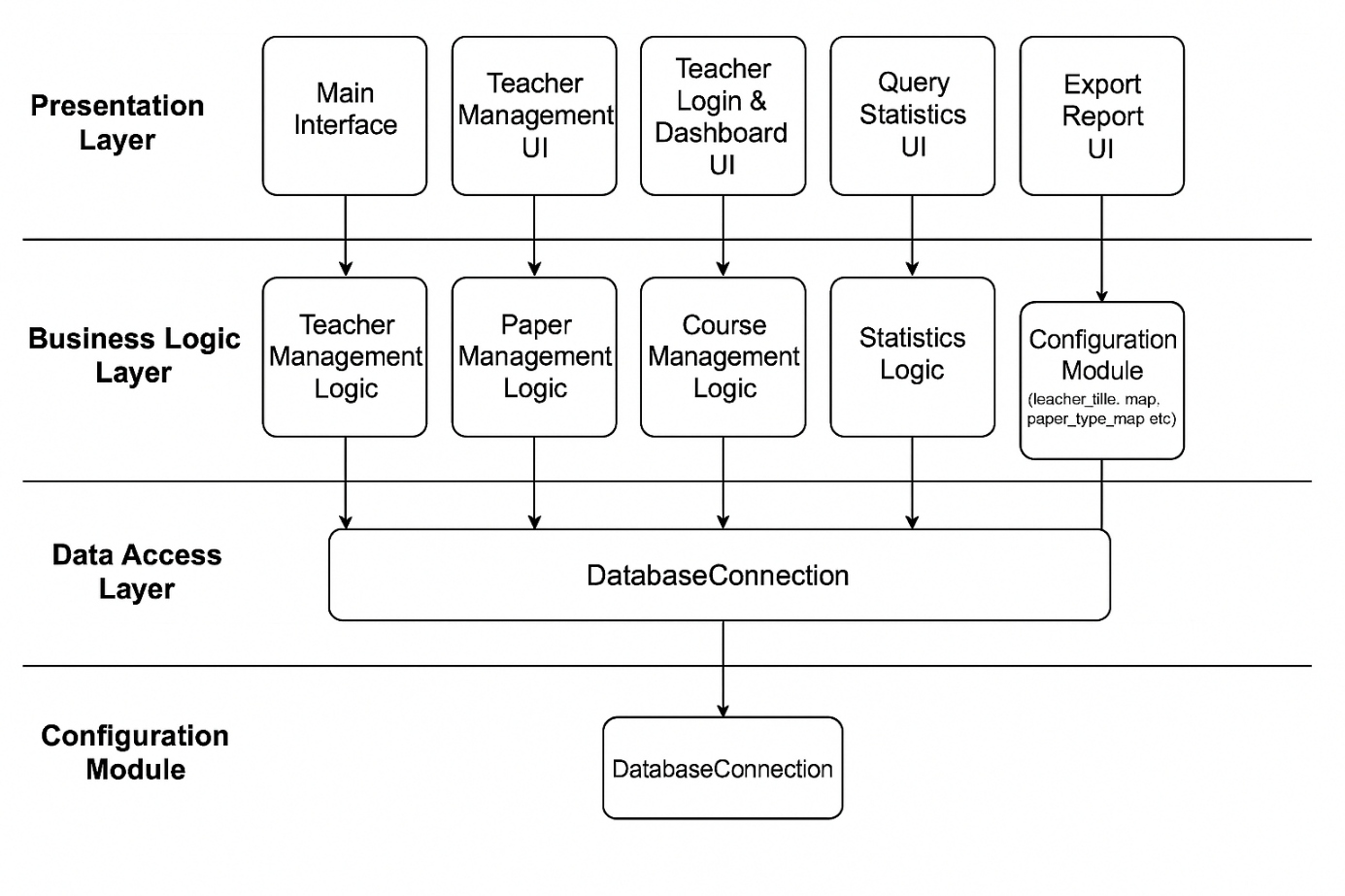
开发一个面向教师的教学科研登记系统，以教师个人为面向管理论文发表、课程教授、项目承担情况。

## 1.2 需求说明

1. 登记发表论文情况：提供教师论文发表信息的的增、删、改、查功能；输入时要求检查：一篇论文只能有一位通讯作者，论文的作者排名不能有重复，论文的类型和级别只能在约定的取值集合中选取（实现时建议用下拉框）。
2. 登记承担项目情况：提供教师承担项目信息的增、删、改、查功能；输入时要求检查：排名不能有重复，一个项目中所有教师的承担经费总额应等于项目的总经费，项目类型 只能在约定的取值集合中选取。
3. 登记主讲课程情况：提供教师主讲课程信息的增、删、改、查功能；输入时要求检查：一门课程所有教师的主讲学时总额应等于课程的总学时，学期。
4. 查询统计：
   1. 实现按教师工号和给定年份范围汇总查询该教师的教学科研情况的功能；例如输入工号“01234”，“2023-2023”可以查询01234教师在2023年度的教学科研工作情况。
   2. \*（选做）实现按教师工号和给定年份范围生成教学科研工作量统计表并导出文档的功能，导出文档格式可以是PDF、Word、Excel等。

# 2 总体设计

## 2.1 系统模块结构

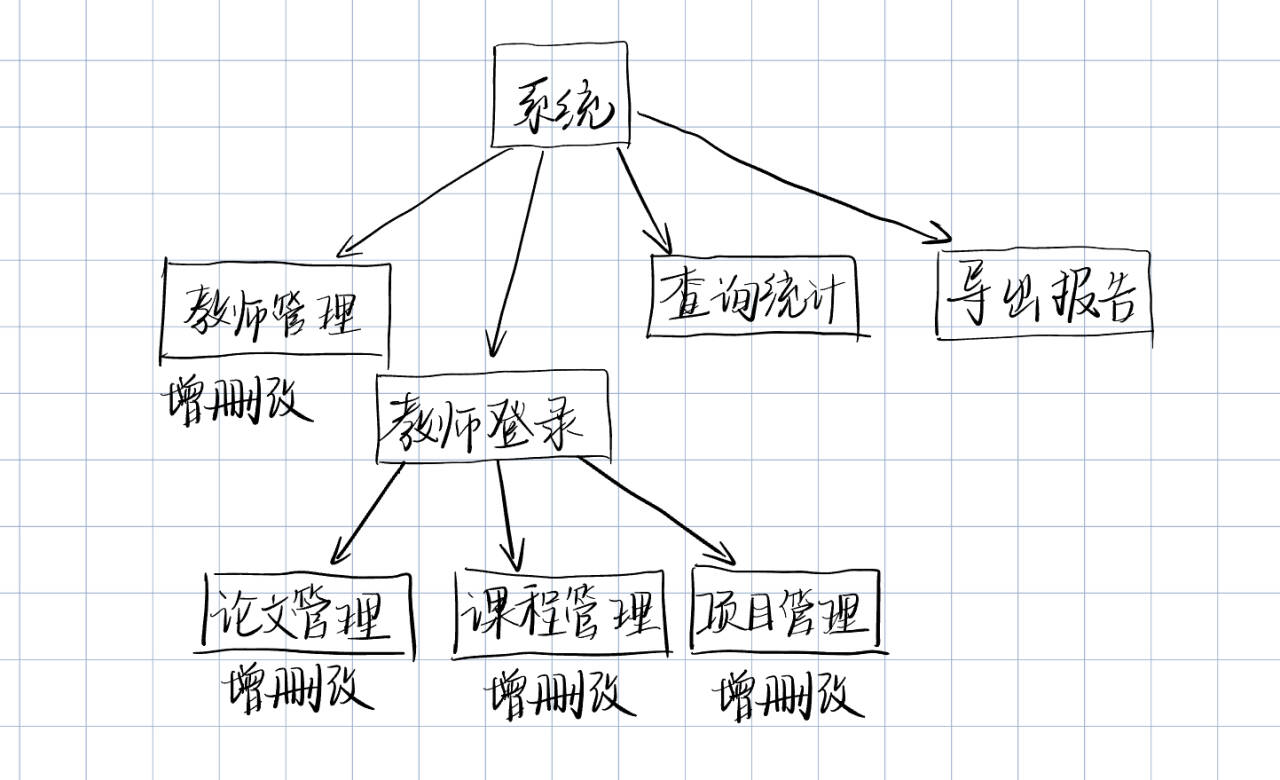


系统采用典型的四层架构，按职责划分为：

1. Presentation Layer（表现层）
   * 1. Main Interface
     2. Teacher Management UI
     3. Teacher Login & Dashboard UI
     4. Query Statistics UI
     5. Export Report UI
2. Business Logic Layer（业务逻辑层）
   * 1. Teacher Management Logic
     2. Paper Management Logic
     3. Course Management Logic
     4. Project Management Logic
     5. Statistics Logic
     6. Reporting Logic
3. Data Access Layer（数据访问层）
   * 1. DatabaseConnection
4. Configuration Module（配置模块）
   * 1. Mapping Tables

## 2.2 系统工作流程

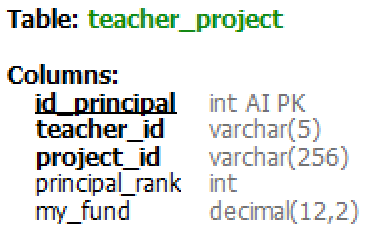
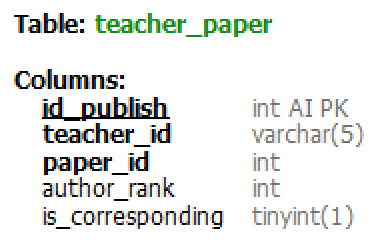
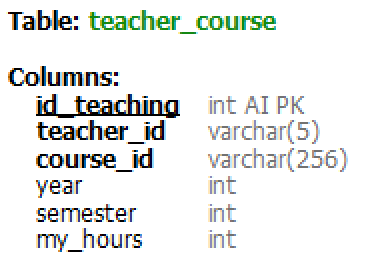
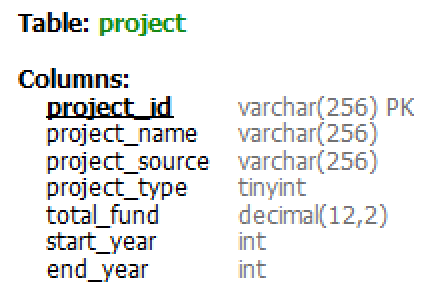
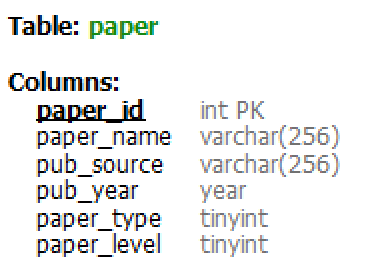
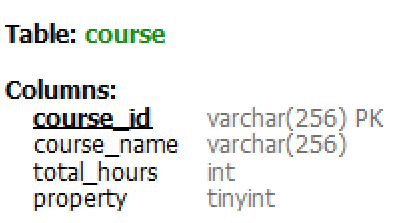
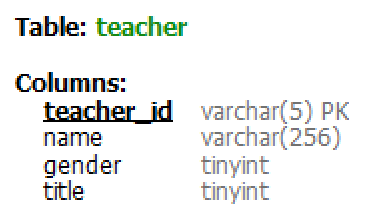
// 给出系统工作流程图



## 2.3 数据库设计

# 

ER图

**物理模型**

# 3 详细设计【可选】

// 一个模块一个小节论述

## 3.1 主页面模块

*def* create\_main\_interface(*self*):

        frame = tk.Frame(*self*.root)

        frame.pack(*expand*=True)

        btn\_add = tk.Button(frame, *text*="教师管理", *width*=20, *height*=2, *command*=*self*.teacher\_management)

        btn\_login = tk.Button(frame, *text*="教师登录", *width*=20, *height*=2, *command*=*self*.teacher\_login)

        btn\_query = tk.Button(frame, *text*="查询统计", *width*=20, *height*=2, *command*=*self*.query\_stats)

        btn\_export = tk.Button(frame, *text*="导出报告", *width*=20, *height*=2, *command*=*self*.export\_report)

        btn\_add.grid(*row*=0, *column*=0, *padx*=20, *pady*=20)

        btn\_login.grid(*row*=0, *column*=1, *padx*=20, *pady*=20)

        btn\_query.grid(*row*=1, *column*=0, *padx*=20, *pady*=20)

        btn\_export.grid(*row*=1, *column*=1, *padx*=20, *pady*=20)

## 3.2 教师登录模块

*def* teacher\_login(*self*):

        win = tk.Toplevel(*self*.root)

        win.title("教师登录")

        win.geometry("300x150")

        tk.Label(win, *text*="工号:").grid(*row*=0, *column*=0, *padx*=10, *pady*=10)

        entry\_id = tk.Entry(win)

        entry\_id.grid(*row*=0, *column*=1, *padx*=10, *pady*=10)

*def* login():

            tid = entry\_id.get().strip()

            if not tid:

                messagebox.showerror("错误", "请输入工号")

                return

            info = *self*.db.execute\_query(

                "SELECT name, CASE title WHEN 1 THEN '博士后' WHEN 2 THEN '助教' WHEN 3 THEN '讲师' WHEN 4 THEN '副教授' WHEN 5 THEN '特任教授' WHEN 6 THEN '教授' WHEN 7 THEN '助理研究员' WHEN 8 THEN '特任副研究员' WHEN 9 THEN '副研究员' WHEN 10 THEN '特任研究员' WHEN 11 THEN '研究员' END as title FROM Teacher WHERE teacher\_id=%s", (tid,))

            if not info:

                messagebox.showerror("错误", "教师不存在")

                return

            name, title = info[0]

            win.destroy()

*self*.open\_teacher\_dashboard(tid, name, title)

        tk.Button(win, *text*="登录", *command*=login).grid(*row*=1, *column*=0, *columnspan*=2, *pady*=10)

## 3.3 教师管理模块

*def* teacher\_management(*self*):

        win = tk.Toplevel(*self*.root)

        win.title("教师管理")

        win.geometry("400x600")

        top\_frame = tk.Frame(win)

        top\_frame.pack(*pady*=10)

        tk.Button(win, *text*="添加教师", *command*=*self*.add\_teacher).pack(*pady*=20)

        # ? frame用于放置教师列表

        list\_frame = tk.Frame(win)

        # ? fill=tk.BOTH, expand=True表示填充整个窗口并允许扩展

        list\_frame.pack(*fill*=tk.BOTH, *expand*=True)

        # ? 创建一个Canvas用于显示教师列表

        canvas = tk.Canvas(list\_frame)

        # ? 创建一个垂直滚动条

        scrollbar = tk.Scrollbar(list\_frame, *orient*="vertical", *command*=canvas.yview)

        # ? 创建一个Frame用于放置教师列表内容

        scroll\_frame = tk.Frame(canvas)

        # ? 绑定Frame的配置事件，以便Canvas可以正确计算滚动区域

        scroll\_frame.bind(

            "<Configure>", *lambda* *e*: canvas.configure(*scrollregion*=canvas.bbox("all"))

        )

        # ? 在Canvas中创建一个窗口，将scroll\_frame放入其中

        canvas.create\_window((0, 0), *window*=scroll\_frame, *anchor*="nw")

        # ? 设置Canvas的滚动条

        canvas.configure(*yscrollcommand*=scrollbar.set)

        # ? 将Canvas和滚动条放置在list\_frame中

        canvas.pack(*side*=tk.LEFT, *fill*=tk.BOTH, *expand*=True)

        # ? 将滚动条放置在list\_frame的右侧

        scrollbar.pack(*side*=tk.RIGHT, *fill*=tk.Y)

        teachers = *self*.db.execute\_query(

            "SELECT \* FROM Teacher ORDER BY teacher\_id ASC"

        )

        for teacher\_id, name, gender, title in teachers:

            frame\_row = tk.Frame(scroll\_frame, *bd*=1, *relief*="solid")

            frame\_row.pack(*fill*=tk.X, *pady*=2, *padx*=2)

            title\_str = teacher\_title\_map.get(title, "未知职称")

            info = *f*"{teacher\_id} | {name} | {'男' if gender == 1 else '女'} | 职称:{title\_str}"

            tk.Label(frame\_row, *text*=info).pack(*side*=tk.LEFT, *padx*=5, *fill*=tk.X)

            tk.Button(frame\_row, *text*="修改", *command*=*lambda* *tid*=teacher\_id: *self*.modify\_teacher(*tid*, win)).pack(*side*=tk.RIGHT, *padx*=5)

            tk.Button(frame\_row, *text*="删除", *command*=*lambda* *tid*=teacher\_id: *self*.delete\_teacher(*tid*)).pack(*side*=tk.RIGHT)

*def* add\_teacher(*self*):

        win = tk.Toplevel(*self*.root)

        win.title("添加教师")

        win.geometry("400x300")

        tk.Label(win, *text*="工号:").grid(*row*=0, *column*=0, *padx*=10, *pady*=5)

        entry\_id = tk.Entry(win)

        entry\_id.grid(*row*=0, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="姓名:").grid(*row*=1, *column*=0, *padx*=10, *pady*=5)

        entry\_name = tk.Entry(win)

        entry\_name.grid(*row*=1, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="性别:").grid(*row*=2, *column*=0, *padx*=10, *pady*=5)

        gender\_var = tk.StringVar()

        combo\_gender = ttk.Combobox(win, *textvariable*=gender\_var, *values*=["1-男", "2-女"], *state*='readonly')

        combo\_gender.grid(*row*=2, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="职称:").grid(*row*=3, *column*=0, *padx*=10, *pady*=5)

        title\_var = tk.StringVar()

        combo\_title = ttk.Combobox(win, *textvariable*=title\_var,

*values*=["1-博士后", "2-助教", "3-讲师", "4-副教授", "5-特任教授", "6-教授", "7-助理研究员", "8-特任副研究员", "9-副研究员", "10-特任研究员", "11-研究员"],

*state*='readonly')

        combo\_title.grid(*row*=3, *column*=1, *padx*=10, *pady*=5)

*def* save():

            tid = entry\_id.get().strip()

            name = entry\_name.get().strip()

            gender = gender\_var.get()

            title = title\_var.get()

            if not all([tid, name, gender, title]):

                messagebox.showerror("错误", "所有字段必须填写")

                return

            if len(tid) > 5:

                messagebox.showerror("错误", "工号不能超过5位")

                return

            g = *int*(gender.split('-')[0])

            t = *int*(title.split('-')[0])

            # 检查是否存在

            exist = *self*.db.execute\_query("SELECT teacher\_id FROM Teacher WHERE teacher\_id=%s", (tid,))

            if exist:

                messagebox.showerror("错误", "工号已存在")

                return

            res = *self*.db.execute\_update("INSERT INTO Teacher (teacher\_id,name,gender,title) VALUES (%s,%s,%s,%s)", (tid, name, g, t))

            if res > 0:

                messagebox.showinfo("成功", "教师添加成功")

                win.destroy()

*self*.teacher\_management()

            else:

                messagebox.showerror("错误", "教师添加失败")

        tk.Button(win, *text*="保存", *command*=save).grid(*row*=4, *column*=0, *columnspan*=2, *pady*=20)

*def* modify\_teacher(*self*, *tid*, *parent*):

        teacher\_info = *self*.db.execute\_query(

            "SELECT teacher\_id, name, gender, title " \

            "FROM Teacher " \

            "WHERE teacher\_id=%s", (*tid*,)

        )

        if not teacher\_info:

            messagebox.showerror("错误", "教师不存在")

            return

        old\_tid, old\_name, old\_gender, old\_title = teacher\_info[0]

        win = tk.Toplevel(*parent*)

        win.title("修改教师信息")

        win.geometry("400x300")

        tk.Label(win, *text*="工号:").grid(*row*=0, *column*=0, *padx*=10, *pady*=5)

        entry\_id = tk.Entry(win)

        entry\_id.insert(0, old\_tid)

        entry\_id.grid(*row*=0, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="姓名:").grid(*row*=1, *column*=0, *padx*=10, *pady*=5)

        entry\_name = tk.Entry(win)

        entry\_name.insert(0, *str*(old\_name))

        entry\_name.grid(*row*=1, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="性别:").grid(*row*=2, *column*=0, *padx*=10, *pady*=5)

        gender\_var = tk.StringVar()

        combo\_gender = ttk.Combobox(

            win,

*textvariable*=gender\_var,

*values*=["1-男", "2-女"],

*state*='readonly'

        )

        combo\_gender.set(*f*"{old\_gender}-" + combo\_gender.cget("values")[old\_gender-1].split('-',1)[1])

        combo\_gender.grid(*row*=2, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="职称:").grid(*row*=3, *column*=0, *padx*=10, *pady*=5)

        title\_var = tk.StringVar()

        combo\_title = ttk.Combobox(

            win,

*textvariable*=title\_var,

*values*=["1-博士后", "2-助教", "3-讲师", "4-副教授", "5-特任教授", "6-教授", "7-助理研究员", "8-特任副研究员", "9-副研究员", "10-特任研究员", "11-研究员"],

*state*='readonly'

        )

        combo\_title.set(*f*"{old\_title}-" + combo\_title.cget("values")[old\_title-1].split('-',1)[1])

        combo\_title.grid(*row*=3, *column*=1, *padx*=10, *pady*=5)

*def* save():

            new\_tid = entry\_id.get().strip()

            new\_name = entry\_name.get().strip()

            try:

                new\_gender = *int*(gender\_var.get().split('-')[0])

                new\_title = *int*(title\_var.get().split('-')[0])

            except:

                messagebox.showerror("错误", "请检查输入格式")

                return

            if not all([new\_tid, new\_name]):

                messagebox.showerror("错误", "所有字段必须填写")

                return

            if len(new\_tid) > 5:

                messagebox.showerror("错误", "工号不能超过5位")

                return

            res = *self*.db.execute\_update(

                "UPDATE Teacher SET teacher\_id=%s, name=%s, gender=%s, title=%s " \

                "WHERE teacher\_id=%s",(new\_tid, new\_name, new\_gender, new\_title, *tid*)

            )

            if res > 0:

                messagebox.showinfo("成功", "教师信息修改成功")

                win.destroy()

*parent*.destroy()

*self*.teacher\_management()

            else:

                messagebox.showerror("错误", "教师信息修改失败")

        tk.Button(win, *text*="保存", *command*=save).grid(*row*=4, *column*=0, *columnspan*=2, *pady*=20)

*def* delete\_teacher(*self*, *tid*):

        if messagebox.askyesno("确认", "确定删除该教师吗?"):

            res = *self*.db.execute\_update("DELETE FROM Teacher WHERE teacher\_id=%s", (*tid*,))

            if res > 0:

                messagebox.showinfo("成功", "教师删除成功")

*self*.teacher\_management()

# 3.4 论文管理模块（增删改查）

合作论文函数

*def* coop\_paper(*self*, *tid*, *parent*):

        win = tk.Toplevel(*parent*)

        win.title("合作论文")

        win.geometry("1000x300")

        # 获取可选论文

        papers = *self*.db.execute\_query(

            "SELECT paper\_id, paper\_name " \

            "FROM Paper " \

            "WHERE paper\_id NOT IN "

                "(SELECT paper\_id " \

                "FROM Teacher\_Paper " \

                "WHERE teacher\_id=%s)", (*tid*,)

        )

        choices = [

*f*"{pid}-{pname}"

            for pid, pname in papers

        ]

        tk.Label(win, *text*="选择论文:").grid(*row*=0, *column*=0, *padx*=10, *pady*=5)

        var = tk.StringVar()

        combo = ttk.Combobox(win, *textvariable*=var, *values*=choices, *state*='readonly', *width*=100)

        combo.grid(*row*=0, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="作者排名:").grid(*row*=1, *column*=0, *padx*=10, *pady*=5)

        entry\_rank = tk.Entry(win, *width*=100)

        entry\_rank.grid(*row*=1, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="是否通讯作者:").grid(*row*=2, *column*=0, *padx*=10, *pady*=5)

        corr\_var = tk.BooleanVar()

        tk.Checkbutton(win, *variable*=corr\_var).grid(*row*=2, *column*=1, *padx*=10, *pady*=5, *sticky*='w')

*def* save():

            sel = var.get().strip()

            if not sel:

                messagebox.showerror("错误", "请选择论文")

                return

            pid = *int*(sel.split('-')[0])

            try:

                prank = *int*(entry\_rank.get().strip())

            except:

                messagebox.showerror("错误", "排名必须为数字")

                return

            # 检查排名是否合法

            if prank < 1:

                messagebox.showerror("错误", "作者排名必须大于0")

                return

            # 通讯作者的唯一性

            pcorr = corr\_var.get()

            if pcorr:

                cnt = *self*.db.execute\_query("SELECT COUNT(\*) FROM Teacher\_Paper WHERE paper\_id=%s AND is\_corresponding=1", (pid,))

                if cnt and cnt[0][0] > 0:

                    messagebox.showerror("错误", "该论文已有通讯作者")

                    return

            # 检查排名是否唯一

            # ? cnt是查询结果，cnt[0][0]是查询到的行数，若没查到则为0

            cnt = *self*.db.execute\_query(

                "SELECT COUNT(\*) FROM Teacher\_Paper WHERE paper\_id=%s AND author\_rank=%s AND teacher\_id<>%s",

                (pid, prank, *tid*)

            )

            if cnt and cnt[0][0] > 0:

                messagebox.showerror("错误", *f*"排名 {prank} 已被其他作者使用，请换一个")

                return

            res = *self*.db.execute\_update("INSERT INTO Teacher\_Paper (teacher\_id,paper\_id,author\_rank,is\_corresponding) VALUES (%s,%s,%s,%s)", (*tid*, pid, prank, pcorr))

            if res > 0:

                messagebox.showinfo("成功", "合作论文添加成功")

                win.destroy()

*parent*.destroy()

*self*.paper\_management(*tid*)

            else:

                messagebox.showerror("错误", "添加失败，可能重复")

        tk.Button(win, *text*="保存", *command*=save).grid(*row*=3, *column*=0, *columnspan*=2, *pady*=20)

查询功能函数

*def* search\_paper\_window(*self*, *tid*):

        term = simpledialog.askstring("查询", "请输入论文名称或序号:")

        if term is None:

            return

        try:

            pid = *int*(term)

            query = "SELECT p.paper\_id, p.paper\_name, p.pub\_source, p.pub\_year, tp.author\_rank, tp.is\_corresponding FROM Teacher\_Paper tp JOIN Paper p ON tp.paper\_id=p.paper\_id WHERE tp.teacher\_id=%s AND p.paper\_id=%s"

            params = (*tid*, pid)

        except:

            query = "SELECT p.paper\_id, p.paper\_name, p.pub\_source, p.pub\_year, tp.author\_rank, tp.is\_corresponding FROM Teacher\_Paper tp JOIN Paper p ON tp.paper\_id=p.paper\_id WHERE tp.teacher\_id=%s AND p.paper\_name LIKE %s"

            params = (*tid*, *f*"%{term}%")

        results = *self*.db.execute\_query(query, params)

        if not results:

            messagebox.showinfo("结果", "未找到相关论文")

            return

        win = tk.Toplevel(*self*.root)

        win.title("查询结果")

        for i, (pid, name, src, yr, rank, corr) in enumerate(results):

            text = *f*"{pid} | {name} | {src} | {yr} | 排名:{rank} | {'通讯作者' if corr else ''}"

            tk.Label(win, *text*=text).pack(*anchor*='w', *padx*=10, *pady*=2)

其他原理类似3.3，不再赘述

# 3.5 课程管理模块（增删改查）

原理类似3.4，不再赘述

# 3.6 项目管理模块（增删改查）

原理类似3.4，不再赘述

# 3.7 查询统计模块

*def* query\_stats(*self*):

        win = tk.Toplevel(*self*.root)

        win.title("查询统计")

        win.geometry("800x600")

        tk.Label(win, *text*="教师工号:").grid(*row*=0, *column*=0, *padx*=10, *pady*=5)

        entry\_id = tk.Entry(win)

        entry\_id.grid(*row*=0, *column*=1, *padx*=10, *pady*=5)

        tk.Label(win, *text*="年份范围(如2022-2023):").grid(*row*=1, *column*=0, *padx*=10, *pady*=5)

        entry\_range = tk.Entry(win)

        entry\_range.grid(*row*=1, *column*=1, *padx*=10, *pady*=5)

        txt = tk.Text(win)

        txt.grid(*row*=2, *column*=0, *columnspan*=2, *padx*=10, *pady*=10, *sticky*='nsew')

        win.grid\_rowconfigure(2, *weight*=1)

        win.grid\_columnconfigure(1, *weight*=1)

*def* search():

            tid = entry\_id.get().strip()

            yr = entry\_range.get().strip()

            if not tid:

                messagebox.showerror("错误", "请输入工号")

                return

            start = end = None

            if yr:

                try:

                    s, e = yr.split('-')

                    start, end = *int*(s), *int*(e)

                except:

                    messagebox.showerror("错误", "年份格式错误")

                    return

                # 检查年份在1958到2100之间

            if start is not None and (start < 1958 or start > 2100):

                messagebox.showerror("错误", "开始年份必须在1958到2100之间")

                return

            if end is not None and (end < 1958 or end > 2100):

                messagebox.showerror("错误", "结束年份必须在1958到2100之间")

                return

            txt.delete(1.0, tk.END)

            # 教师基本信息

            info = *self*.db.execute\_query(

                "SELECT name, "

                "CASE gender WHEN 1 THEN '男' WHEN 2 THEN '女' END as gender, "

                "CASE title WHEN 1 THEN '博士后' WHEN 2 THEN '助教' WHEN 3 THEN '讲师' WHEN 4 THEN '副教授' "

                "WHEN 5 THEN '特任教授' WHEN 6 THEN '教授' WHEN 7 THEN '助理研究员' WHEN 8 THEN '特任副研究员' "

                "WHEN 9 THEN '副研究员' WHEN 10 THEN '特任研究员' WHEN 11 THEN '研究员' END as title "

                "FROM Teacher WHERE teacher\_id=%s", (tid,)

            )

            if not info:

                messagebox.showerror("错误", "教师不存在")

                return

            name, gender, title = info[0]

            txt.insert(tk.END, *f*"教师基本信息:\n工号:{tid}\t姓名:{name}\t性别:{gender}\t职称:{title}\n\n")

            # 论文

            txt.insert(tk.END, "论文:\n")

            qp = (

                "SELECT p.paper\_name, p.pub\_source, p.pub\_year, p.paper\_type, p.paper\_level, "

                "tp.author\_rank, tp.is\_corresponding "

                "FROM Teacher\_Paper tp JOIN Paper p ON tp.paper\_id=p.paper\_id "

                "WHERE tp.teacher\_id=%s"

            )

            params = [tid]

            if start is not None and end is not None:

                qp += " AND p.pub\_year BETWEEN %s AND %s"

                params += [start, end]

            qp += " ORDER BY p.pub\_year DESC"

            papers = *self*.db.execute\_query(qp, *tuple*(params))

            if papers:

                for pname, src, pyr, ptype, plevel, rank, corr in papers:

                    type\_desc = paper\_type\_map.get(ptype, '未知类型')

                    level\_desc = paper\_level\_map.get(plevel, '未知级别')

                    txt.insert(tk.END, *f*"{pname} | {src} | {pyr} | 类型:{type\_desc} | 级别:{level\_desc} | 排名:{rank} | {'通讯作者' if corr else ''}\n")

            else:

                txt.insert(tk.END, "无论文记录\n")

            # 课程

            txt.insert(tk.END, "\n课程:\n")

            # txt.insert(tk.END, "------------------\n")

            qc = (

                "SELECT c.course\_id, c.course\_name, tc.year, tc.semester, tc.my\_hours "

                "FROM Teacher\_Course tc JOIN Course c ON tc.course\_id=c.course\_id "

                "WHERE tc.teacher\_id=%s"

            )

            params = [tid]

            if start is not None and end is not None:

                qc += " AND tc.year BETWEEN %s AND %s"

                params += [start, end]

            qc += " ORDER BY tc.year, tc.semester"

            courses = *self*.db.execute\_query(qc, *tuple*(params))

            if courses:

                for cid, cname, yr2, sem, hrs in courses:

                    sem\_desc = semester\_map.get(sem, *str*(sem))

                    txt.insert(tk.END, *f*"课程号:{cid}\t课程名{cname}\t学期:{yr2}{sem\_desc}\t主讲学时:{hrs}\n")

            else:

                txt.insert(tk.END, "无课程记录\n")

            # 项目

            txt.insert(tk.END, "\n项目:\n")

            qpr = (

                "SELECT p.project\_id, p.project\_name, p.project\_type, p.project\_source, "

                "CONCAT(p.start\_year, '-', p.end\_year), tp.principal\_rank, tp.my\_fund, p.total\_fund "

                "FROM Teacher\_Project tp JOIN Project p ON tp.project\_id=p.project\_id "

                "WHERE tp.teacher\_id=%s"

            )

            params = [tid]

            if start is not None and end is not None:

                qpr += " AND NOT(p.end\_year < %s OR p.start\_year > %s)"

                params += [start, end]

            qpr += " ORDER BY p.start\_year DESC"

            projs = *self*.db.execute\_query(qpr, *tuple*(params))

            if projs:

                for pid, pname, ptype, src, dur, rank, mf, ttlf in projs:

                    type\_desc = project\_type\_map.get(ptype, '未知类型')

                    txt.insert(tk.END, *f*"{pname} | {src} | 类型:{type\_desc} | 年度:{dur} | 排名:{rank} | 总经费:{ttlf} | 承担经费:{mf}\n")

            else:

                txt.insert(tk.END, "无项目记录\n")

        tk.Button(win, *text*="查询", *command*=search).grid(*row*=3, *column*=0, *columnspan*=2, *pady*=10)

# 3.8 导出报告模块

*def* export\_report(*self*):

        tid = simpledialog.askstring("输入", "请输入教师工号:")

        yr = simpledialog.askstring("输入", "请输入年份范围(如2022-2023):")

        if not tid or not yr:

            return

        try:

            s, e = yr.split('-')

            start = *int*(s)

            end = *int*(e)

        except:

            messagebox.showerror("错误", "年份格式错误")

            return

        # 检查年份在1958到2100之间

        if start < 1958 or start > 2100 or end < 1958 or end > 2100:

            messagebox.showerror("错误", "年份必须在1958到2100之间")

            return

        # 检索数据

        info = *self*.db.execute\_query(

            "SELECT name, CASE gender WHEN 1 THEN '男' WHEN 2 THEN '女' END as gender, CASE title WHEN 1 THEN '博士后' WHEN 2 THEN '助教' WHEN 3 THEN '讲师' WHEN 4 THEN '副教授' WHEN 5 THEN '特任教授' WHEN 6 THEN '教授' WHEN 7 THEN '助理研究员' WHEN 8 THEN '特任副研究员' WHEN 9 THEN '副研究员' WHEN 10 THEN '特任研究员' WHEN 11 THEN '研究员' END as title FROM Teacher WHERE teacher\_id=%s", (tid,))

        if not info:

            messagebox.showerror("错误", "教师不存在")

            return

        name, gender, teacher\_title = info[0]

        papers = *self*.db.execute\_query(

            "SELECT p.paper\_name, p.pub\_source, p.pub\_year, p.paper\_type, p.paper\_level, "

            "tp.author\_rank, tp.is\_corresponding "

            "FROM Teacher\_Paper tp JOIN Paper p ON tp.paper\_id=p.paper\_id "

            "WHERE tp.teacher\_id=%s AND p.pub\_year BETWEEN %s AND %s "

            "ORDER BY p.pub\_year DESC",

            (tid, start, end)

        )

        courses = *self*.db.execute\_query(

            "SELECT c.course\_id, c.course\_name, tc.year, tc.semester, tc.my\_hours "

            "FROM Teacher\_Course tc JOIN Course c ON tc.course\_id=c.course\_id "

            "WHERE tc.teacher\_id=%s AND tc.year BETWEEN %s AND %s "

            "ORDER BY tc.year, tc.semester",

            (tid, start, end)

        )

        projs = *self*.db.execute\_query(

            "SELECT p.project\_id, p.project\_name, p.project\_type, p.project\_source, "

            "CONCAT(p.start\_year, '-', p.end\_year), tp.principal\_rank, tp.my\_fund, p.total\_fund "

            "FROM Teacher\_Project tp JOIN Project p ON tp.project\_id=p.project\_id "

            "WHERE tp.teacher\_id=%s AND NOT(p.end\_year < %s OR p.start\_year > %s) "

            "ORDER BY p.start\_year DESC",

            (tid, start, end)

        )

        # 生成PDF - 使用Paragraph支持自动换行

        filename = *f*"报告\_{name}\_{yr}.pdf"

        c = canvas.Canvas(filename, *pagesize*=letter)

        width, height = letter

        y\_position = height - 40

        # 设置样式

        styles = getSampleStyleSheet()

        # 标题样式

        title\_style = styles['Title']

        title\_style.fontName = chinese\_font

        title\_style.fontSize = 18

        title\_style.leading = 22

        # 教师信息样式

        info\_style = styles['Normal']

        info\_style.fontName = chinese\_font

        info\_style.fontSize = 12

        info\_style.leading = 15

        # 内容样式

        content\_style = styles['BodyText']

        content\_style.fontName = chinese\_font

        content\_style.fontSize = 10

        content\_style.leading = 14

        # 绘制标题

        header = Paragraph(*f*"教师教学科研报告 ({yr})", title\_style)

        header.wrap(width - 80, 40)

        header.drawOn(c, 40, y\_position)

        y\_position -= 40

        # 绘制教师基本信息

        info\_text = *f*"<b>教师基本信息:</b><br/>" \

*f*"工号: {tid} &nbsp; 姓名: {name} &nbsp; 性别: {gender} &nbsp; 职称: {teacher\_title}"

        info\_para = Paragraph(info\_text, info\_style)

        info\_para.wrap(width - 80, 40)

        info\_para.drawOn(c, 40, y\_position)

        y\_position -= 40

        # 课程部分

        courses\_text = "<b>课程:</b><br/>"

        if courses:

            for cid, cname, yr2, sem, hrs in courses:

                sem\_desc = semester\_map.get(sem, *str*(sem))

                short\_name = cname if len(cname)<=50 else cname[:50]+"..."

                courses\_text += (

*f*"• {cid} | {yr2}-{sem\_desc} | {short\_name} | 学时:{hrs}<br/>"

                )

        else:

            courses\_text += "无课程记录<br/>"

        courses\_para = Paragraph(courses\_text, content\_style)

        courses\_height = courses\_para.wrap(width - 80, height - 100)[1]

        if y\_position - courses\_height < 50:

            c.showPage()

            y\_position = height - 40

        courses\_para.drawOn(c, 40, y\_position - courses\_height)

        y\_position -= courses\_height + 20

        # 论文部分

        papers\_text = "<b>论文:</b><br/>"

        if papers:

            for pname, src, pyr, ptype, plevel, rank, corr in papers:

                corr\_str = " (通讯作者)" if corr else ""

                # 类型/级别映射

                type\_desc  = paper\_type\_map.get(ptype, '未知类型')

                level\_desc = paper\_level\_map.get(plevel, '未知级别')

                # 长度截断

                short\_name = pname if len(pname)<=100 else pname[:100]+"..."

                papers\_text += (

*f*"• {pyr} | {short\_name} | {src} | "

*f*"类型:{type\_desc} | 级别:{level\_desc} | 排名:{rank}{corr\_str}<br/>"

                )

        else:

            papers\_text += "无论文记录<br/>"

        papers\_para = Paragraph(papers\_text, content\_style)

        papers\_height = papers\_para.wrap(width - 80, height - 100)[1]

        if y\_position - papers\_height < 50:  # 检查空间是否足够

            c.showPage()

            y\_position = height - 40

        papers\_para.drawOn(c, 40, y\_position - papers\_height)

        y\_position -= papers\_height + 20

        # 项目部分

        projs\_text = "<b>项目:</b><br/>"

        if projs:

            for pid, pname, ptype, src, dur, rank, mf, tf in projs:

                short\_name = pname if len(pname)<=80 else pname[:80]+"..."

                type\_desc = project\_type\_map.get(ptype, '未知类型')

                projs\_text += (

*f*"• {pid} | {short\_name} | 类型:{type\_desc} | {src} | "

*f*"年度:{dur} | 排名:{rank} | 承担经费:{mf} | 总经费:{tf}<br/>"

                )

        else:

            projs\_text += "无项目记录<br/>"

        projs\_para = Paragraph(projs\_text, content\_style)

        projs\_height = projs\_para.wrap(width - 80, height - 100)[1]

        if y\_position - projs\_height < 50:

            c.showPage()

            y\_position = height - 40

        projs\_para.drawOn(c, 40, y\_position - projs\_height)

        c.save()

        messagebox.showinfo("成功", *f*"PDF报告已生成: {filename}")

# 4 实现与测试

## 4.1 实现结果

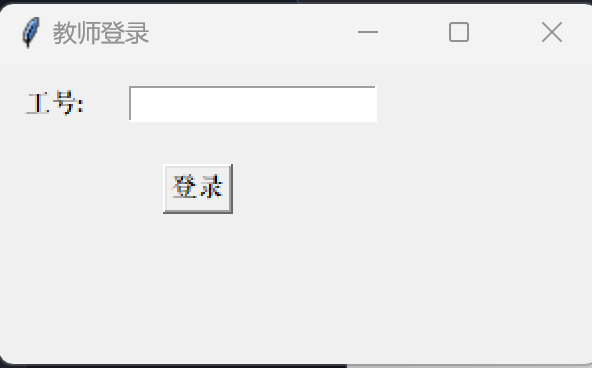
主页面



教师管理界面



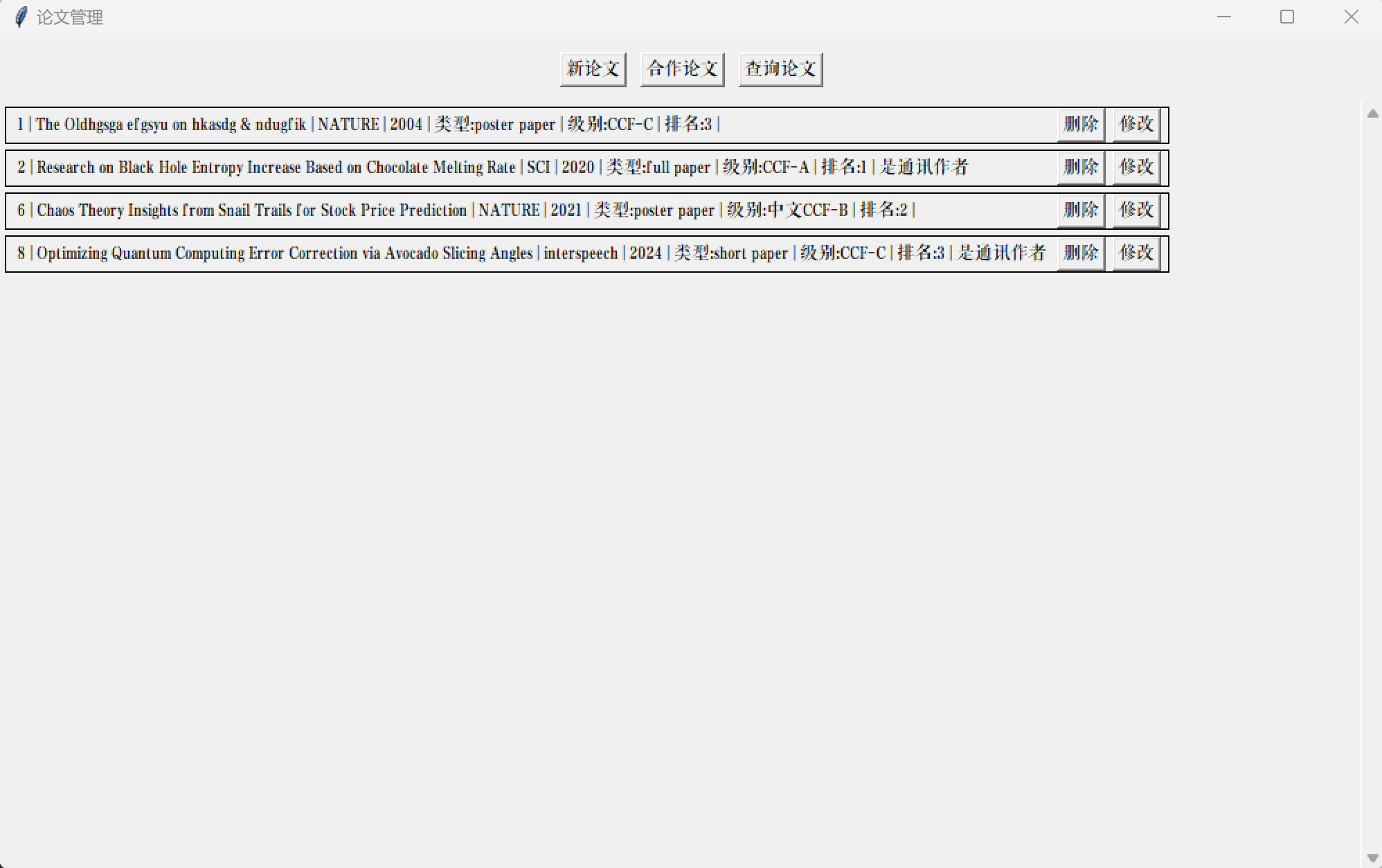
教师登录界面



登陆成功后的教师界面



论文管理界面



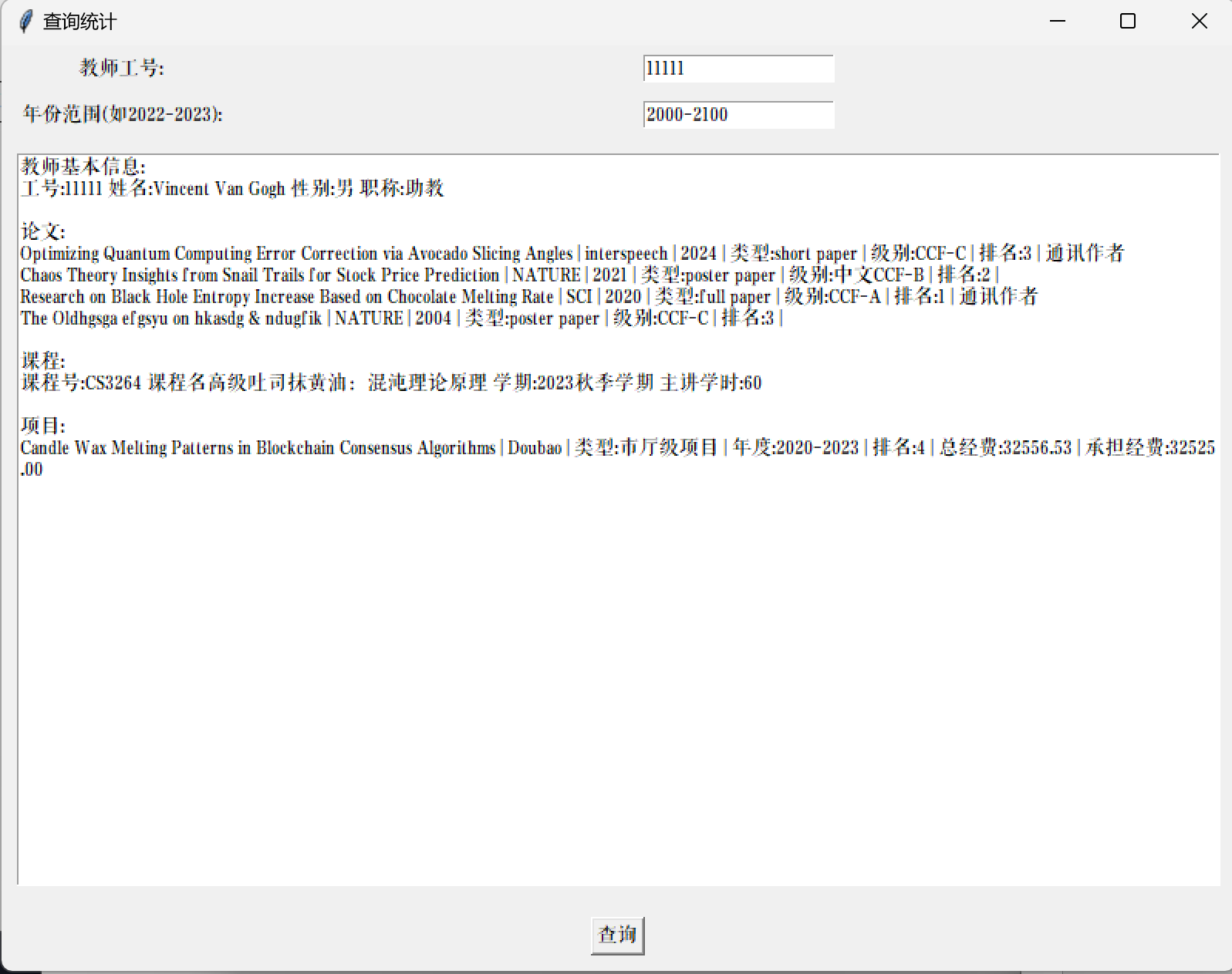
课程管理界面



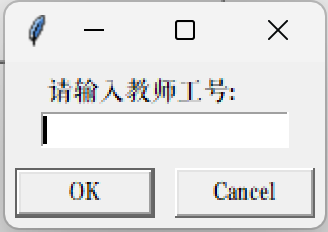
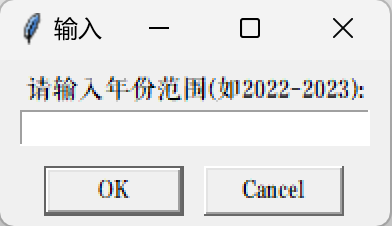
项目管理界面



查询统计界面



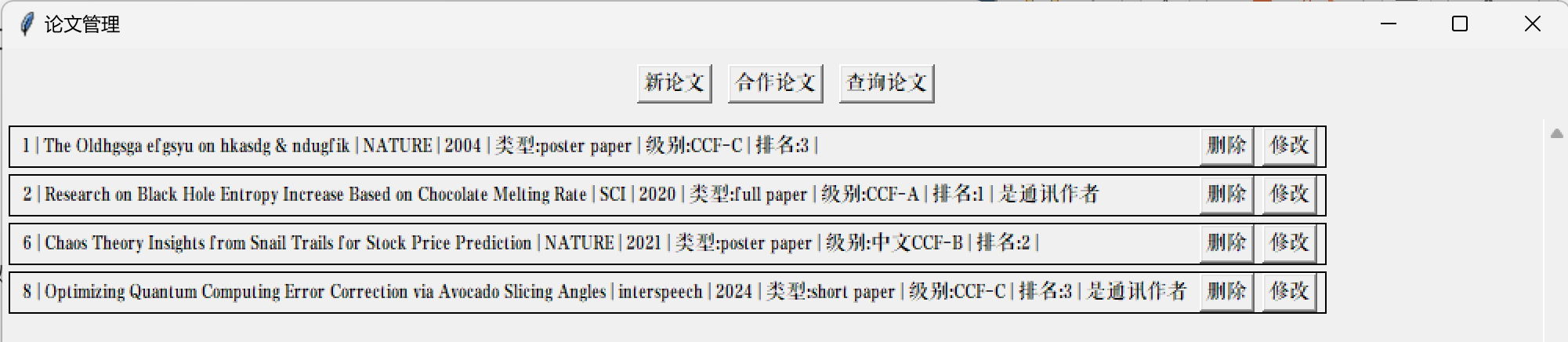
导出报告界面

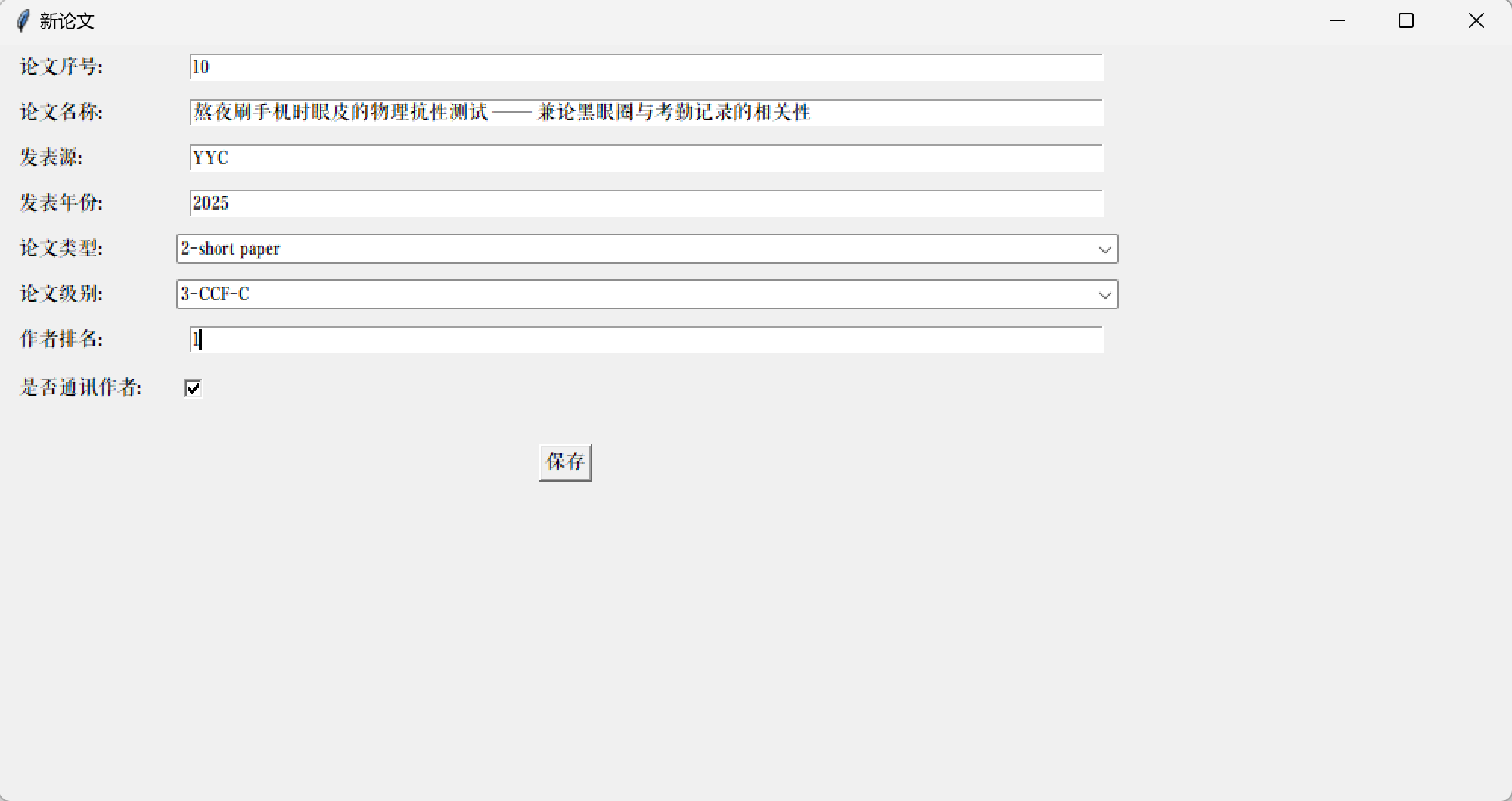
## 4.2 测试结果

// 给出各个功能需求的测试用例和测试结果。

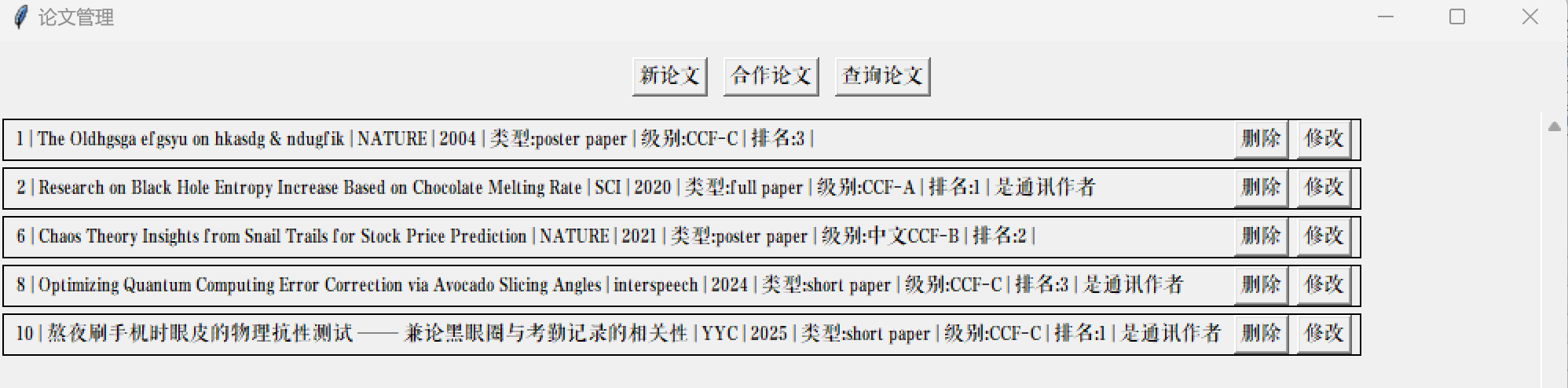
1. 增加功能（以论文为示例），在操作前论文列表如下



现在点击新论文可以增加新的论文

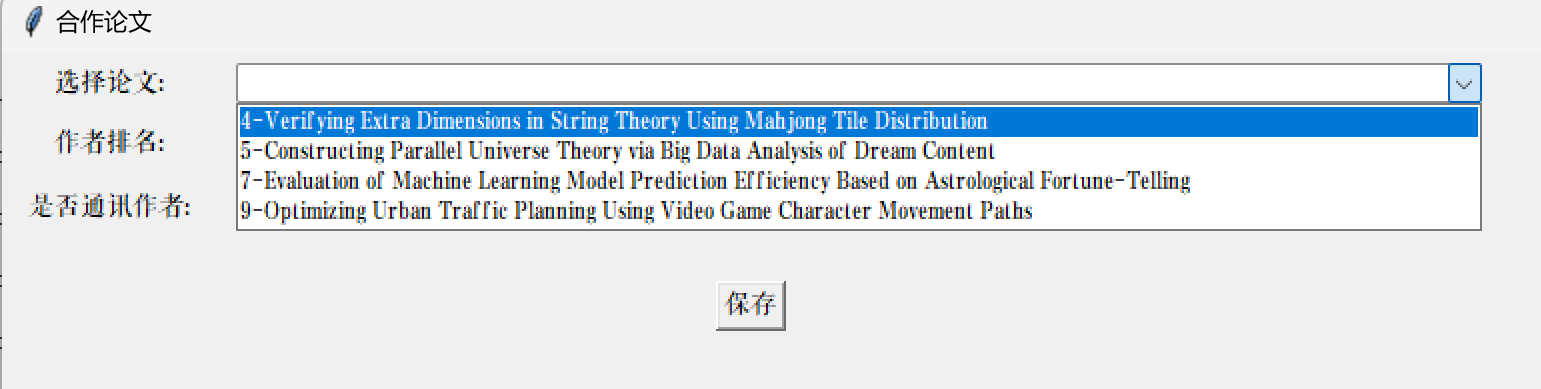


保存后论文管理界面自动刷新并显示如下



1. 合作功能（以课程为示例）。

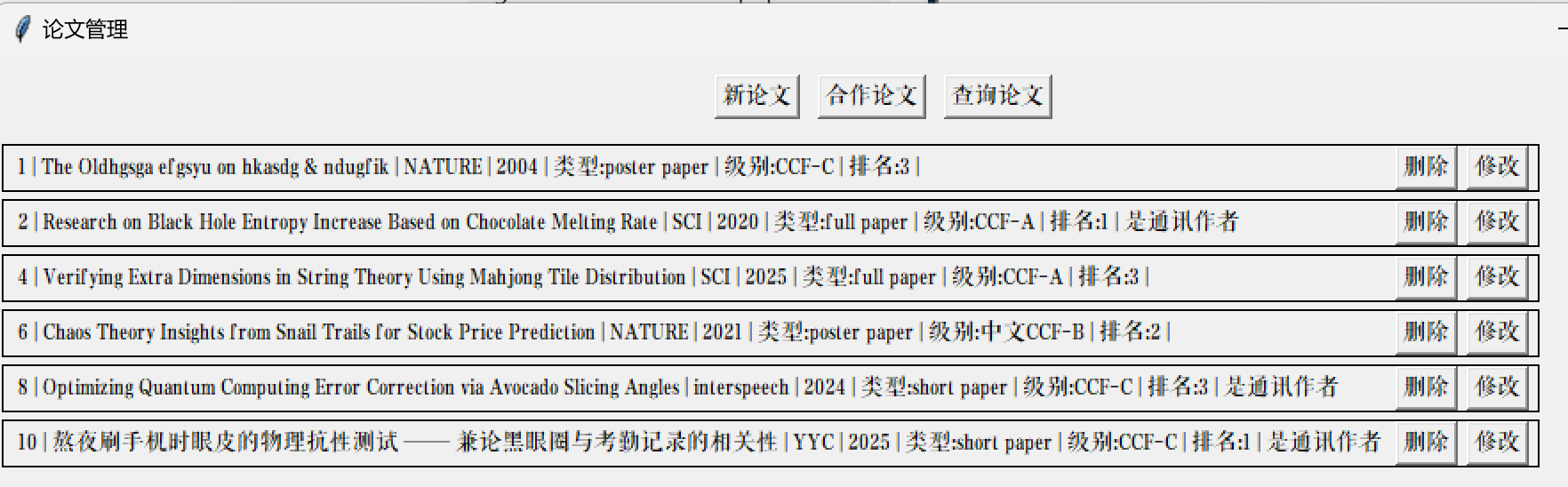
下拉表中能显示所有该教师未参与的论文，按序号排序



选择4号论文，若该论文已经有通讯作者还勾选了通讯作者，会提示

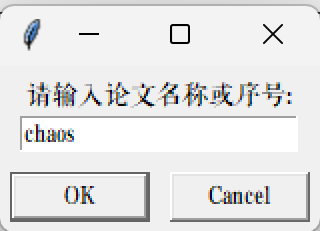


保存后在论文管理界面看到

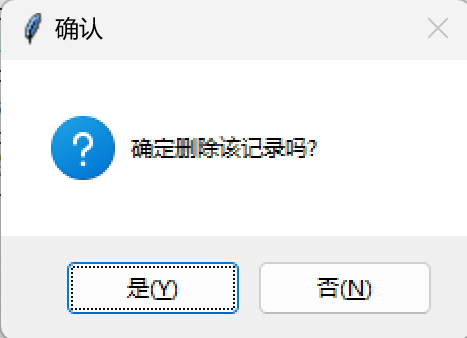


1. 查询功能（以论文为示例）

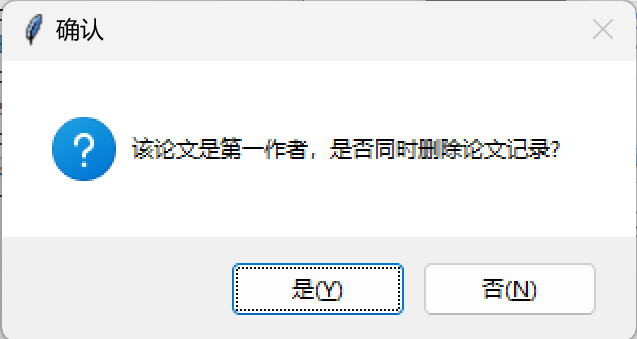
查询框内可以对名称或者序号进行搜索

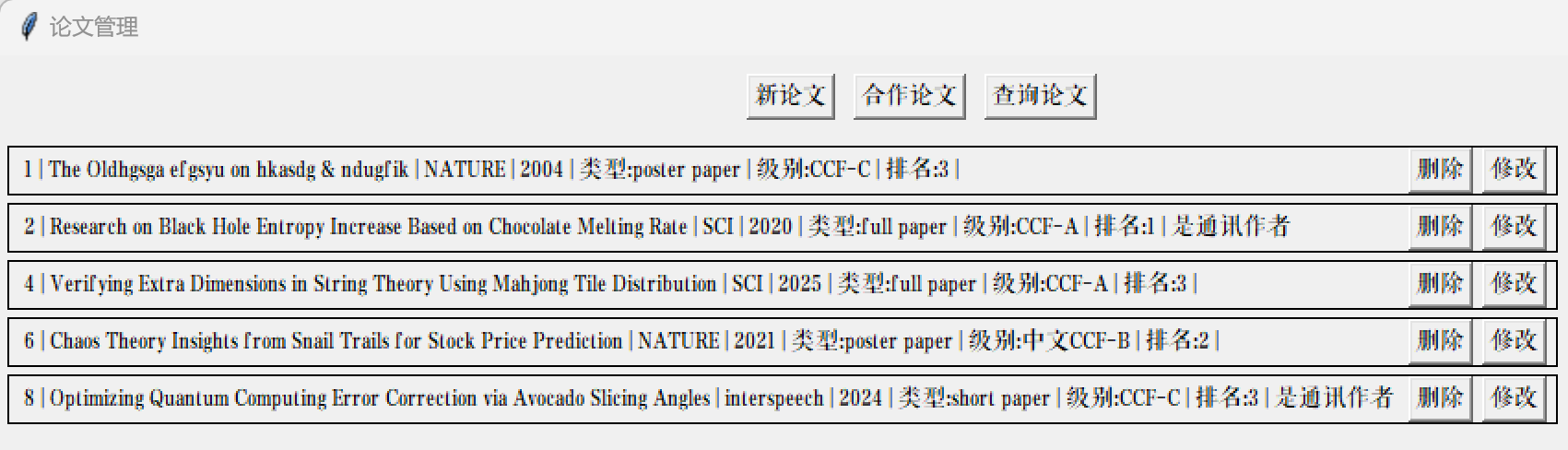
1. 删除功能（以论文为示例）



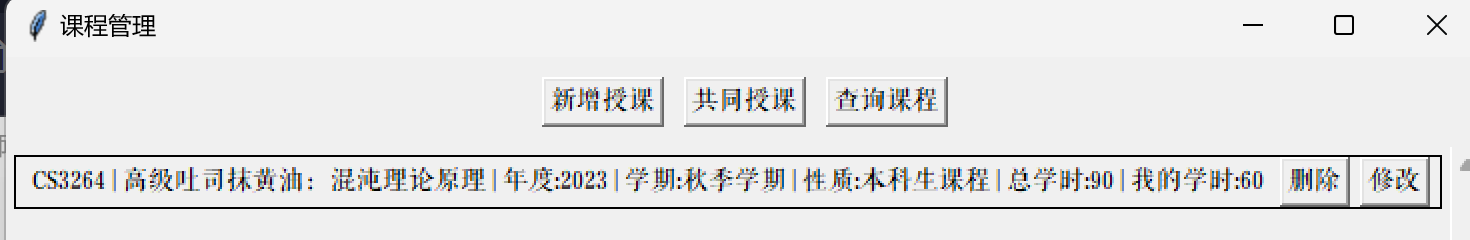
若恰好是该论文第一作者，会提示是否同时删除该论文



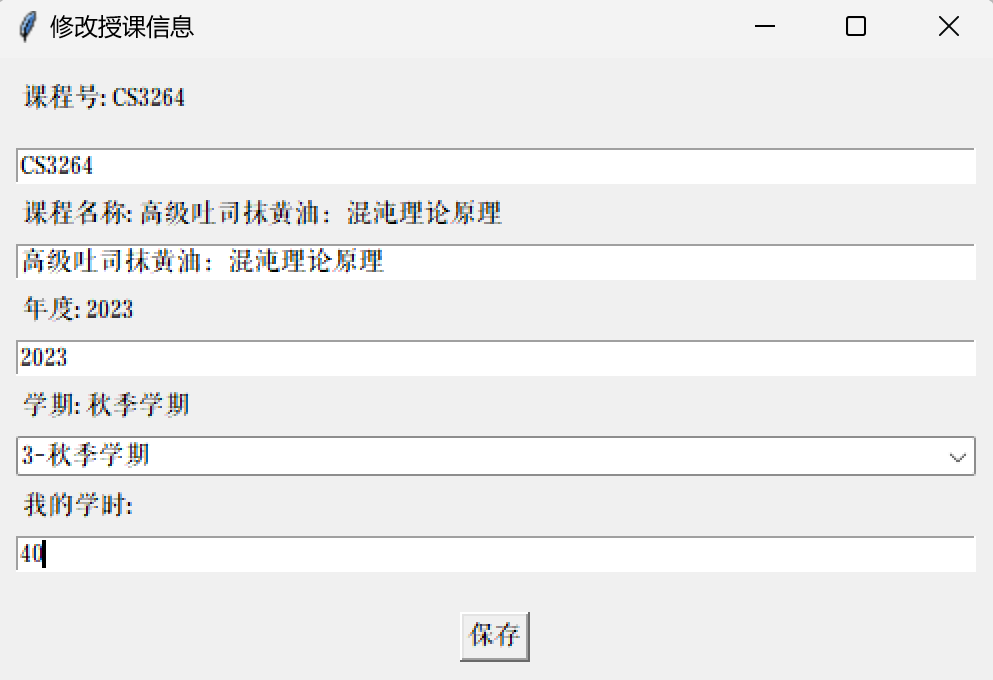
删除后是成功的



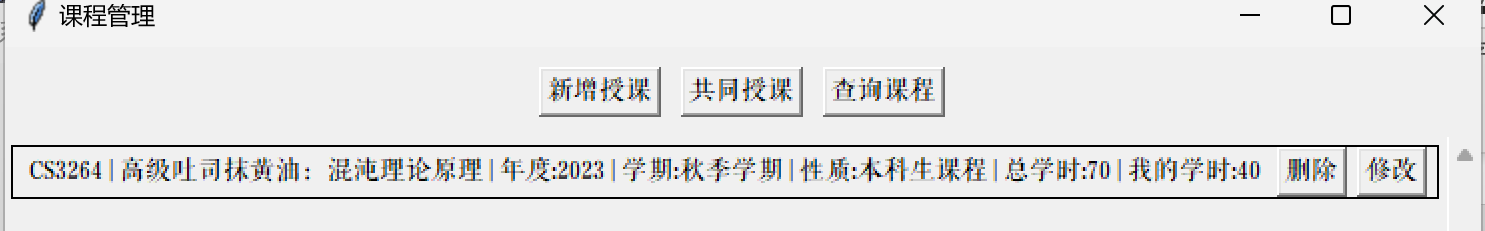
1. 修改功能（以课程为示例），一开始是这样



将我的学时从60改为40

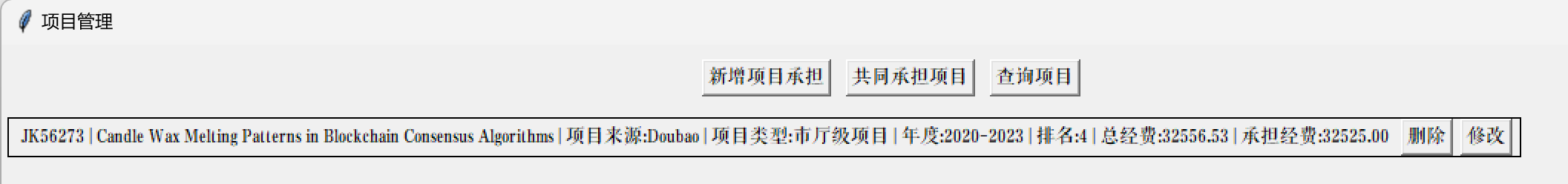


修改完成后如下，可以看到学时总数同步减少了



（以项目为示例）

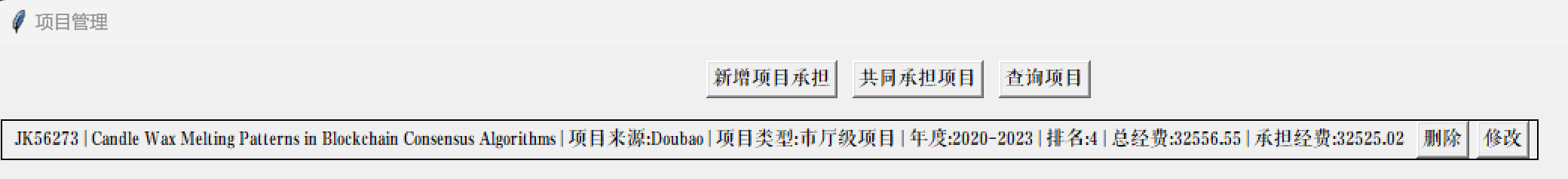
一开始是这样



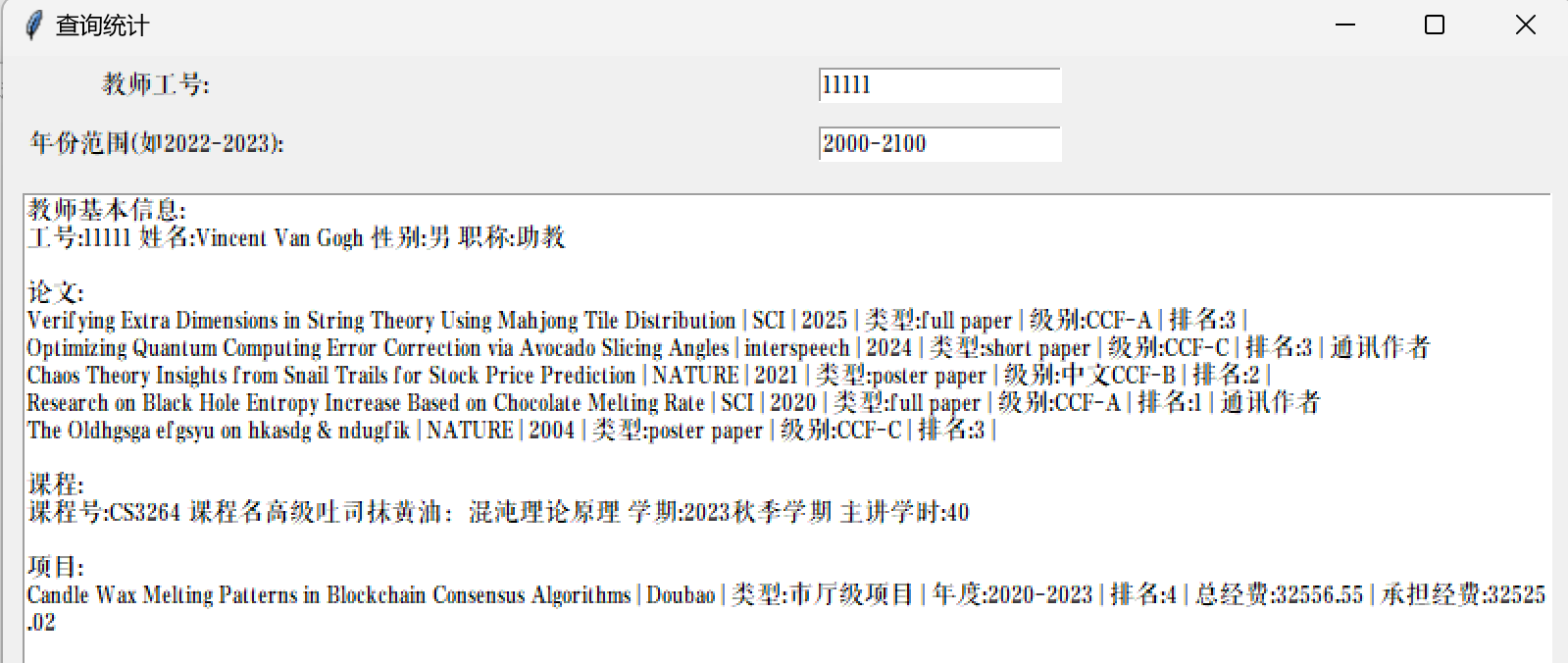
将我的经费从32525.00改成32525.02



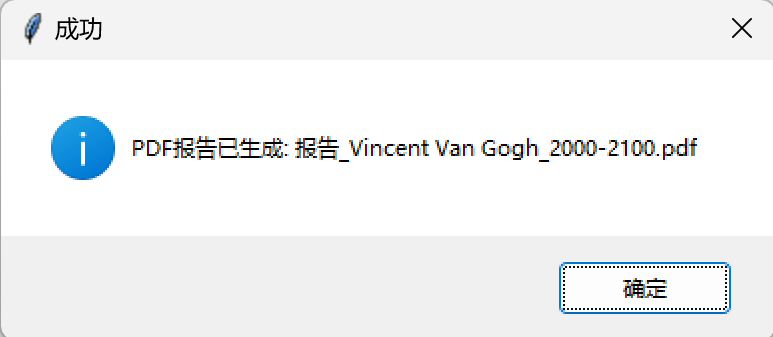
保存后看到总经费也增加了0.02



1. 查询统计功能，输入工号和年份范围后，可以得到一个详细记录



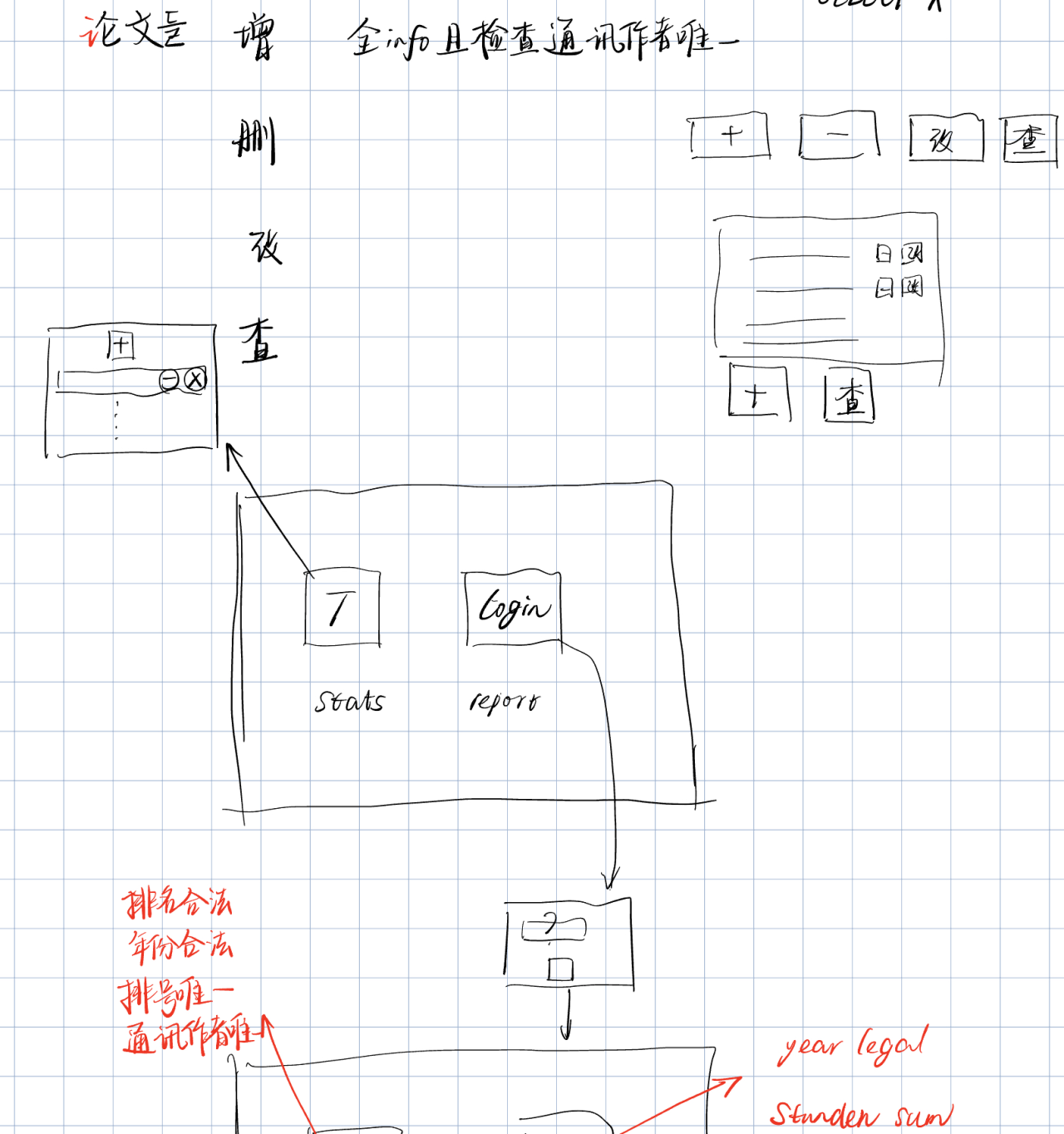
1. 导出报告功能，输入工号和年份范围后，可以得到一个带有详细记录的pdf文件





## 4.3 实现中的难点问题及解决【可选】

此次设计的难点在于，如果合理地设计一个便于操作的概念模型，保证逻辑结构的合理性同时保证操作的便利性。考虑到本系统的面向对象是教师，于是将主界面设计成了含有登录界面，登录后教师可以对自己的论文、课程、项目进行分开的管理。以下是在设计初期的草图。而后考虑到论文、课程、项目既可以直接创建新的，也可以和已经存在的条目进行合作，于是分裂出一个“合作”功能，该功能对该教师未参与但是存在的条目进行列举，因此教师可以选择加入，此时只需要修改关系联系表上的属性。



# 5 总结与讨论

1. 概念模型的设计很重要，关乎用户体验的后端维护难度，本次实验实现了数据库管理系统的核心功能；
2. 在导出报告中不慎将 title 变量重用，导致 PDF 中出现对象字符串；改用 report\_header、teacher\_title 等清晰命名后消除了混淆。
3. 在多入口的滚动列表中，按钮对齐和文本框宽度的调整较为繁琐，Pack 与 Grid 的混用增加了调试成本。
4. 保证课程总学时与项目总经费的实时同步需要在各处插入额外的更新 SQL，未来可考虑触发器或存储过程来统一管理。
5. 进一步改进空间：
   1. 引入 ttk 主题或第三方库（如 Tkinter Designer、PySimpleGUI）提升 UI 美观度；
   2. 将课程与项目的“总量同步”逻辑移到数据库端（触发器）或更高层的服务接口中，减少前端重复代码；
   3. 增加导出为 Excel、Word 等格式的报告选项；
   4. 编写单元测试和 UI 自动化测试，保证各功能在迭代后依然稳定；