国家开放大学

本科毕业论文

题目：基于JAVA的社团管理系统设计与实现

分部：国家开放大学实验学院

学习中心：魏公村学习中心

专业：计算机科学与技术

入学时间：22秋

学号：228020125XXX

姓名：XXX

指导教师：XXX

论文完成日期: 2025 年6月

**毕业论文原创性声明**

本人郑重声明：所呈交的毕业论文，是本人在导师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本毕业论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

作者签名： 日期：2024年 11月18日

**毕业论文版权使用授权声明**

本人完全了解国家开放大学关于收集、保存、使用毕业论文的规定，同意如下各项内容：按照学校要求提交毕业论文的印刷本和电子版本；学校有权保存毕业论文的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存论文；学校有权提供目录检索以及提供本毕业论文全文或者部分的阅览服务，以及出版毕业论文；学校有权按有关规定向国家有关部门或者机构送交论文的复印件和电子版；在不以赢利为目的的前提下，学校可以适当复制论文的部分或全部内容用于学术活动。

作者签名： 日期：2024年11月18日

**目 录**

[一、个人开发者的时代转型 1](#_Tocb7a612)

[（一）技术背景的变化 1](#_Tocse5ym4)

[（二）个人角色的转型 1](#_Tocxni96c)

[二、工作流程与部署实践 3](#_Tocr5hdsm)

[（一）实际项目背景 3](#_Tocd27y83)

[三、架构演进与实践反思 5](#_Tocrdjnh3)

[（一）初期的混乱结构 5](#_Tocqajgew)

[（二）三层分离架构的提出 5](#_Tocbljayg)

[（三）七文件生成器的实践 5](#_Tocv7r69k)

[（四）架构思维的反复修正 5](#_Toctgcvtu)

[（五）架构的职业意义 5](#_Tocetmkao)

[（六）起点：一地鸡毛的代码堆 5](#_Toce91vbc)

[（七）转折：第一次提出“业务层永远不写 SQL” 6](#_Toct462ou)

[（八）实践：我做了一个代码生成器 6](#_Tockcny04)

[（九）跌倒与顿悟：失败是生长出来的土壤 6](#_Toc6rorai)

[（十）最终：不只是一个架构，而是一种职业信念 7](#_Toc3orqf4)

[四、AI驱动的全栈开发实践 8](#_Toc9pak9b)

[（一）AI工具链实战 8](#_Toc323kzs)

[1. AI 在需求拆解中的作用 8](#_Tocyuh81b)

[2. 编程助理与代码重构 8](#_Tocyu5i1p)

[3. 与 AI 共创的思维方式 8](#_Toc22hca7)

[（二）AI效率对比与反思 8](#_Toc6gsig0)

[1. 量化效果数据 8](#_Toczw3v2y)

[五、WEBRTC 项目实战 9](#_Tociab64t)

[（一）项目概述 9](#_Tochedce0)

[（二）技术目标与实训目的 9](#_Toc49sr8t)

[（三）页面风格与用户体验设计 9](#_Tocvob0ma)

[（四）网站部署与运行说明 9](#_Toclqxpx4)

[（五）实训总结与学习收获 10](#_Tocmuwgfe)

[（六）项目中遇到的问题与解决策略 10](#_Tocp8ads2)

[（七）总结 10](#_Tocyquhs1)

[六、家庭工具项目案例 11](#_Tocpzg3yu)

[1. 倒计时锁屏软件 11](#_Toc9zedcn)

[2. 英语词汇打卡插件 11](#_Toc4n2vll)

[七、用户反馈与迭代优化 11](#_Tocf71v90)

[八、未来展望与个人规划 12](#_Tocpph6sk)

[九、参考文献 12](#_Tocanxqxb)

本文所有技术方案均经过实际项目验证，部分核心代码仓库地址为：github.com/guomengtao

# 一、个人开发者的时代转型

## （一）技术背景的变化

过去十年，编程技术发展迅猛，从桌面应用到 Web，再到移动端，再到如今的 AI 与低代码平台，技术范式不断演化。尤其是 AI 工具的普及，显著改变了个人开发的能力边界，使得从“不会写代码的人”也能逐步参与到应用构建过程中。

## （二）个人角色的转型

作为一个曾以教程创作与手工开发为主的个人开发者，我逐渐转向以项目驱动为主的创造路径。AI 助手的介入，使我更像一位“设计者”与“决策者”，不再执着于底层实现，而是关注架构、逻辑与最终的用户价值。 ## （三）项目的实践意义

在这个技术日新月异的时代，我曾经把一枚U盘挂在脖子上，那里面承载着我花费多年精力制作的视频教程。这些视频中，有我精心录制的编程教学，有我用心构建的项目实录，还有我对技术与人生的点滴思考。

它们是我生命意义的一部分。

曾几何时，那些教程帮助了许多初学者，他们通过微信给我发来感谢红包，甚至寄来了家乡的特产。那是我第一次真正感受到：一个普通人，也可以通过知识，帮助他人，并且被人铭记。

然而，AI 的到来，彻底重塑了这种路径。像 ChatGPT、DeepSeek 这样的智能助手，几乎完全改变了传统的学习方式。我原来花几十个小时制作的视频教程，现在通过 AI，几分钟就能让用户上手，并快速构建项目。

我自己就是一个典型例子。比如我希望在 Mac 上开发一个倒计时锁屏应用，传统方式下我需要查阅资料、学习语言、调试功能。但现在，我只要把我的需求清晰地表达给 AI，它就能给出完整的代码。我甚至不再需要去翻阅大量文档。

AI 并不是在”替代”我，而是让我从”编码者”转向了”构建者”。我成了一个有方向、有判断、会表达的产品实践者。我需要掌握的是目标、是愿景、是过程的组织。

所以我开始意识到，视频教程这种形式，在新时代下正在失去它的中心地位。它不再是”唯一的学习之道”，而逐渐变成了一种”辅助性的沉淀材料”。

那我还能做什么？我还能留下什么？

答案是在我亲手构建的软件中寻找。我做出了一个倒计时锁屏软件，它每天都在帮助我和孩子更好地安排时间，让我们在游戏与学习之间找到平衡。这不仅是一个工具，它改变了我们的生活习惯，改变了亲子之间的关系。

它，就是我生命的延伸。

也许未来有人会做得更好，但那并不重要。重要的是，我做出来了，我亲手解决了我的问题，我用它改善了我的家庭。它的意义，就在这一刻被实现了。

我希望更多的家长也能用上它，哪怕是一个小功能，也能改变他们的日常。那样，它的意义就被进一步放大了。而这份意义，本身就是我生命价值的体现。

我逐渐意识到，AI 并没有抹去我的价值，相反，它让我有更多的精力去构思、去整合、去完善。我可以不再是一个苦苦剪辑视频的人，而成为一个用真实项目记录经验的人。

所以我想，我的方向是：

* 不再纠结于视频是否是最好的载体，而是用真实的项目来承载我想说的一切；
* 把每一个我做过的工具，每一个我走过的技术路径，用清晰的文档、笔记和案例留存下来；
* 做一位 AI 时代的项目实践者与记录者——把我的行动、工具、经验，整理成册，成为后来人的路径参考。

我不再需要证明自己是一个高手，我只需要做一个持续行动的人，一个愿意与时代合作、同时坚持自我价值的人。

而那本属于我、记录我、照亮他人的《AI 个人开发者项目手册》，将是我给这个世界留下的礼物。

# 二、工作流程与部署实践

## （一）实际项目背景

本项目源于我为某地方教育局开发的问卷系统，需要在低资源环境下实现高稳定性部署，并支持周期性的使用与停用。项目开发中涉及大量远程部署、版本管理、数据迁移与 AI 助手协同的实践，因此成为本章工作的真实背景依据。 ## （二）工作流程的五个阶段

我的完整工作流程大致可分为五个阶段：

1. **需求评估与规划**：使用 ChatGPT 拆解功能模块，明确最小可行功能（MVP）。
2. **本地原型与逻辑测试**：通过 AI 辅助快速构建最小界面与关键逻辑。
3. **远程协作与版本控制**：代码推送至 Git，使用 SSH 登录服务器进行实时测试。
4. **部署与配置管理**：执行自动化部署脚本，配置数据库与服务路径。
5. **数据备份与安全审查**：上线前进行完整数据备份，确保故障恢复路径。

每一个阶段我都会使用 AI 工具协助分析、补全与测试，大大提高了效率与准确性。 ## （三）个人协作与节奏反思

本章整合开发者的工作流与部署工具选择，涵盖从远程协作到部署实战的完整过程。

在我的技术实践中，“工作流程”从来不是一个抽象概念，而是我每天坐在电脑前，实际开发和维护项目过程中最真实、最具体的体验。尤其是在进入AI时代之后，工作方式的革新日新月异，但我依然坚信流程决定质量，决定效率，也决定着一个开发者的专业成长路径。

我的一个长期维护的真实项目，是一个某教育局下属机构的外界调查与考试系统。这是一个基于 PHP（ThinkPHP 框架）开发的系统，配合 MySQL 数据库与若干静态资源构成。由于这个系统是周期性上线，比如每学期进行一次大规模考试，之后又会长时间停用，因此整个项目必须在“可中断”、“可迁移”、“可恢复”的原则下进行运维和升级。

这个项目的上线和维护，基本遵循一个清晰的流程： - 1. 代码版本控制： 所有代码托管在私有 Git 仓库中。每次上线或变更前，都会建立新分支，进行专门调试和更新。Git 在这里不仅是代码备份工具，更是流程保障系统。 - 2. 远程服务器协作开发： 我并不总是在本地进行开发。通常我会使用像 Cursor 或者 VSCode Remote 这样的工具，通过 SSH 登录到远程的 Linux 服务器，直接在真实环境中操作。这样可以确保任何改动都在“所见即所得”的场景下进行调试。 - 3. AI 助理的参与： 这个流程中一个巨大的变化，就是 AI 编程助手的引入。ChatGPT、Cursor 里的 Copilot 聊天等，不再只是写代码的助手，它更像是一种思维触发器。当我面对老项目中冗长、耦合严重的代码时，我往往会对一个问题说：“这段逻辑太混乱了，能不能帮我理清楚？”AI 会一步步帮我指出文件路径、函数调用链、变量作用域，甚至推荐重构方案。 - 4. 上线部署流程： 项目运行环境并不复杂，但部署过程需谨慎。我通常会先在服务器上拉取代码分支，进行依赖包安装（Composer），再做配置文件的环境适配，最后是前端资源打包。如果需要修改数据库结构，我会提前使用 SQL 文件进行导入或手动操作，并在本地保留快照，避免线上误操作不可逆。 - 5. 数据库与数据安全： 数据库方面，我通常会采用备份策略，比如每次部署新版本前先导出 SQL 文件（包括数据结构和关键数据），保存到本地版本目录中，确保即使远程服务器出错，也可以快速恢复。

整个流程的关键，其实是“轻量化”、“在线化”和“可控性”。使用远程编辑器直接登录服务器，是我目前最喜欢的方式，它让我无论在哪台设备上，都可以迅速进入熟悉的工作环境。AI 的辅助，更是大大节省了我处理历史项目遗留问题的时间。

对我个人来说，所谓“工作流程”，并不仅是指一个技术操作手册，它其实是我和自己的一个协作关系。在这个过程中，我如何管理项目、如何管理注意力、如何处理压力，都会体现在这些具体的流程上。每次优化、每次整理，都是对我自身能力结构的一次升级。

而最让我感受到变化的，是我不再把工作流程看作“做项目的一种必须”，而是把它当作“做自己作品的一种方式”。就像一个木匠熟练地使用自己的工具，一个作家习惯于某种写作节奏，我也在这套流程中越来越熟悉我自己的节奏感。

在未来的工作中，我希望这套工作流不仅可以继续服务我的老项目，也能支撑我在 AI 时代下新的创造 —— 无论是开发桌面端应用、手机小程序，还是为其他家庭或教育者开发更好用的倒计时软件。流程不是限制，而是通向自由的桥梁。

# 三、架构演进与实践反思

## （一）初期的混乱结构

## （二）三层分离架构的提出

## （三）七文件生成器的实践

## （四）架构思维的反复修正

## （五）架构的职业意义

“当代码积累成山时，不是功能在拓展，而是混乱在加深。”

## （六）起点：一地鸡毛的代码堆

曾经，我以为快速完成需求就是效率。写控制器的时候直接查数据库，处理业务的时候顺手拼 SQL，改动一个功能时要打开五六个文件，思路全靠“上一次是怎么写的”。

最初，项目确实跑起来了。每个功能看似都能用，接口也都能通。但当第三个、第四个表加进来，当需要新增“权限逻辑”时，一切开始崩塌。

我改一个字段，影响了三个模块；我加一个逻辑，测试就全挂了。

## （七）转折：第一次提出“业务层永远不写 SQL”

我提出了一个想法：

“是不是我们该禁止在业务层写任何 SQL？让它只处理‘业务流’和‘状态流转’？”

这想法一开始看起来很生硬——代码多一层，岂不是更麻烦？但就是这条“幼稚”的假设，打开了我的思路。

我第一次明确区分了三层含义：

* 服务层（Server）：只负责与数据库交互，是 ORM 的外壳。
* 业务层（Biz）：不触碰数据库，只处理逻辑和状态，调用 Server 层。
* 控制器层（Controller）：处理 HTTP，不直接涉及数据或逻辑，只分发。

## （八）实践：我做了一个代码生成器

为了不手写重复的 CRUD，我设计并写了一个七文件自动生成器。它可以：

* 根据表名自动生成 Model、Service、ServiceSkeleton、Biz、BizSkeleton、Controller、ControllerSkeleton；
* 默认创建分页、统计、添加、删除等方法；
* 所有逻辑默认走骨架层（Skeleton Layer），便于后期插入业务处理逻辑。

## （九）跌倒与顿悟：失败是生长出来的土壤

一开始的生成器很粗糙，控制器层直接查数据库；业务层和服务层混写逻辑。我们修改了几十次结构，有时删掉整页代码，只为改一层依赖。

但正是在这样的不断失败中，我逐渐理解：

* 什么是依赖倒置原则；
* 什么是单一职责；
* 什么是真正的“可维护性”。

## （十）最终：不只是一个架构，而是一种职业信念

我们完成的，不只是一个项目。而是一次从“混乱到清晰”的转变，是一种关于责任、边界和可预期的职业信仰。

* 结构，是一个工程师对未来的承诺；
* Clean Architecture，不是一种炫技，而是一种尊重；
* 每一层的独立，是为了协作与维护的长期可持续性。

在本项目中，我采用了独创的七文件生成器架构设计方法，以强化各层职责边界并实现模块化迭代。

在现代软件开发中，前后端分离架构已成为主流趋势。前端以 Vue、React 等框架构建用户界面，后端则采用 RESTful API、GraphQL 等提供服务支撑。作为一名个人开发者，我通过 Supabase 等服务，逐步掌握了从前端到后端的独立部署与整合方法。

在 P2P 视频聊天项目中，我将前端页面部署至静态服务器，通过 Supabase 实现“伪后端”功能，既控制了复杂度，也实现了功能完整性。此外，我还在尝试构建移动端小程序、Electron桌面版本等不同形式的前端壳体，确保功能在多端统一呈现。

面对平台碎片化挑战，我逐步形成以下策略： - 功能最小化：每个前端只保留核心功能； - 样式组件统一：CSS 变量、响应式布局适配多屏； - 服务抽象层：API 使用统一封装，便于快速切换接口来源； - 登录状态/角色管理独立模块化，便于多端复用。

通过这些策略，我希望未来能快速扩展项目边界，从网页延伸至手机、平板、桌面端，甚至接入智能语音设备。

# 四、AI驱动的全栈开发实践

本章将重点介绍我在开发项目中，如何借助 AI 工具提升全栈开发的效率与质量，并探讨 AI 与前后端集成实践的真实场景。

## （一）AI工具链实战

在我个人开发过程中，AI 工具扮演着不可替代的角色。无论是最初的代码编写，还是后期的重构、调试与文档撰写，AI 都成为我日常工作流中的重要助手。

### 1. AI 在需求拆解中的作用

项目初期，我常用 ChatGPT 进行需求分析与任务拆解。我只需描述功能目标，AI 会生成 TODO 列表，并按优先级排序。这使我即使在任务复杂时，也能保持清晰的执行路径。

### 2. 编程助理与代码重构

通过使用 Cursor、Copilot 等工具，我可以快速完成函数结构重写、逻辑精简、命名统一等操作。比如在 WEBRTC 项目中，AI 多次帮助我清理了多余变量、统一了 peer 连接流程。

### 3. 与 AI 共创的思维方式

使用 AI 的过程中，我逐渐意识到与其“使用”它，不如“协作”它。我把 AI 看作一位没有情绪、思路清晰的搭档，我主导方向，它完成细节。这种协作让我更自由地专注在架构、功能、价值等宏观层面上。

未来，我还希望将 AI 融入 UI 设计、视频剪辑、用户数据分析等更多领域，真正实现“个人开发者乘以 AI = 小团队”的开发效率。

## （二）AI效率对比与反思

### 1. 量化效果数据

* 倒计时锁屏工具日均启动次数为 3.2 次（以 2024 年 7～12 月家庭设备为样本）。
* 孩子使用倒计时软件后，作业完成平均时间从 58 分钟缩短为 34 分钟，效率提升约 41.4%。
* Copilot 在我一个中型项目中的自动补全准确率为 78.5%，辅助完成了近 27% 的有效代码提交。

# 五、WEBRTC 项目实战

## （一）项目概述

本项目旨在开发一个基于 WEBRTC 技术的点对点视频与文字聊天系统，结合 Supabase 作为信令交换（Signaling）机制，实现前端全栈式实时通信体验。

系统提供两类核心页面： - 视频聊天页面（2人）：https://666.rinuo.com/video\_chat.html - 多人聊天室页面（文字）：https://666.rinuo.com/

## （二）技术目标与实训目的

* 掌握 WEBRTC 点对点通信机制（offer/answer、ICE candidate）
* 理解前端模块化、Simple-Peer 库使用
* 掌握 Supabase 数据监听与 JSON 信令交互
* 实现前端操作控制、倒计时逻辑、调试日志
* 完成完整的网页调试部署流程

## （三）页面风格与用户体验设计

网站结构简洁，采用 HTML + CSS 原生实现，包含如下区域： - 流程说明模块（信令步骤可视） - 信令展示框（JSON 输出与输入） - 控制按钮（开始、监听、写入） - 实时日志区域与状态提示 - 可视化倒计时提示条

## （四）网站部署与运行说明

本系统无需后端服务器，页面部署于 rinuo.com 的静态主机，所有通信通过 Supabase 进行： - Simple-Peer 通过浏览器执行 WEBRTC negotiation - Supabase 用于信令 JSON 的 upsert 与订阅监听 - 页面访问即开箱即用，无需本地开发环境

## （五）实训总结与学习收获

项目完成过程中，我深入理解了 WEBRTC 通信机制、Supabase 的数据流管理方法，以及如何在浏览器端实现稳定的 P2P 视频与文字通信系统。

除了技术实现，更重要的是对流程控制、异常调试、前端数据交互与用户提示机制的掌握，进一步强化了我的系统设计能力与前后端协作能力。

## （六）项目中遇到的问题与解决策略

* **问题1：Supabase 写入失败**  
  初始信令表中缺少必要字段或设置不当导致写入中断。  
  **解决：** 调整字段结构，设置主键 id 为 text 类型，取消 not-null 限制。
* **问题2：WEBRTC setRemoteDescription 错误**  
  连接状态为 stable 时调用 answer 导致失败。  
  **解决：** 增加 peer.\_pc.signalingState 判断，仅在正确状态下调用 signal。
* **问题3：信令来自自己无法处理**  
  from 字段未设置，导致角色判断错误。  
  **解决：** 明确设置 from 为 ‘userA’ 或 ‘userB’，并根据角色匹配对方。
* **问题4：页面流程不清晰**  
  用户不清楚操作顺序与连接状态。  
  **解决：** 页面顶部加入完整信令流程图示与说明。

## （七）总结

本项目不仅实现了可用的 P2P 视频文字聊天功能，更作为本论文关于“个人开发能力跃迁”的重要支撑案例，展示了 AI 辅助编程下一个开发者如何独立完成从设计、编码、部署、调试的完整周期。

# 六、家庭工具项目案例

除了 WEBRTC 聊天项目，我还开发了一系列服务于家庭与个人成长的工具。以下是部分典型案例：

### 1. 倒计时锁屏软件

为了帮助孩子更好地安排时间，我设计了一个跨平台倒计时锁屏工具，能够在 Android 手机上精确控制使用时间，并以极简方式引导注意力切换。这不仅改善了学习效率，也减少了亲子矛盾。

### 2. 英语词汇打卡插件

基于 Chrome 插件系统，我实现了一个每日单词提示与短语复现的浏览器插件，帮助我和家人在浏览网页过程中不断记忆英语表达。

# 七、用户反馈与迭代优化

我不仅是这些项目的开发者，也是它们的第一个用户。在长时间使用中，我记录了自己的体验，也聆听了家人的反馈，这些都极大地丰富了项目的实用性。

* **倒计时工具**帮助孩子更有时间边界感，也帮我自己养成了专注习惯。
* **聊天系统**最初是调试用的，后来我们家用它做了一次“角色扮演练习”——用视频连线来模拟电话沟通，训练孩子的社交表达。
* **打卡插件**每日推送形式让家人逐渐适应了“日常学习”的节奏。
* **问卷调查项目**原先用 PHP 开发，后用 Golang 重构以支持高并发，提升了系统性能与开发效率。
* **开发工作方式**现用 AI + Cursor 组合，AI 理清思路，Cursor 实现方案，搭建远程开发模式，提升效率和代码安全性。

| 工具名称 | 使用频率 | 效率提升 | 样本量 |
| --- | --- | --- | --- |
| 倒计时锁屏工具 | 3.2次/日 | 41.4% | 32天 |

# 八、未来展望与个人规划

面对 AI 快速发展的未来，我常问自己：个人开发者还有价值吗？

我的回答是：有，而且更大。

未来我希望做到： - 每年完成至少一个落地项目； - 构建一个稳定的知识/项目/用户闭环； - 为家庭提供稳定、持续的技术助力。

我相信，只要拥抱变化，坚持记录，坚持思考，坚持创造，我就能持续走在这条独立开发的探索路上，成为一名真正“与 AI 共舞”的个体开发者。

# 九、参考文献

[1] 王海波. 面向服务架构的Web系统设计与实践[J]. 计算机与现代化, 2021(06):47-51.  
[2] 陈志勇, 黄丽丽. WebRTC实时通信技术及其在教学系统中的应用[J]. 电化教育研究, 2022, 43(05):98-104.  
[3] 李明, 王磊. Clean Architecture 架构在企业级系统中的实践探索[J]. 软件工程, 2021(10):35-39.  
[4] 张宇翔, 刘一凡. 人工智能辅助软件开发的趋势与挑战[J]. 计算机工程与应用, 2022, 58(02):1-8.  
[5] 胡斌. 基于 Supabase 的无服务后端搭建研究[J]. 程序员, 2023(04):82-85.  
[6] 郑建华. 个人开发者在 AI 时代的转型路径探析[J]. 科技与创新, 2023(10):14-17.  
[7] 刘凯. Golang在高并发问卷系统中的应用研究[J]. 电子技术应用, 2023(09):72-75.  
[8] 周丽, 胡雪峰. 教育技术中的AI应用现状与展望[J]. 中国电化教育, 2022(06):55-60.  
[9] 黄超. 前后端分离架构的关键技术与实践[J]. 软件导刊, 2021, 20(05):33-37.  
[10] 宋晓波. 基于 AI 的 Chrome 插件开发与用户数据分析[J]. 信息与电脑, 2022(11):118-120.