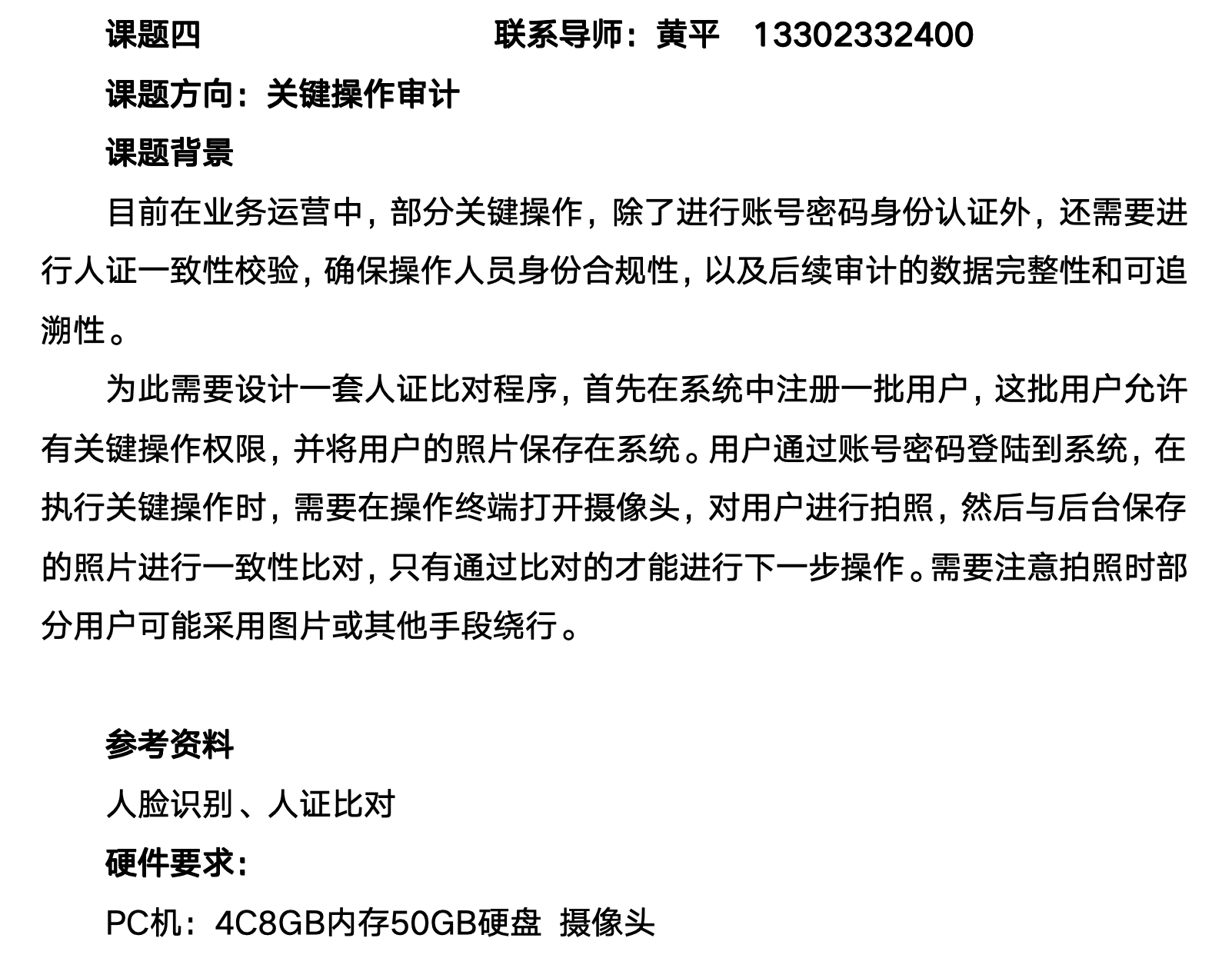
软件工程课程设计\_工作说明



1. 可行性分析

**技术可行性**：

* 人脸识别技术成熟，可使用开源库（如OpenCV、Dlib）或云服务（如阿里云、腾讯云人脸识别API）。
* 防伪机制（如活体检测）可通过检测眼部眨动、头部运动或深度摄像头实现。
* 审计日志可基于数据库（如MySQL）或文件系统实现。

**资源可行性**：

* 硬件：PC（4核，8GB内存，50GB硬盘）和摄像头满足要求。
* 软件：编程语言可选Python（配合OpenCV）、Java或C++，数据库可选用MySQL或SQLite。
* 网络：联网环境支持云API调用和参考资料查阅。

1. 项目计划

项目组长召集小组成员，针对所选项目讨论开发计划，进一步明确项目的大小，范围，复杂程度等。并进行任务分解与任务分配，确定任务进度，确定完成项目的环境和工具。

**项目规模和范围**：

* 核心功能：用户注册/登录、照片存储、实时拍照、人脸比对、防伪验证、审计日志。
* 非核心功能（视时间）：用户管理界面、操作统计报表。
* 规模：中小型系统，适合3-4人小组，核心功能约2000-3000行代码。

**任务分解与分配**：

* 任务1：用户注册与登录模块（数据库设计、密码加密）。
* 任务2：照片存储与管理（文件系统或数据库存储）。
* 任务3：人脸识别与比对（集成API或算法）。
* 任务4：防伪机制（活体检测或简单动作验证）。
* 任务5：审计日志（记录操作时间、用户、结果）。
* 任务6：测试与调试（正常和异常场景）。

**开发环境**：

* 编程语言：Python（推荐，易于集成OpenCV）、Java或C++。
* 框架/库：OpenCV、Dlib或云API（Face++、AWS Rekognition）。
* 数据库：MySQL或SQLite存储用户数据和日志。
* 工具：Git（代码管理）、PyCharm/VSCode（IDE）、Postman（API测试）。

1. 需求分析

按照项目任务分配结果和项目进度要求，由任务承担人开展项目的需求分析的主要工作,不得超越前述项目的大小，范围，复杂程度等。项目小组的其他成员配合任务承担人，完成项目的需求分析。

**功能需求**：

1. **用户注册**：输入用户名、密码，上传照片，存储到数据库。
2. **用户登录**：输入用户名、密码，验证身份。
3. **关键操作验证**：用户选择关键操作，系统启动摄像头，拍摄照片，与存储照片比对。
4. **防伪机制**：检测照片是否为实时拍摄（e.g.,要求用户眨眼或转头）。
5. **审计日志**：记录操作时间、用户ID、操作类型、比对结果。
6. **错误处理**：处理摄像头不可用、光线不足、比对失败等情况。

**非功能需求**：

* 性能：人脸比对响应时间<2秒，系统支持至少100个注册用户。
* 安全性：密码加密存储（如使用bcrypt），照片数据加密传输。
* 可用性：界面简洁，支持PC端操作。

1. 概要设计

按照项目任务分配结果和项目进度要求并依据《需求分析规格说明》，由任务承担人完成项目的概要设计任务，项目小组的其他成员配合任务承担人，完项目的概要设计任务。

**系统架构**：

* **前端**：简单Web界面（HTML+CSS+JavaScript，或使用框架如Flask/Django），显示登录、注册、操作界面。
* **后端**：处理用户认证、照片存储、人脸比对、日志记录（Python Flask+MySQL）。
* **人脸识别模块**：调用OpenCV/Dlib或云API进行比对。
* **摄像头模块**：捕获实时图像，支持活体检测。
* **数据库**：存储用户数据（用户名、加密密码、照片路径）、操作日志。

**模块划分**：

1. 用户管理模块：注册、登录、照片上传。
2. 人脸比对模块：实时拍照、特征提取、比对。
3. 防伪模块：活体检测或动作验证。
4. 审计模块：记录操作日志。
5. 错误处理模块：处理异常情况。

**接口定义**：

* 注册接口：POST /register {username, password, photo}
* 登录接口：POST /login {username, password}
* 比对接口：POST /verify {user\_id, live\_photo}
* 日志接口：GET /logs {user\_id, time\_range}

1. 详细设计

按照项目任务分配结果和项目进度要求并依据《概要设计文档》由任务承担人完成项目的详细设计任务。项目小组的其他成员配合任务承担人完成项目的详细设计任务。

**用户管理模块**：

* 数据库表：Users (user\_id, username, hashed\_password, photo\_path)。
* 注册流程：用户输入信息，密码用bcrypt加密，照片存储到文件系统或数据库。
* 登录流程：验证用户名和bcrypt加密密码。

**人脸比对模块**：

* 算法：使用OpenCV人脸检测（Haar级联或DNN）+特征提取（FaceNet或Dlib）。
* 流程：捕获实时照片，提取特征向量，与存储照片的特征向量计算相似度（余弦距离或欧氏距离）。
* 阈值：相似度>0.9视为通过（可调）。

**防伪模块**：

* 方法1：活体检测，要求用户完成指定动作（如眨眼、转头），使用OpenCV跟踪眼部或头部运动。
* 方法2：检测图像来源，确保来自摄像头而非静态图片。

**审计模块**：

* 数据库表：Logs (log\_id, user\_id, operation, timestamp, result)。
* 记录每次操作的详细信息，定期清理旧日志。

**错误处理**：

* 异常场景：摄像头断开、光线不足、比对失败。
* 提示用户：友好错误信息，重试机制。

1. 编码

此阶段的主要任务是根据详细设计说明书，选定一门熟悉的编程语言（如：C++、JAVA或其他的）进行编码。要求编码过程中，应该依据编码规范，命名规范等相关规范开展编码工作，源代码需有适当的注释。

**实现步骤**：

1. 搭建Web框架（Flask），实现注册/登录页面。
2. 集成数据库（MySQL），存储用户数据和日志。
3. 实现照片上传和存储（文件系统或数据库BLOB）。
4. 集成OpenCV，捕获摄像头照片，提取人脸特征。
5. 实现人脸比对逻辑，设置相似度阈值。
6. 增加防伪功能（如眨眼检测）。
7. 实现审计日志，记录每次操作。
8. 测试

要求学习并掌握开发环境所提供的调试工具，能够阅读在线帮助。在此基础之上，应根据系统的需求设计相应的测试数据方案（特别是一些异常情况的处理），最终完成该系统的测试和调试任务。并要写出相应的测试说明书。

**测试用例**：

1. **功能测试**：
   1. 注册：测试正常注册、重复用户名、照片格式错误。
   2. 登录：测试正确/错误密码、账户不存在。
   3. 人脸比对：测试相同用户照片、不同用户照片、光线变化。
   4. 防伪：测试静态图片、视频重放、正常活体动作。
   5. 审计：验证日志记录是否准确。
2. **异常测试**：
   1. 摄像头断开：提示用户重新连接。
   2. 光线不足：提示用户调整环境。
   3. 网络中断（若用云API）：切换本地算法或提示错误。
3. **性能测试**：
   1. 比对时间：<2秒。
   2. 系统支持100个用户注册和并发操作。