一、项目概述

项目名称：
校园体育场馆预约与信用管理系统

核心目标：
为校内学生和教师提供统一的体育场馆预约平台，实现在线预约、违约信用管理、场馆使用数据统计与可视化，并通过网络爬虫联动天气与闭馆通知，辅助科学管理与公平使用。

与课程要求的对应关系：

《Python 程序设计》课程实践指导书

- 有完整用户界面（GUI 界面）
- 使用数据库（如 SQLite/MySQL）存储用户、场地、预约、信用等数据
- 通过网络模块实现远程登录访问（Flask/Django 或 socket）
- 支持用户登录/注销，不同身份权限区分（普通用户/管理员）
- 附加功能：爬虫获取天气或学校通知、数据统计与可视化（热力图等）

二、系统角色与典型场景

1. 角色

1）普通用户（学生/教师）

- 注册、登录、退出
- 查看场馆开放时段、实时空闲情况
- 提交/查看/取消预约
- 查看个人信用分与违约记录
- 查看个人运动统计（次数、时长图表）
- 教师特权：教师端（体育老师）可以提供课表，课表中对应的时间段的对应场地自动预约并不能被其他人预约。

2）管理员（体育部/场馆管理员）

- 管理场馆与场地信息
- 配置开放时间段与可预约容量
- 审核或强制取消预约
- 查看场馆使用统计报表与热力图

- 管理违约记录与黑名单
- 管理系统公告（例如：闭馆、赛事占用）

三、功能模块设计

模块 0：系统架构与技术栈建议

推荐技术栈（简洁可行）：

- 应用框架：Flask（轻量 Web 框架，课程文档允许）

  《Python 程序设计》课程实践指导书

- 前端界面：（GUI 路线）PyQt+ socket 服务器
- 数据库：SQLite（开发简单、跨平台、内置支持）
- 爬虫：requests + BeautifulSoup
- 可视化：matplotlib 生成图片文件，在页面中展示
- 网络：Flask 本身就是 HTTP 服务器，天然支持远程访问

模块 1：用户与权限管理

1.1 用户注册/登录/注销

- 注册字段：
  - 学号/工号（唯一）、姓名、密码（加密存储）、身份（学生/教师）、联系方式
- 登录：
  - 基于学号/工号 + 密码，成功后创建会话（session）
- 注销：
  - 清除会话信息

1.2 权限控制

- 未登录：只能浏览场馆基础信息，不可预约
- 普通用户：预约、查看自己记录、行使职能
- 管理员：拥有场馆维护、统计分析、公告管理权限

模块 2：场馆与时间段管理

2.1 场馆信息

- 场馆类型：篮球馆、羽毛球馆、足球场、游泳馆等
- 每个场馆可有多个“场地/区域”（如羽毛球 1–10 号场）

2.2 开放时间段

- 每个场地配置：
  - 日期 / 可重复规则（如每周一至周五 18:00–22:00）
  - 切分为固定时间段（如每段 1 小时）

2.3 管理员功能

- 添加/编辑/删除场馆与场地
- 配置或修改场馆开放时间
- 设置每个时间段可预约人数上限

模块 3：预约与冲突/公平机制

3.1 预约核心流程

- 用户选择：
  - 场馆 → 场地 → 日期 → 时间段
- 系统检查：
  1. 是否在允许的预约时间范围内
  2. 用户在该时间段有没有其他冲突预约
  3. 场地该时间段剩余名额是否足够
  4. 用户信用分是否低于限制（如 <60 不允许预约热门时段）
- 若全部通过，则生成一条预约记录，状态为“已确认”。

3.2 热门时间段的公平分配（拓展亮点）

- 对指定热门时段（如工作日 19:00–21:00）采用“抽签/排队”机制：
  - 在一个“报名截止时间”前所有人均可报名
  - 截止后系统统一分配：
    - 把所有报名用户放入列表
    - 随机洗牌（或按信用分加权）后取前 N 人成功
    - 其他人标记为“未中签”

可在管理端提供“抽签运行”按钮，或设定自动运行逻辑。

模块 4：违约与信用分管理（特色功能）

4.1 违约定义

- 预约未提前取消且未使用 → 标记为“爽约”
- 取消时间距离预约开始时间过短（如 <1 小时） → 视为“迟取消”

4.2 信用分机制（示例）

- 初始信用分：100 分
- 每次爽约：扣 10 分
- 每次迟取消：扣 5 分
- 若一个月内正常使用次数 ≥N，则加 5 分（上限 100）

4.3 信用影响规则

- 信用 < 60：
  - 禁止预约周末或热门时段
- 信用 < 40：
  - 一周内只能预约一次，且不可预约热门场地
- 管理员可查看“低信用用户列表”，手动调整。

模块 5：统计分析与可视化

5.1 场馆利用率分析

- 维度：
  - 按场馆/场地统计：预约率、占用时长
  - 按时间段统计：一周中哪几天/哪一时段最火
- 输出：
  - 柱状图：场馆预约次数对比
  - 热力图：横轴星期，纵轴时间段，颜色表示预约率

5.2 用户个人运动统计

- 每位用户：
  - 最近 X 周的运动次数折线图
  - 最常去场馆 Top3
- 可以在“个人中心”展示，对应加分的“数据可视化”。

模块 6：爬虫与外部信息联动（差异化亮点）

6.1 天气爬虫

- 选择某天气网站的城市 7 日天气预报页面
- 使用 requests 请求，BeautifulSoup 解析：
  - 日期、天气情况（晴/雨/雪）、温度范围
- 将天气信息写入本地数据库的 weather_info 表

使用方式：

- 在预约界面，对户外场地：
  - 若当天为中雨/大雨，显示醒目提示“可能受天气影响”
  - 或限制预约某些户外场地

6.2 校园公告爬虫（可选）

- 定期访问学校体育部/教务处通知页面
- 搜索关键词如“体育馆”“闭馆”“停用”等
- 如果有相关公告，则将该日期对应场馆自动标记为“关闭”，并同步到预约系统中显示为不可预约。

四、数据库设计（可直接按此建表）

以 SQLite 为例，给出主要表结构（字段可再细化）：

1. 用户表 users

- user_id (PK, int, auto)
- username (varchar, 登录账号，学号/工号)
- password (varchar, 加密后)
- name (varchar, 真实姓名)
- role (varchar, 'student'/'teacher'/'admin')
- phone (varchar)
- credit_score (int, default 100)
- create_time (datetime)

2. 场馆表 venues

- venue_id (PK)
- venue_name
- venue_type (如 'basketball', 'badminton')
- is_outdoor (bool)
- location
- description

3. 场地表 courts

- court_id (PK)
- venue_id (FK → venues)
- court_name（如“羽毛球场 1 号”）
- capacity（此时段可容纳多少人）

4. 时间段表 time_slots

- slot_id (PK)
- court_id (FK)
- date (date)
- start_time (time)
- end_time (time)
- max_reservations (int)
- current_reservations (int)
- is_hot (bool, 是否热门时段)

若你们想简化，也可以不单独建时间段表，而是直接在预约表中记录时间区间，但有时间段表更规范。

5. 预约表 reservations

- reservation_id (PK)
- user_id (FK)
- slot_id (FK)
- status (varchar: 'confirmed'/'cancelled'/'no_show'/'pending_lottery')
- create_time (datetime)
- cancel_time (datetime, 可空)

6. 信用记录表 credit_logs

- log_id (PK)

- user_id (FK)
- change_amount (int, 正数加分/负数扣分)
- reason (varchar, 如“爽约扣分”“完成多次运动加分”)
- time (datetime)

7. 公告表 announcements

- announcement_id (PK)
- title
- content
- related_venue_id (可空)
- start_date / end_date
- create_time

8. 天气表 weather_info

- weather_id (PK)
- date (date)
- weather_text (varchar, 如“中雨”)
- temp_low / temp_high
- update_time

五、核心业务流程（可作为时序图/用例图基础）

流程 1：用户预约场地

1. 用户登录 → 进入“场馆预约”页面
2. 选择日期、场馆、场地 → 系统查询 time_slots 中可预约的时间段
3. 用户选中时间段 → 系统检查：
    - 用户信用分
    - 是否已有冲突预约
    - current_reservations < max_reservations
4. 检查通过 → 写入 reservations，更新 current_reservations +1
5. 返回预约成功，并提示天气/公告信息。

流程 2：管理员查看使用统计

1. 管理员登录 → 进入“统计分析”页面

2. 选择时间范围、场馆
3. 系统在 reservations、time_slots 上进行统计
4. 用 matplotlib 绘图 → 生成 png → 在页面展示
5. 管理员可导出数据或查看不同维度报表。

流程 3：违约处理定时任务（可简化为每天手动按钮）

1. 到某一时间点（如每天晚上），程序扫描当天所有仍为“confirmed”的预约：
    - 若当前时间已超过预约结束时间，但没有签到记录（可以不做签到，简单认为都未标记为“使用完毕”）→ 视为爽约
2. 将这些预约状态改为 no_show，为其对应用户：
    - credit_score -= 10
    - 在 credit_logs 中新增一条记录

六、组内分工示例（验收时方便讲）

假设 3 人一组：

- A：后端与数据库设计
    - 设计并实现所有表结构
    - 编写用户/场馆/预约相关的接口
    - 实现信用与违约机制逻辑
- B：前端与可视化
    - 负责所有页面（登录、预约、列表、统计）
    - 集成 matplotlib 图表展示
    - 简单美化界面，保证易用
- C：爬虫与统计分析
    - 完成天气/公告爬虫模块
    - 与数据库对接，写入 weather_info、announcements
    - 编写统计逻辑（场馆利用率、个人运动数据）

验收时可以明确说明每个人主要负责模块与对应亮点。