数据库系统原理课程设计说明书

课程名称 ： **数据库系统原理**
学号 ： **21281280**
姓名 ： **柯劲帆**
班级 ： **物联网2101班**
指导老师 ： **郝爽**
修改日期 ： **2024年6月19日**

1. 系统规划与可行性分析报告

1.1. 系统名称

KJF航班订票系统 (KJF's FlightBooking System)

1.2. 组织架构图及相关业务部门

组织架构图：

```
航空公司
├── 运营部门
|    ├── 航班管理
|    ├── 机场管理
|    └── 票务管理
├── IT 部门
|    ├── 系统开发
|    ├── 数据库管理
|    ├── 应用支持
|    └── 安全与合规
└── 客户服务部门
     ├── 客服中心
     └── 技术支持
```

涉及的相关业务部门：

- **运营部门**：负责航班和机场数据的管理、维护和更新。
- **IT 部门**：负责系统的开发、维护和安全管理，确保系统稳定运行。
- **客户服务部门**：提供客户支持，解决用户在使用系统过程中遇到的问题。

1. 用户使用系统开展业务的场景

   用户（旅客）：

   1. **注册**：用户在飞行订票系统的注册页面填写个人信息，完成账号注册。**
   2. **查询航班**：登录后，用户在系统中输入出发地、目的地和日期，查询可用航班和座位级别信息。
   3. **预订机票**：选定航班和座位级别后，用户通过系统完成支付，生成电子机票，并在行程日期前往机场使用电子机票登机。

   运营部门员工：

   1. **登录**：员工使用分配的账号登录系统。
   2. **管理航班和机场信息**：员工进入后台管理界面，添加或修改航班信息（如航班编号、起飞时间、到达时间、座位级别等）和机场信息（如机场名称、所在国家和城市等）。

2. 系统性能指标

   - **并发用户数**：支持同时在线用户数量 50,000 人。
   - **总用户数**：系统应支持注册用户数量 10,000,000 人。
   - **核心业务响应时间**：查询航班和预订机票的响应时间应不超过 1 秒。
   - **数据更新频率**：航班和机场数据应能在5分钟内更新。

3. 系统的战略地位

飞行订票系统作为航空公司的核心业务系统，其战略地位体现在以下几个方面：

- **客户服务**：提供便捷的在线订票服务，提高客户满意度，增强客户忠诚度。
- **运营效率**：通过电子化管理航班和机场数据，提高航空运营的效率和准确性。
- **市场地位**：占据航空票务市场的主要份额，增强在交通出行领域的市场竞争力。
- **收益**：通过在线售票获得直接收益，同时通过提升运营效率间接节约成本。

4. 投资和运营成本及收益分析

投资成本：

- **硬件投入**：服务器、存储设备、网络设备等。
- **软件投入**：数据库软件、应用服务器软件、安全防护软件等。
- **开发投入**：系统开发费用，包括需求分析、设计、编码、测试等。
- **培训费用**：对员工的培训费用。

运营成本：

- **维护成本**：系统的日常维护，包括硬件维护、软件升级、数据备份等。
- **人力成本**：技术支持人员、客户服务人员、运营管理人员等的工资。
- **安全成本**：系统安全防护，包括防火墙、入侵检测、防病毒等。

收益分析：

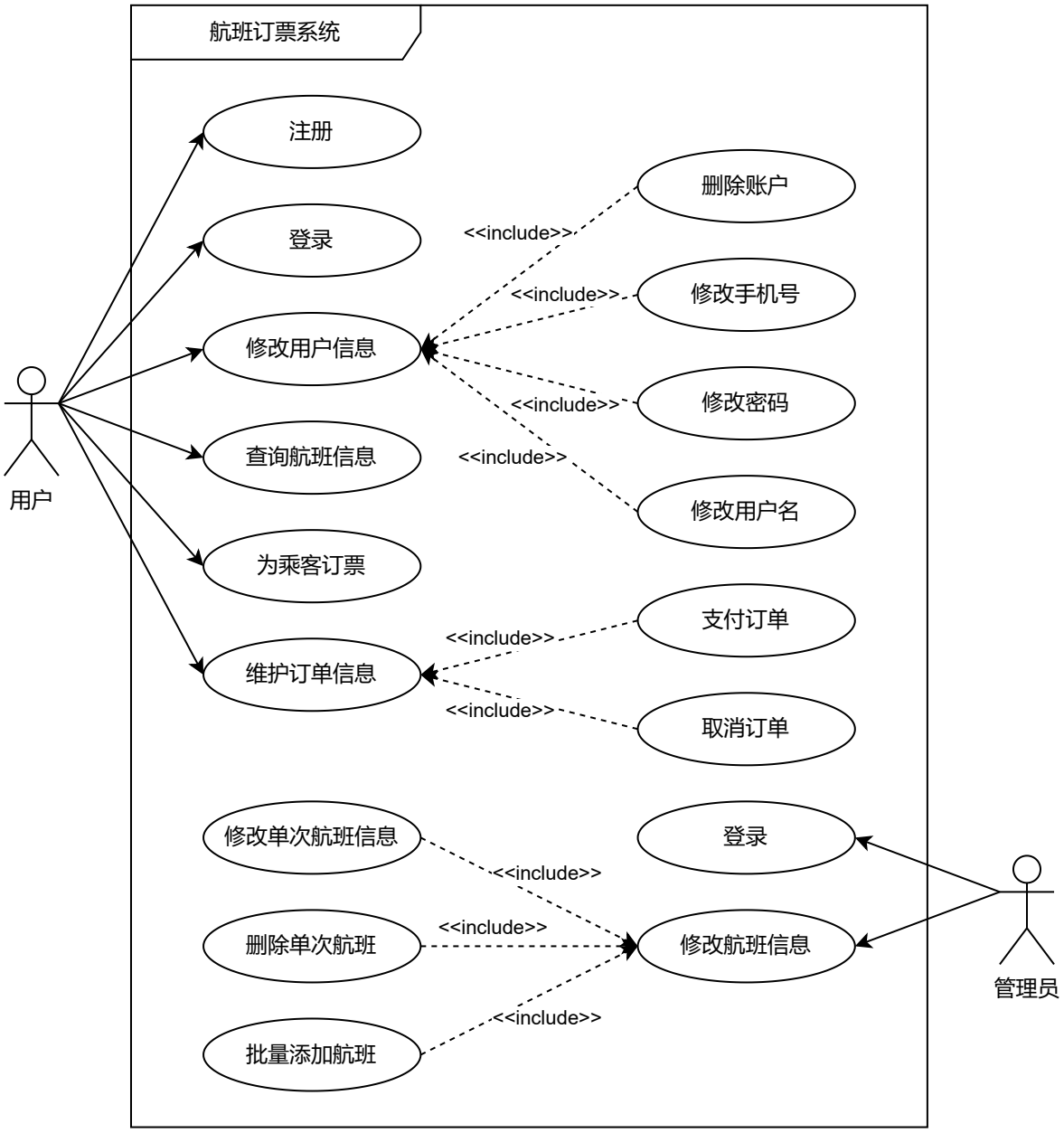
- **直接收益**：机票销售收入。
- **间接收益**：提高运营效率，降低人力成本和错误率，提升客户满意度和忠诚度。

5. 技术选型规划

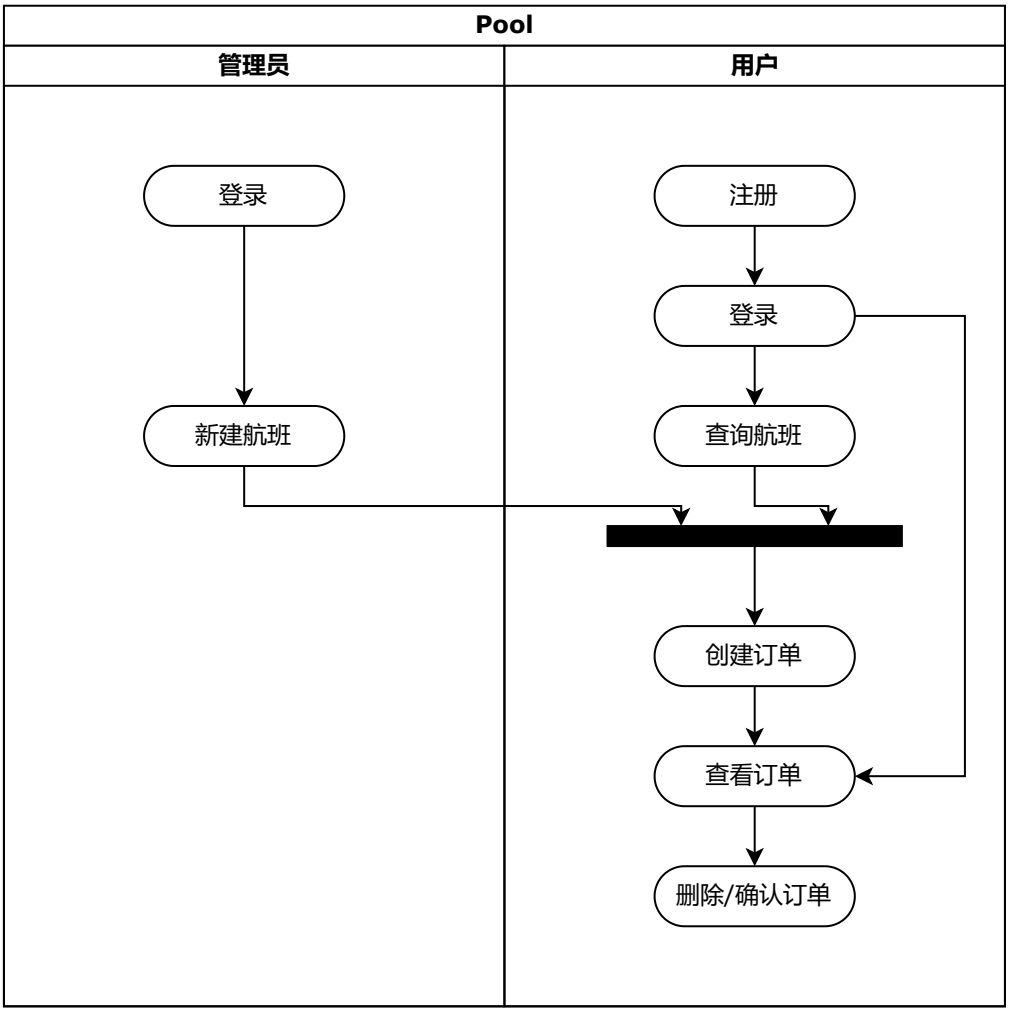
- **数据库**：选用高性能的关系型数据库，如 MySQL、PostgreSQL，用于存储用户、机场、航班和机票数据。
- **应用服务器**：采用分布式架构，使用 Java EE、Spring Boot 等技术实现系统的业务逻辑。
- **前端技术**：使用 React、Vue.js 等框架开发用户界面，提供良好的用户体验。
- **安全技术**：部署 SSL 证书，采用 OAuth 2.0 进行用户认证，使用防火墙、入侵检测系统等保护系统安全。

2. 系统需求规格说明书

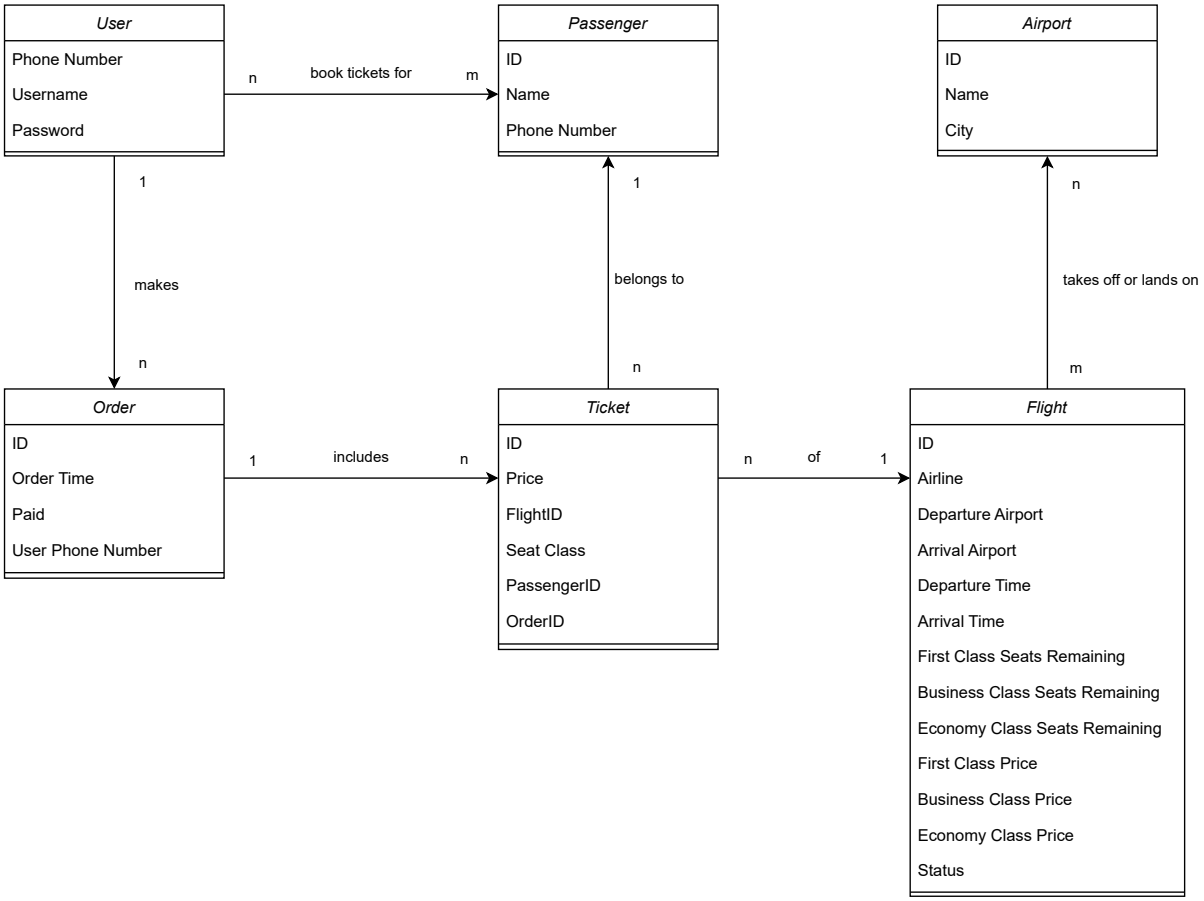
2.1. 用例图



2.2. 泳道图



2.3. 数据结构

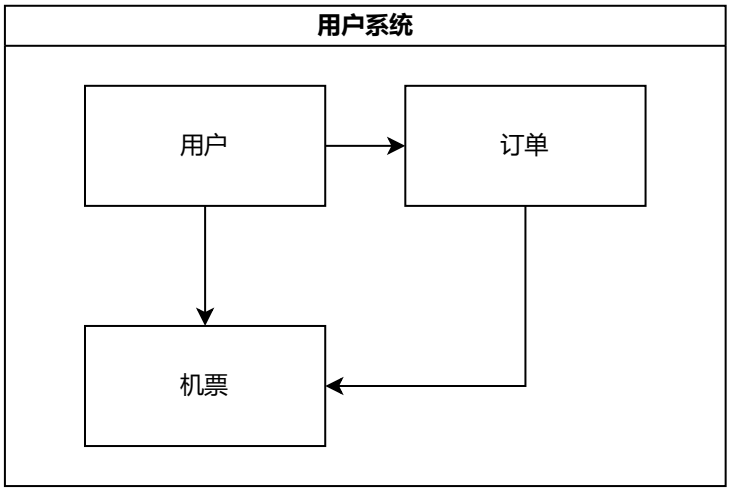


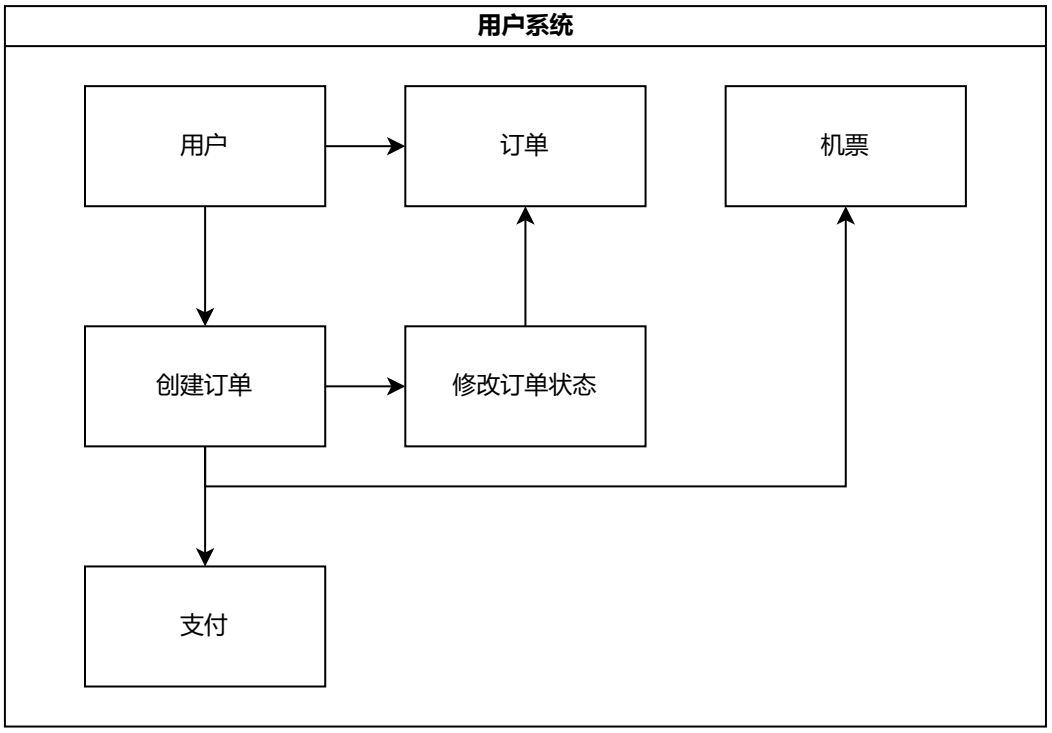
数据字典

- **Passengers 表**
  - ID: 身份证号，主键，18位，正则验证。
  - Name: 姓名，必填，长度255。
  - Phone_number: 手机号，必填，长度11，正则验证。
- **Users 表**
  - Phone_number: 手机号，主键，长度11，正则验证。
  - Username: 用户名，必填，长度255。
  - Password: 密码，必填，长度255。
- **Airports 表**
  - ID: 机场三字码，主键，长度3。
  - Name: 机场名称，必填，唯一，长度255。
  - City: 城市，必填，长度255。
- **Flights 表**
  - ID: 航班编号，主键，长度255。
  - Airline: 航空公司，必填，长度255。
  - Departure_airport: 出发机场，必填，长度3，外键。
  - Arrival_airport: 到达机场，必填，长度3，外键。
  - Departure_time: 出发时间，必填，日期时间格式。
  - Arrival_time: 到达时间，必填，日期时间格式。
  - First_class_seats_remaining: 头等舱剩余座位数，必填，整数。
  - Business_class_seats_remaining: 商务舱剩余座位数，必填，整数。
  - Economy_class_seats_remaining: 经济舱剩余座位数，必填，整数。
  - First_class_price: 头等舱票价，必填，数值格式。
  - Business_class_price: 商务舱票价，必填，数值格式。
  - Economy_class_price: 经济舱票价，必填，数值格式。
  - Status: 航班状态，必填，长度255。
- **Orders 表**
  - ID: 订单编号，自增，主键。
  - Order_time: 订单时间，必填，日期时间格式。
  - Paid: 是否支付，必填，布尔值。
  - User_phone_number: 用户手机号，必填，长度11，外键。
- **Tickets 表**
  - ID: 机票编号，自增，主键。
  - Price: 票价，必填，数值格式。
  - FlightID: 航班编号，必填，长度255，外键。
  - Seat_class: 座位等级，必填，长度255。
  - PassengerID: 乘客身份证号，必填，长度18，外键。

- OrderID: 订单编号，必填，整数，外键。

2.4. 数据流图





2.5. 非功能性需求

业务处理性能

- **响应时间**: 系统必须在用户请求操作后的2秒内做出响应。
- **并发处理能力**: 系统需支持至少1000个并发用户。

安全性

- **数据加密**: 所有传输的隐私数据必须使用密文传输。
- **身份验证**: 系统必须实现多因素身份验证（MFA）。
- **权限控制**: 不同用户角色拥有不同的操作权限，确保数据的访问安全。

完整性

- **数据备份**: 系统需实现每日数据备份，防止数据丢失。

- **数据校验**: 系统在数据录入时需进行校验，确保数据的准确性和一致性。

3. 系统详细设计说明书

3.1. 系统功能概述

系统主要分为两个部分：管理员和用户。

管理员主要维护航班信息。而用户可以订票，在数据库中新增乘客、订单和机票。

3.2. 系统功能模块结构

3.2.1. 前端功能

对于管理员，前端提供：登录、查询航班信息、修改航班信息（包括删除航班、修改余票额、修改票价、修改航班状态）、批量导入新航班功能。

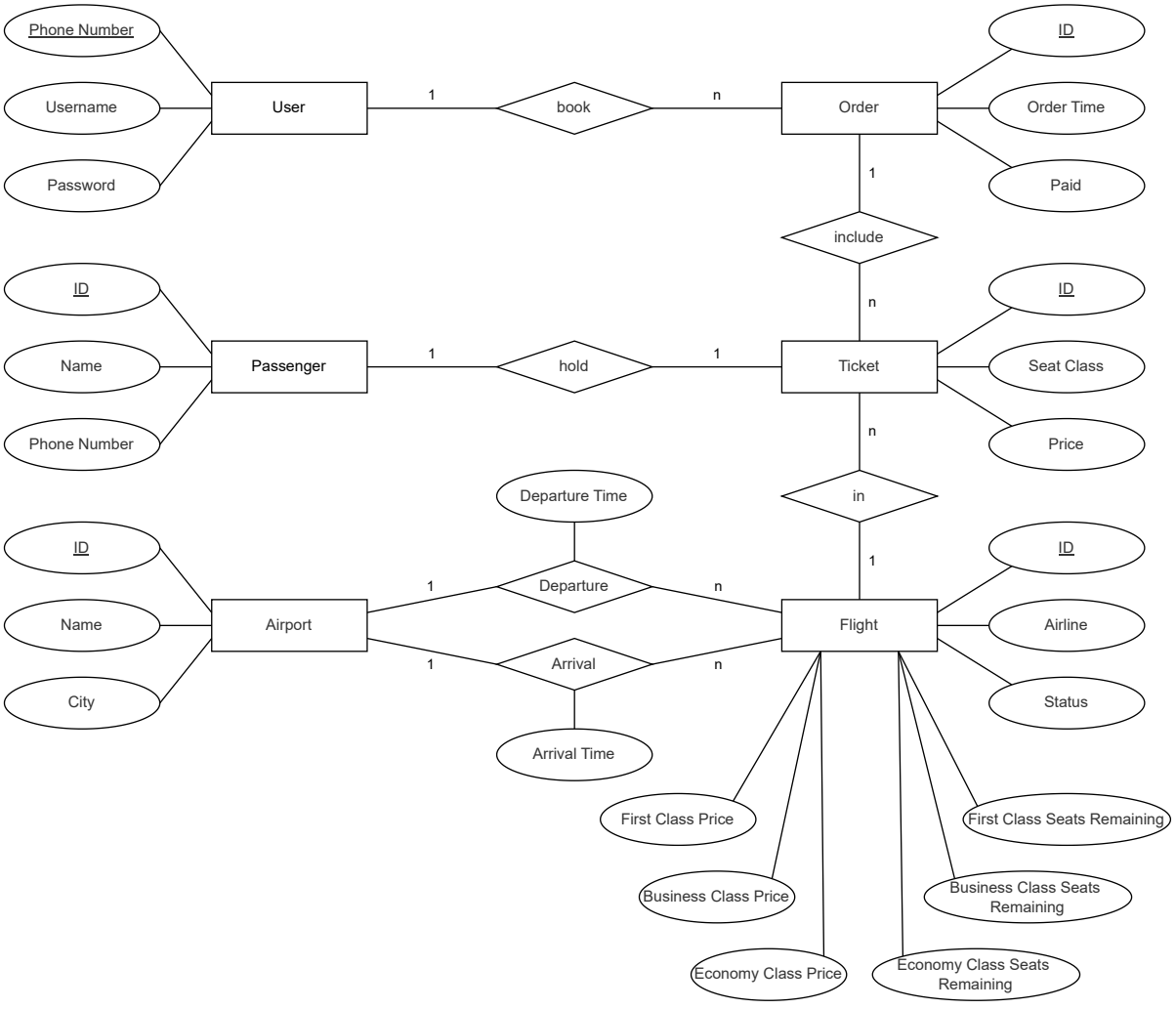
对于用户，前端提供：注册、登录、修改用户信息（包括删除账号、修改手机号、修改密码、修改用户名）、查询航班、下订单、查看订单列表、支付订单、删除订单。

3.2.2. WEB服务端

WEB服务端提供前端功能对应的接口，同时与数据库进行交互。

3.2.3. 数据库端

3.2.3.1. 关系



3.2.3.2. 触发器

设计该触发器，当用户删除订单时，在正式将机票记录删除前，触发将机票对应余座数更新操作。

```
CREATE TRIGGER restore_seats
BEFORE DELETE ON Tickets
FOR EACH ROW
BEGIN
    IF OLD.Seat_class = 'First Class' THEN
        UPDATE Flights
        SET First_class_seats_remaining = First_class_seats_remaining + 1
        WHERE ID = OLD.FlightID;
    ELSEIF OLD.Seat_class = 'Business Class' THEN
        UPDATE Flights
        SET Business_class_seats_remaining = Business_class_seats_remaining + 1
        WHERE ID = OLD.FlightID;
    ELSEIF OLD.Seat_class = 'Economy Class' THEN
        UPDATE Flights
        SET Economy_class_seats_remaining = Economy_class_seats_remaining + 1
        WHERE ID = OLD.FlightID;
    END IF;
END //
```

3.2.3.3. 存储过程

创建该存储过程，将用户下订单后的一系列操作（更新乘客信息、更新余座数、插入机票信息）封装为一个存储过程。

```
CREATE PROCEDURE AddPassengerAndTicket(
    IN p_passenger_id VARCHAR(18),
    IN p_name VARCHAR(255),
    IN p_phone_number BIGINT,
    IN p_seat_class VARCHAR(255),
    IN p_flight_id VARCHAR(255),
    IN p_price DECIMAL(7, 2),
    IN p_order_id INT
)
BEGIN
    -- 插入或更新乘客信息
    INSERT INTO Passengers (ID, Name, Phone_number)
    VALUES (p_passenger_id, p_name, p_phone_number)
    ON DUPLICATE KEY UPDATE Name=VALUES(Name), Phone_number=VALUES(Phone_number);

    -- 更新航班座位数
    IF p_seat_class = 'First Class' THEN
        UPDATE Flights
        SET First_class_seats_remaining = First_class_seats_remaining - 1
        WHERE ID = p_flight_id;
    ELSEIF p_seat_class = 'Business Class' THEN
        UPDATE Flights
        SET Business_class_seats_remaining = Business_class_seats_remaining - 1
        WHERE ID = p_flight_id;
```

```
    ELSEIF p_seat_class = 'Economy Class' THEN
        UPDATE Flights
        SET Economy_class_seats_remaining = Economy_class_seats_remaining - 1
        WHERE ID = p_flight_id;
    END IF;

    -- 插入机票信息
    INSERT INTO Tickets (Price, FlightID, Seat_class, PassengerID, OrderID)
    VALUES (p_price, p_flight_id, p_seat_class, p_passenger_id, p_order_id);
END //
```

3.3. 系统界面设计

3.3.1. 登陆界面设计

该页面设计了一个登录框，并将一个轮播图组建作为背景。

3.3.2. 主页页面设计

主页设计包括一个导航栏，导航栏包含跳转至主页和订单列表页的超链接。

其下面是一个轮播图组建作为背景。

再下面是搜索页面的入口，可以输入航班查询条件跳转到查阅页面。出发日期默认是当前日期的下一天；出发地和目的地默认为北京和上海；乘客人数默认为1。

3.3.3. 查询页面设计

显示查询结果。鼠标悬停在列表行上，列表行会高亮显示。点击列表行，可以跳转至订票页面。

3.3.4. 订票页面设计

3.3.5. 订单页面设计

3.3.6. 订单列表页面设计

3.4. 系统物理模型

表如下：

- Passengers(ID, Name, Phone_number)
- Users(Phone_number, Username, Password)
- Airports(ID, Name, City)
- Flights(ID, Airline, Departure_airport, Arrival_airport, Departure_time, Arrival_time, First_class_seats_remaining, Business_class_seats_remaining, Economy_class_seats_remaining, First_class_price, Business_class_price, Economy_class_price, Status)
- Orders(ID, Order_time, Paid, User_phone_number)
- Tickets(ID, Price, FlightID, Seat_class, PassengerID, OrderID)

其索引为：

- Flights(Departure_airport) REFERENCES Airports(ID)

- Flights(Arrival_airport) REFERENCES Airports(ID)
- Orders(User_phone_number) REFERENCES Users(Phone_number)
- Tickets(FlightID) REFERENCES Flights(ID)
- Tickets(PassengerID) REFERENCES Passengers(ID)
- Tickets(OrderID) REFERENCES Orders(ID)

无视图设计。

3.5. 系统安全体系设计

对于用户管理与控制，采用管理员控制航班信息，用户控制订票信息的设计方式。

对于存储与恢复，采用定时自动将数据库导出备份的方式。

3.6. 系统运行环境设计与部署结构

本系统使用Python Flask的运行环境。

部署结构如下：

```
.
├── Manager
|   ├── func
|   |   ├── config.py       # 配置文件
|   |   ├── index.py        # 主页后端
|   |   └── login.py        # 登录后端
|   ├── main.py
|   ├── static
|   |   ├── css             # 各页面CSS文件
|   |   |   ├── index.css
|   |   |   └── login.css
|   |   └── js              # 各页面JS文件
|   |       ├── index.js
|   |       └── login.js
|   └── templates           # 各页面HTML文件
|       ├── index.html
|       └── login.html
├── Service
|   ├── func
|   |   ├── __init__.py
|   |   ├── book.py         # 订票后端
|   |   ├── cancel_order.py # 取消订单后端
|   |   ├── config.py       # 配置文件
|   |   ├── index.py        # 主页后端
|   |   ├── login.py        # 登录后端
|   |   ├── modify.py       # 修改用户信息后端
|   |   ├── order.py        # 订单详情后端
|   |   ├── order_list.py   # 订单列表后端
|   |   ├── pay_confirm.py  # 支付后端
|   |   ├── search.py       # 查询后端
|   |   ├── signup.py       # 注册后端
|   |   └── utils.py        # 工具函数
|   ├── main.py
|   ├── static
```

```
|   |   ├── css             # 各页面CSS文件
|   |   |   ├── book.css
|   |   |   ├── index.css
|   |   |   ├── login.css
|   |   |   ├── modify.css
|   |   |   ├── order.css
|   |   |   ├── order_list.css
|   |   |   ├── search.css
|   |   |   └── signup.css
|   |   └── js              # 各页面JS文件
|   |       ├── checkInfo.js
|   |       ├── index.js
|   |       ├── login.js
|   |       ├── modify.js
|   |       ├── search.js
|   |       ├── signup.js
|   |       └── slideshow.js
|   └── templates           # 各页面HTML文件
|       ├── book.html
|       ├── index.html
|       ├── login.html
|       ├── modify.html
|       ├── order.html
|       ├── order_list.html
|       ├── search.html
|       └── signup.html
├── data_source             # 数据
    ├── airports
    |   ├── airports.csv    # 机场信息
    |   ├── airports.sql    # 机场信息插入sql
    |   ├── make_data.py    # 将raw数据转换为csv和sql
    |   └── raw.txt         # 机场信息原始文本
    ├── db_user.sql         # 创建用户sql
    ├── flights
    |   ├── add.csv         # 批量新建航班样例
    |   └── flights.sql     # 航班列表初始化sql
    ├── init_manager_db.sql # 新建管理员数据库sql
    └── init_service_db.sql # 新建用户数据库sql
```

4. 用户安装与使用手册

进入项目文件夹后，安装环境依赖：

```
pip install -r requirements.txt
```

然后初始化数据库：

```
mysql> source src/data_source/init_service_db.sql    -- 新建用户数据库
mysql> source src/data_source/init_manager_db.sql    -- 新建管理员数据库
mysql> source src/data_source/db_user.sql            -- 创建用户
mysql> source src/data_source/airports/airports.sql -- 创建机场表
mysql> source src/data_source/flights/flights.sql    -- 创建初始航班表
```

随后进入Manger并启动管理员服务：

```
cd src/Manager && python main.py
```

在另一命令行窗口进入Service并启动用户服务：

```
cd src/Service && python main.py
```

最终，分别在浏览器访问主机 `IP` 的 `8889` 和 `8888` 端口操作管理员和服务端。

5. 所有源代码与脚本

见本项目的 `src` 文件夹。

6. 系统设计总结

通过本次课程设计，我完成了从需求分析、系统设计、数据库设计、功能实现到测试和部署的全过程。这不仅加深了我对数据库系统原理的理解，也提升了我在实际项目中应用这些原理的能力。虽然过程中遇到了一些挑战，但通过不断学习，我最终成功地完成了项目。

本系统的成功设计和实现为我今后在数据库系统和软件开发领域的进一步学习和实践打下了坚实的基础。我也认识到，在实际开发中，系统的可扩展性和维护性是至关重要的，这需要在设计初期就进行充分的考虑和规划。

7. 课程总结与建议

本学期的《数据库系统原理》课程，通过理论学习与实践操作相结合的方式，使我对数据库系统的设计、管理与应用有了深刻的理解和掌握。在课程中，我不仅学习了数据库的基本原理和操作，还通过课程设计项目，实际应用这些知识，构建了一个完整的航班订票系统。以下是我在本课程中的几点总结：

1. **理论与实践并重**：
    - 通过课堂教学，我系统学习了数据库的基本概念、数据库设计理论、SQL语言以及事务管理等核心知识。
    - 课程设计项目则提供了实践平台，让我将理论知识应用于实际开发，从需求分析、系统设计、数据库设计到编码实现与测试，整个过程对知识的理解更加深入。
2. **系统设计能力提升**：
    - 课程设计要求我从零开始设计并实现一个数据库系统，这大大提升了我的系统设计能力。我学会了如何进行需求分析、如何设计数据库表结构、如何进行数据建模等关键技能。
    - 特别是在处理复杂业务逻辑、确保数据一致性与完整性、优化数据库性能等方面，我掌握了许多实用的方法和技巧。
3. **问题解决能力的提升**：
    - 在项目开发过程中，我遇到了许多技术难题，如数据库设计中的各种约束实现、复杂查询的优化、系统的安全性设计等。通过查阅资料，我逐一解决了这些问题，大大提升了我的独立解决问题的能力。

建议：本门课的课设负担较重，希望可以多人合作完成。