航空公司乘客管理系统设计总结

本航空公司乘客管理系统旨在构建一个功能全面、结构清晰且具备一定健壮性的信息管理平台。系统整体采用成熟的B/S（浏览器/服务器）三层架构，将用户交互界面、业务逻辑处理以及数据持久化存储有效分离，从而提高了系统的模块化程度、可维护性和可扩展性。其核心目标是实现对航空公司乘客信息的便捷管理，包括乘客基本信息的增删改查，以及与乘客相关的航班预订等业务功能。

在前端设计方面，系统通过 passengers.html 文件提供了一个交互友好且具有现代科技感的用户界面。该界面利用HTML、CSS（Bootstrap框架）和JavaScript技术构建，实现了乘客信息的动态展示、表单输入与校验、以及与后端API的异步通信。用户可以通过该界面直观地浏览乘客列表，便捷地添加新乘客、编辑现有乘客信息或删除乘客记录，并能即时收到操作反馈。界面的视觉设计注重用户体验，采用了暗色主题和动态效果，提升了操作的流畅性和美观度。

WEB服务端作为系统的核心业务逻辑处理单元，基于Python Flask框架开发（app.py）。它负责接收并处理来自前端的HTTP请求，执行相应的业务规则，并与后端MySQL数据库进行数据交互。服务端提供了一系列RESTful API接口，这些接口既支持直接通过SQL语句操作数据库，也支持通过调用预定义的数据库存储过程来执行更复杂的业务操作。服务端还包含了完善的数据库连接管理和错误处理机制，能够有效地处理数据库操作异常，并以统一的JSON格式向前端返回处理结果或错误信息，保证了前后端数据交互的规范性和可靠性。

数据库端是采用MySQL关系型数据库管理系统。其物理模型设计周全，定义了包括航空公司、机场、乘客、航线、航班、预订、支付、机票以及航班状态等在内的多张核心数据表，并通过主键、外键、唯一约束等保证了数据的完整性和一致性。为了提升数据库操作的效率和封装复杂的业务逻辑，系统大量运用了存储过程，例如用于查询乘客及其预订详情的GetPassengerAndBookings，以及用于添加、删除、修改乘客信息的系列存储过程。同时，系统还巧妙地利用了触发器来自动化处理某些关联业务，例如在预订记录发生变化时自动更新航班的已预订座位数，或在乘客邮箱变更时自动记录审计日志，这些设计都大大增强了系统的自动化和智能化水平。此外，合理的索引策略，如在主键、外键及常用查询字段上建立索引，也为系统的查询性能提供了保障。

针对并发事务处理这一数据库应用中的核心挑战，本系统高度重视。在应用层面，所有数据库写操作均被封装在事务中，通过标准的 commit 和 rollback 机制来确保操作的原子性和数据的一致性。更进一步，系统通过编写专门的Python并发测试脚本，模拟了多用户并发访问场景，并针对不同的MySQL事务隔离级别（如读未提交、读已提交、可重复读），深入验证和分析了可能出现的脏读、不可重复读、丢失修改等典型并发问题。这些模拟实验不

仅验证了数据库自身的并发控制能力，也加深了对并发事务设计的理解，为构建高并发、高可靠系统奠定了理论和实践基础。

在系统安全与运行环境方面，虽然未在应用层面实现复杂的用户权限管理，但通过数据库层面的用户访问控制、前后端的数据校验以及事务管理，为系统提供了一定的安全保障。系统设计考虑了在 macOS 操作系统、MySQL 8.0、Python 3.12 及 Flask 框架下的运行环境。部署结构清晰，易于理解和实施。

综上所述，航空公司乘客管理系统在功能完整性、模块清晰度、数据模型合理性、数据库高级特性应用（存储过程与触发器）以及并发控制理解与验证方面均表现出良好的设计水平。系统不仅实现了核心的乘客信息管理功能，还通过前端的友好交互和后端的高效处理，为用户提供了流畅的操作体验。数据库层面的精心设计和并发控制的深入探究，进一步提升了系统的稳定性和数据处理的可靠性，为后续可能的系统扩展和性能优化打下了坚实的基础。