Analysis and Design

Team Freelance

How we used MVC

The model used in this system is a class called SQL_Database, which was a class extending an interface Database. This class interfaces with a Sqlite database to store flight, customer, and employee information. Methods are implemented within this class to get information from and store information to the database, as needed.

The controllers correspond to each view of the system. For example, there is a CustomerMainViewController to be an interface between the model (SQL_Database) and the customer view.

The views use JavaFX to interface with the user, providing prompts, accepting input, and outputting information as needed. The views interface with the corresponding controllers through methods called upon a button press and input field variables.

Design Patterns Used

A **singleton** pattern was used for the database, customer controller, and employee control ler classes to ensure that all sections of the code refer to the same database connection. This was implemented with a getInstance() function and making the constructor private for the database interface class (SQL_Database).

Additionally, the **factory** design pattern was used in the employee and customer controllers for populating cells in the flight search table.

Object Oriented Design Principals Used

Encapsulation was used by making variables and methods only used within the scope of their class *private*. Additionally, parts of the program that change frequently were encapuslated within a single class. For example, the database, which changes frequently, was encapsulated within the SQL_Database class.

Abstraction was used by making SQL_Database and a text database (used in the early stages of the project) implement the same interface. This way, a Database object could be created and the same methods would work regardless of the type of database used. This abstraction

simplified the process of moving from a text to SQL database since many of the method calls could be used unmodified.

Single Responsibility was used for many of the controller classes, which only implemented a single functionality, such as login. Likewise, single responsibility was also used for many of the view classes.

Class Diagrams