Design Pattern

To complete the Live Venue Matchmaker application I used a combination of the MVC, DAO, and Singleton design patterns to effectively address the business problem.
Model-View-Controller - The MVC pattern was used to separate the application models, views, and business logic. This allowed for the codebase to be neatly organised and ensured that each class had a clear purpose and goal. In addition to the controllers a series of Handler classes were added which are classes that perform business logic, have the potential to be shared between controllers, but are not directly related to a view.
Data Access Object - DAO objects are used in support of models to facilitate CRUD operations with the database. While technically the function of the DAO could be included into respective models, abstracting away this functionality increased compliance with the Single Responsibility Principle as models were pure data structures, and DAOs performed business logic for those structures.
Singleton - The DatabaseHandler class was implemented as a singleton class to reduce overhead. As there only needs to be one instance of the DatabaseHandler and it needs to persist the lifetime of the application code complexity is reduced through being a singleton class.

Adherence to SOLID

Single Responsibility Principle
The application of MVC with Handler and DAO classes ensures that all classes are single responsibility. Each Handler class serves the functions dictated by its name, e.g. FileHandler contains the code for handling files, DatabaseHandler controllers database connections, BookingHandler controls venue/event matching business logic. DAO classes are an intermediary class which pull data from the database and convert the data to a class. Controllers only deal with use interaction from the views and do not perform any other functions.
Open-Closed Principle
The abstract FileHandler class is used to implement the Open-Closed Principle through the getRecordFromCSV and getLineFromCSV functions which when extended on in by the RequestHandler and VenueHandler, allow for different formats of CSV to be read without needing the basic read functions implemented twice.
Liskov Substitution Principle
N/A
Interface Segregation Principle
Interfaces are used for the application's DAO classes in order to create contracts for the concrete DAO classes. To avoid an overlap of interfaces each model DAO implements its own independent interface.
Dependency Inversion Principle
N/A

What I Learnt

This project taught me alot about implementing Java in the MVC format and about using FXML in practice. The biggest take away for me was that I should be identifying the common functions throughout my interfaces and creating a single interface that my classes share. While this would mean that my DAOs would most likely implement 2 interfaces, it would cut down on the total number of files in the codebase, cut down on the total amount of code, and increase compliance with the Open-Closed Principle.