COSC2288 Further Programming

SP1, 2024

Assessment Task (AT) 2:

Assignment 2 Specification V0.9 DRAFT

Live Music Venue Matchmaker Desktop App

✗	Assessment type: **individual assignment; no group work**. Submit online via Canvas → Assignments → Assignment 2. Marks are awarded for meeting requirements as closely as possible according to assignment specifications and the supplied rubric. Clarifications/updates may be made via announcements/relevant discussion forums.
📅	Due date: **Week 13, Sunday, 26th May 2024 at 23:59**. Late submission penalty will be applied unless special consideration has been granted. In a separate assignment submission page you will need to submit a **recorded demo presentation**. You will describe and explain the key concepts adopted in your code, demonstrate code execution, and articulate lessons learnt in the milestones. Instructions for milestone checking will be provided separately in the milestone submissions.
⌄⌄	Weighting: 35 points for Assignment 1 code and documentation project bundle. 5 points for AT2 Milestone 1 video presentation recording Due: 14 April 23:59. 5 points for AT2 Milestone 2 video presentation recording Due: 12 May 23:59. 5 points for AT2 demo and interview. One online 20 minutes meeting with class tutor 27-31 May at an agreed appoinment time.

1. Overview

NOTE: Carefully read this document. In addition, regularly follow the Canvas assignment discussion board for assignment related clarifications and discussion.

In this assignment you are required to work individually to implement an advanced object-oriented Java application program that will run on PC desktop, called **Live Music Venue Matchmaker**. You will practice your deeper knowledge on Java, object-oriented (OO) concepts, Java Collections Framework, exceptions, whilst adding new skills on Graphical User Interface design, file input/output, and database. You will use Java SE 17 or later. You will also incorporate in your program Java libraries including JavaFX 17 and JDBC.

This specification document may be revised later to help clarify requirements, remove ambiguity, improve format, etc.

Background Story

You have been hired by a Melbourne-based venue broker company as an analyst programmer to upgrade an existing system that finds suitable venues to host live music events for a variety number of audience per show.

Existing user at this stage will be the manager, who works with various clients based on job orders. As the company has grown, there will be two additional users which belong to the staff category.
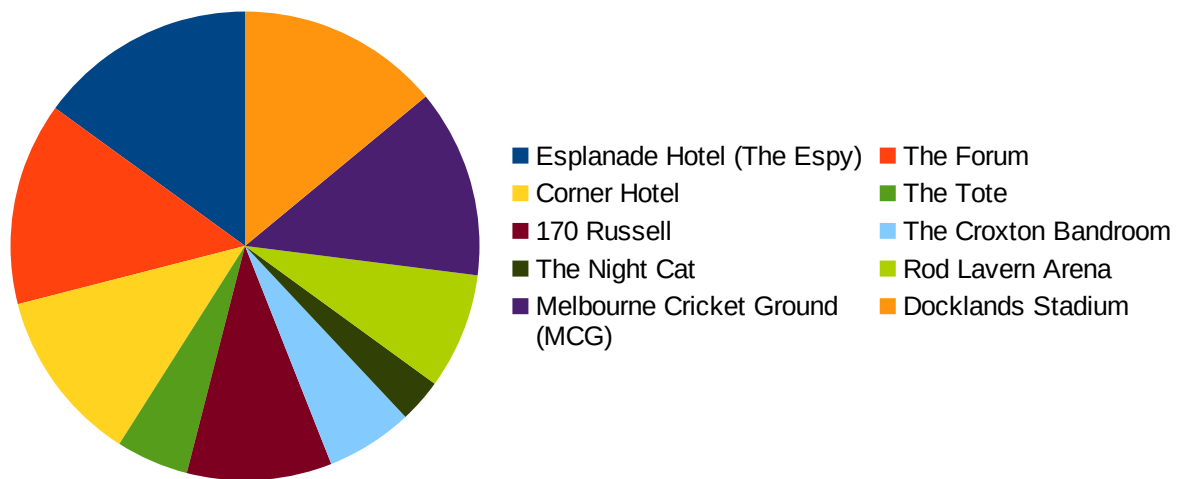

Software Requirements

The system will be used by the staff members to perform the following existing tasks using Graphical User Interface which include visually appealing design and provide the ability to interacat with keyboard, mouse. This GUI version replaces the existing terminal interface.

- List current events requiring a venue based on job requests data.

- Select and show event details including the main artist and supporting acts.

- List venues based on category and name search.

- Select and show a venue details including availability, capacity and hire price. At first, all venues are available for booking.

- Import and extract venue data from csv files.

- Import and extract job request data from csv files.

- Auto-matching feature: Get recommendation for potential venues for all the active job requests. This should be done by comparing attributes between request data and venue data, including availability, capacity, event type (large live concert or gig), and category (indoor or outdoor).

- Hire the venues on behalf of the client.

- Calculate commissions based on brokering and other fees.
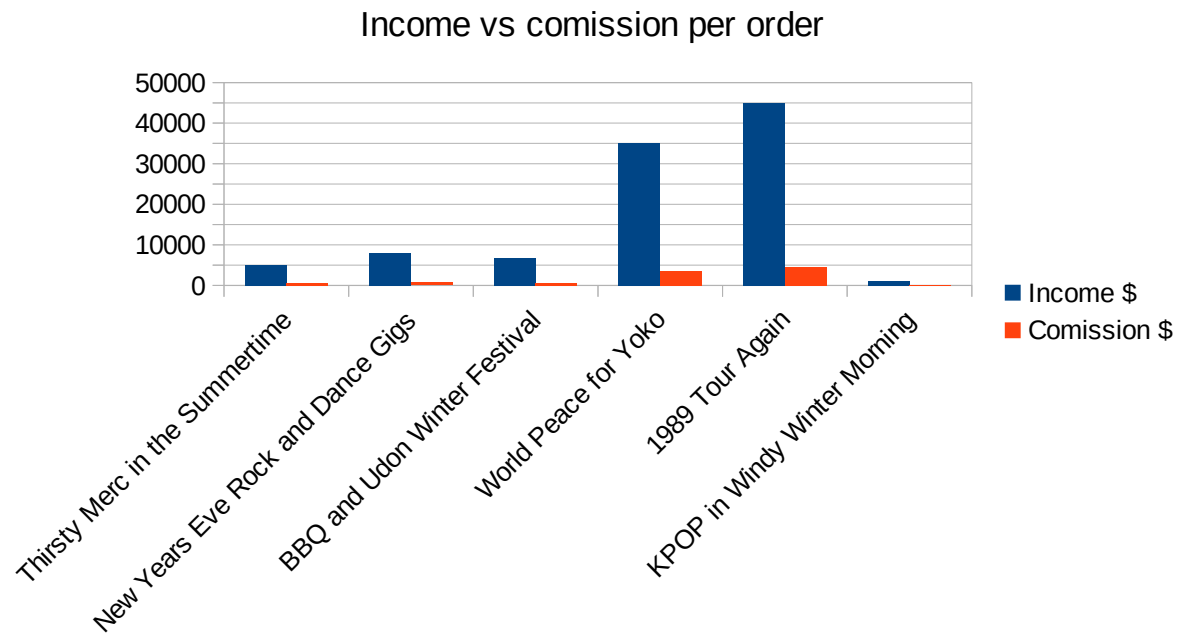
- Show all order details to-date.

The system will be used by the manager who has access to all the above features, plus the following exclusive features accessible by only the manager:

- Add, update, delete staff account.

- Perform core data backup, which includes the ability to:

- Save current list of orders, list of venues, and list of requests in a serilased file format with .lmvm extension.
- Import .lmvm files back into the program which essentialy load orders, venues, and requests data back into the program's data structures.

- Show order summary on a UI screen:
  - Comission per job.
  - Total comission per client.
  - Total comission to date.

- Programmatically generate a visual report that includes 2 key statistics, showing on a UI screen:
  - Pie chart showing venue utilisation shown as per percentage for each venue compared to each other, based on bookings to date. For example:

■ Esplanade Hotel (The Espy) ■ The Forum
■ Corner Hotel ■ The Tote
■ 170 Russell ■ The Croxton Bandroom
■ The Night Cat ■ Rod Lavern Arena
■ Melbourne Cricket Ground (MCG) ■ Docklands Stadium

  - Bar chart that represent income and comission for each order transaction, as per follows:

Income vs comission per order

Bar chart showing Income $ (dark blue) and Comission $ (orange) per order. Y-axis ranges from 0 to 50000 in increments of 10000.

| Order	Income $	Comission $
Thirsty Merc in the Summertime	~5000	~500
New Years Eve Rock and Dance Gigs	~8000	~800
BBQ and Udon Winter Festival	~6500	~700
World Peace for Yoko	~35000	~3500
1989 Tour Again	~45000	~4500
KPOP in Windy Winter Morning	~1500	~100

Software Design

You were to implement a Graphical User Interface program that will run on a full desktop / laptop PC set that harness the convience of a mouse, keyboard, monitor screen, and decent CPU unit.

You were to connect the software program with a database file that resides in local hard drive.

Based on this updated information, now that you have gained a stronger understanding of OOP concepts through the terminal version of the application (Assignment 1), please come up with your own version of UML Class Diagram, Enity Relationship Diagram, and screen mockup design to further extend what you have built in Assignment 1 and re-use classes that are relevant as much as you can.

Following is a sample dashboard screen for the **Live Music Venue Matchmaker**.
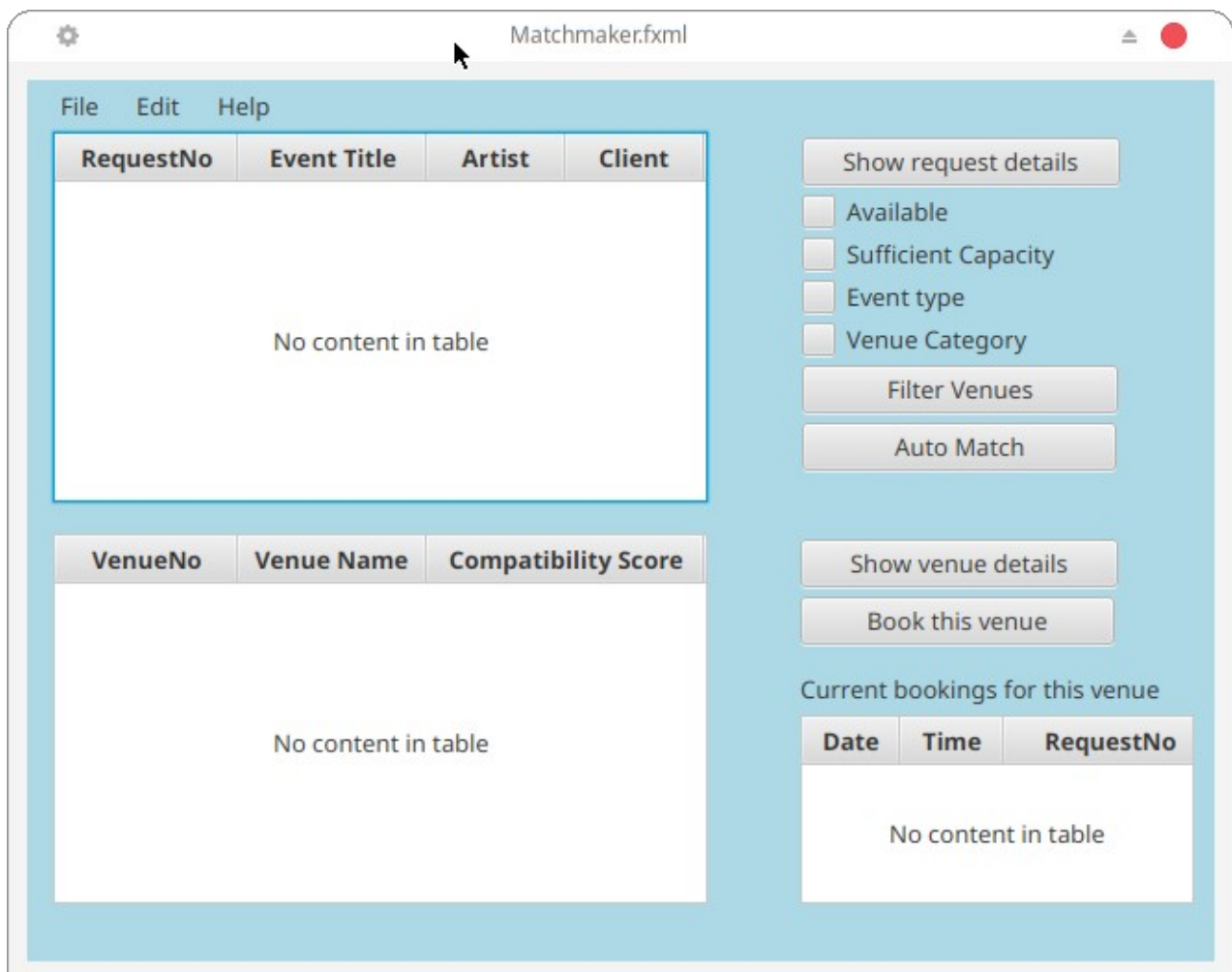
2. Software Specifications

Business Rules

Business rules to incorporate in your program are as follows:

- Brokerage commission is 10% from total venue order.

- When a client made multiple job requests, they are eligible for 1% discount, hence the manager will only receive 9% comission payable by the client.

- When the manager run the auto-match function, the system will scan through a list of venue candidates for each event listed in the job requests based on: capacity, category and suitability keywords and make recommendation for the most suitable one.

- The menu item selections for categories and venues should be coming from a dynamically populated list based on the provided data file. At the end of this assignment, none of these items should be hard-coded, because data files can grow / shrink and frequently getting updated.

- Venue booking can be done manually or through the auto-match feature. For each job request, the system will run a compare and match 4 key criterias against a list of venue data. If all 4 of 4 criteria are matching, the suitability of the venue for that particular event is 100% and therefore a compatibility score of 100. If the app cannot find 100% match, then it will recommend the second most suitable venue for the event, e.g. display those with the compatibility score of 75, 50, or 25. If it cannot find a suitable match, i.e. score of 0% for each compared venues, then it will display the message: "Unable to find a match, please add more venue data or try losen up your criteria".

Tecnical Specification

- Libraries and dependencies:
  - [Java SE Development Kit 17](#)
  - [JavaFX 17](#)
  - [SceneBuilder any version compatible with JavaFX 17](#)
  - [SQLite Studio](#) and SQLite (included in SQLIte Studio)
  - [JDBC driver for SQLite](#)
- The system should now incorporate a database to supply data in the following tables:
  - Users
  - Events
  - Venues
  - Bookings
  - Venue unavailable date and times
  - Clients (Event organisers that produces the requests)
  - Any intermediary table that may be required to represent many-to-many relationships between 2 or more entities.
- Alerts and feedback messages:
  - System Alert 1: The system should prohibit user from making a double-booking, that is making a booking for a venue that is not available for the particular requested date and time.
  - System Alert 2: The system should give warning if one of these criteria cannot be fulfilled and offer the user to still proceed with the booking or cancel the booking and go back to main screen.
  - Other system alerts and feedback messages: The system have adequate notification when there is an error, warning, or invalid data entries, invalid file extension, unreadable file content and display notifications where appropriate.

3. Assessment Criteria

This assessment will determine your ability to implement Object-Oriented Java code according to the specifications shown below. In addition to functional correctness (i.e., getting your code to work) you will also be assessed on code quality. Specifically:

- You are required to demonstrate your understanding of more advanced object-oriented programming principles, including encapsulation, abstraction, design patterns and SOLID principles.

-

- You are required to use well-tested Java collections.
- You are required to validate all user inputs and catch and handle all exceptions.
- You are required to read all system data from files and parse the information correctly.
- You are required to write properly documented code.

- You are required to connect your Java app with SQLite database file via JDBC.
- You are required to implement GUI design using JavaFX 17. No other Java framework like Swing/AWT is allowed and will be considered out-of-scope and uneligible to get any mark.

Documentation requirements
- You are required to write a short design report / presentation slide showing UML class diagrams, screen lo-fi sketches of and mockup, and database design using Entity Relationship Diagram – which will be assessed during milestone 1 check.
- You are required to write a 1-page reflection report at the end, which should include your reflection on:
  - Which design pattern did you adopt?
  - Which specific parts of your program are adhering to SOLID principles? How?
  - What will you do next time if you were given similar tasks?
- Read.me file detailing:
  - how to install required dependencies.
  - how to compile and run the program.
- Your program should be user-friendly, hence user manual is not required.

4. Learning Outcomes

This assessment is relevant to the following Learning Outcomes:

CLO1: Explain the purpose of OO design and apply the following OO concepts in Java code: inheritance, polymorphism, abstract classes, interfaces, and generics.

CLO2: Describe the Java Collections Framework (JCF) and apply this framework in Java code.

CLO3: Describe and Document Diagrammatically the OO design of the JavaFX APIs and apply these APIs to create graphical user interface (GUI) code.

CLO4: Demonstrate proficiency using an integrated development environment such as Eclipse for project management, coding and debugging.

CLO5: Describe and Document Diagrammatically common OO design patterns such as Model View Controller (MVC), Singleton, Facade and apply in Java code.

CLO6: Describe how streams are used for I/O in Java, and apply in code with different types of files (text, binary, random access).

5. Assessment details

You are required to implement the functions mentioned in the task specification in Section 1 - Overview.

- You will incorporate more advanced object-oriented concepts (e.g., abstraction and encapsulation, and design patterns). In particular, you will define several classes to support the OO implementation using Model View Controller, or Model View Presenter.
- All operations must be done via GUI screens that you will design using JavaFX
- GUI FXML design tools such as SceneBuilder is highly recommended but you can design UI components programmatically without it.
- You will also need to choose appropriate data structures to store relevant information. You MUST use Java Collections.
- Your program should perform basic input validation (e.g., out-of-bound option) and provide clear screen messages for invalid inputs and further actions.
- You program have sample data pre-populated in the database.
- You are required to read all information from the given files for venues and demands (HINT: you can use Scanner class and its relevant methods to read from text file and parse the information and save them in relevant database tables). The import CSV function should ignore / reject any duplicate record.
- The order summary should be well-formatted.
- The auto-matching function requires an algorithm to compare the values of relevant attributes of 2 types of objects from relevant collections. Once you worked out the search function, you can take this on as an extra challenge which will earn you more marks. HINT: You may write your own search and compare algorithm, or use existing Java List API methods such as removeAll, retainAll and workout the difference between 2 lists. Please do not use 3rd party library as this would be considered out-of-scope.
- You will need to document the code properly by following Code Conventions for the Java Programming Language by Oracle: https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html
- You are required to catch and handle all exceptions in your program. For example, if you use Scanner class to read the text file, you will need to catch and handle the *FileNotFoundException* specified by Scanner class. Avoid throwing an unspecific Exception; throw a specific exception (e.g., throw a *NumberFormatException* instead of an *IllegalArgumentException*).

6. Submission format

- You must submit a zip file of your project via Canvas, including all documentations.
- Please include a README with instructions on how to run and test your program.
- You can submit your assignment as many times as you would like prior to the due date. The latest submission will be graded.
- You MUST NOT submit compiled files (*.class files). Please check your submission carefully, make sure all java files have been included.

7. Referencing guidelines

You are free to refer to textbooks or notes and discuss the design issues (and associated general solutions) with your fellow students or on Canvas; however, the assignment should be your OWN WORK.

The submitted assignment must be your own work. For more information, please visit the academic integrity website: https://www.rmit.edu.au/students/student-essentials/assessment-and-results/academic-integrity.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students' study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment work even if you have very similar ideas.

Plagiarism-detection tools will be used for all submissions. Penalties may be applied in cases of plagiarism.

8. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e., directly copied), summarized, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,

- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own. RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviors, including:

- Failure to properly document a source

- Copyright material from the internet or databases
- Collusion between students