Coursework on

Software Tools and Techniques (COMP-1618-M01-2023-24)

Submitted By: Md. Sharif Alam

ID: 001262753
Program: MSc in Computing and Information System
Department: Computer and Mathematical Science
Project: London Musicals Ticket System (Java GUI)

Table of Contents

List of Figures

List of Tables

Introduction

This project endeavors to design and develop a Java-based London Musical Ticket System. The objective is to create a user-friendly software application that enables customers to seamlessly purchase musical tickets, choose from different showtimes within a month, and receive printable receipts.

The inspiration for this project stems from the need to enhance the ticket-buying experience for customers interested in London musicals. By harnessing the power of Java programming, the system aims to provide a robust and intuitive platform where users can explore available musicals, view show schedules, and securely book tickets of various types, including Adult, Senior, or Student.

To facilitate the design and development process, the project will progress through distinct stages, starting with a basic understanding of the system's requirements and GUI design. Subsequent stages involve the outline implementation of the GUI, creating a basic working version, testing and validation to ensure input correctness, and finally, saving data externally using PDF files and integration with MySQL database for fetching and storing data.

Design and Development

The London Musical Ticket System was meticulously designed and developed to meet the specified requirements. The development process involved several key stages, each contributing to the overall functionality and user experience of the system.

GUI Design

The initial phase involved designing the graphical user interface (GUI) of the application. Sketches were created to plan the layout, considering real-world examples of ticket machines. The GUI was designed to simulate the input of movie selection, show time, ticket types, and ticket number allowing users to specify the day, time slot, ticket type, after that the seat number, and ticket number will be auto generated.

Prototype Implementation

Following the GUI design, a prototype version of the system was implemented without full functionality. The focus was on achieving the appearance outlined in the design phase. The implementation utilized elements such as spinners, drop-down menus (JComboBox), buttons, and text fields. The goal was to create a visually representative prototype of the final system.

Basic Working Version

The system's basic working version was then implemented, incorporating functionality to support the selection of musicals, show times, ticket types and number of tickets. Users could interact with the system to buy tickets, and the GUI displayed relevant information dynamically. The implemented features allowed users to experience the core functionalities of the ticketing system.

1. Musical List Fetched form MySQL Database. And users can filter musicals by title, category, runtime, and all other columns. The database column is also dynamically populated to the JComboBox.

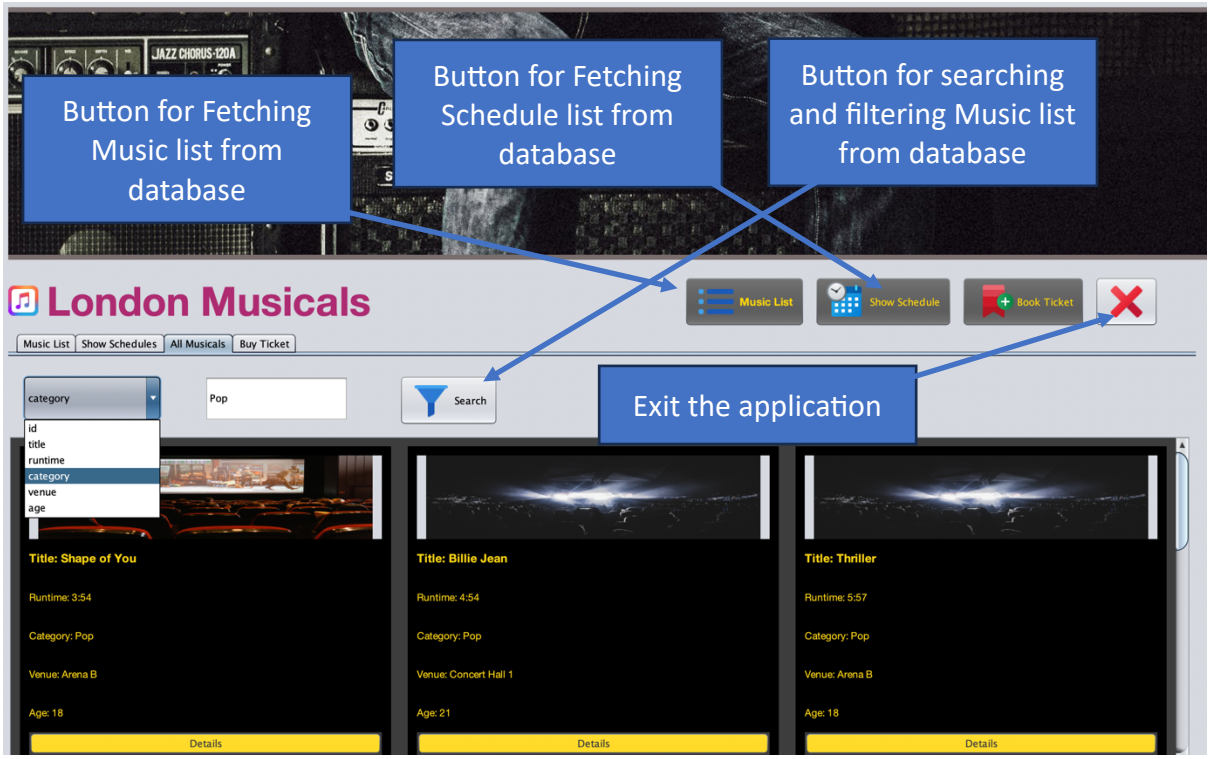
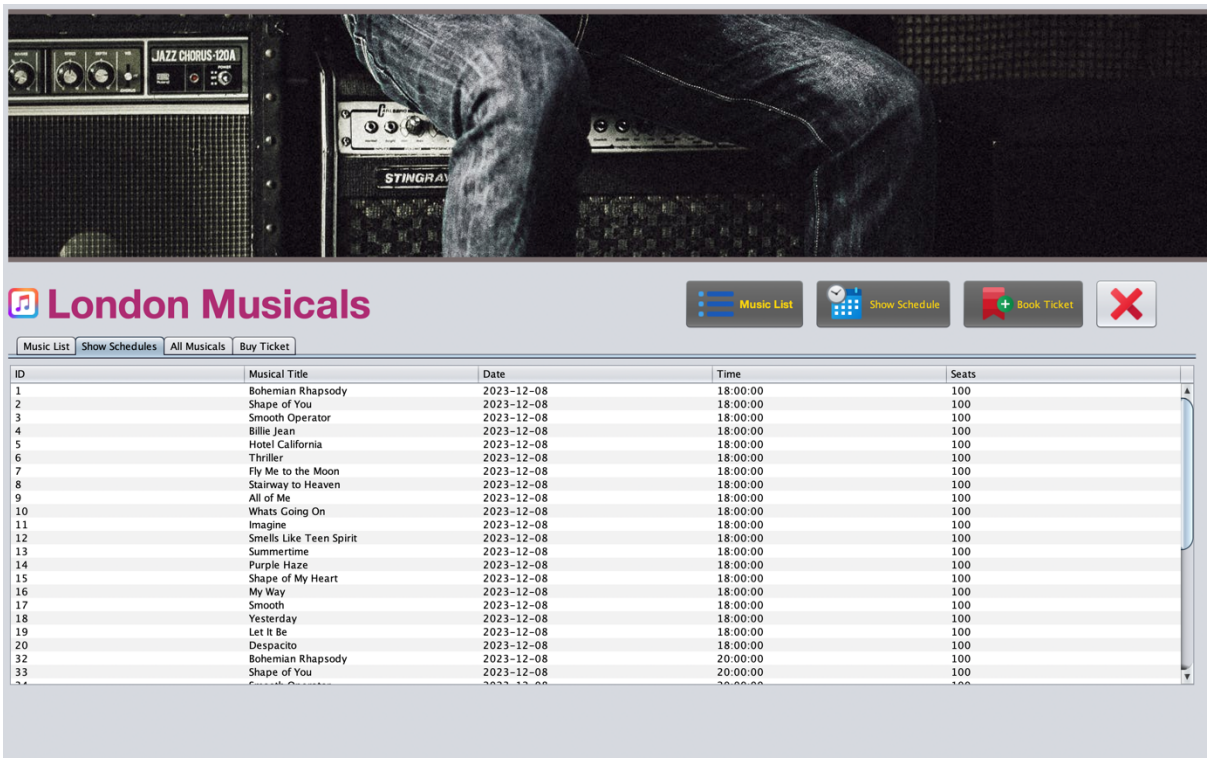


Figure 1: Music List With Filter and Search Feature

2. Schedule List is also fetched from MySQL database schedules table which is related with the music table with foreign Key music_id. Each musical has one of many schedules on the database.



ID	Musical Title	Date	Time	Seats
1	Bohemian Rhapsody	2023-12-08	18:00:00	100
2	Shape of You	2023-12-08	18:00:00	100
3	Smooth Operator	2023-12-08	18:00:00	100
4	Billie Jean	2023-12-08	18:00:00	100
5	Hotel California	2023-12-08	18:00:00	100
6	Thriller	2023-12-08	18:00:00	100
7	Fly Me to the Moon	2023-12-08	18:00:00	100
8	Stairway to Heaven	2023-12-08	18:00:00	100
9	All of Me	2023-12-08	18:00:00	100
10	Whats Going On	2023-12-08	18:00:00	100
11	Imagine	2023-12-08	18:00:00	100
12	Smells Like Teen Spirit	2023-12-08	18:00:00	100
13	Summertime	2023-12-08	18:00:00	100
14	Purple Haze	2023-12-08	18:00:00	100
15	Shape of My Heart	2023-12-08	18:00:00	100
16	My Way	2023-12-08	18:00:00	100
17	Smooth	2023-12-08	18:00:00	100
18	Yesterday	2023-12-08	18:00:00	100
19	Let It Be	2023-12-08	18:00:00	100
20	Despacito	2023-12-08	18:00:00	100
32	Bohemian Rhapsody	2023-12-08	20:00:00	100
33	Shape of You	2023-12-08	20:00:00	100

Figure 2:Schedule List for the Musicals

3. In the Booking Page, List of all musicals are dynamically fetched in a drop-down or JComboBox for the users to pick single musical. When user selects a musical then another drop-down for selecting the schedule get populated dynamically based of the selected musical.

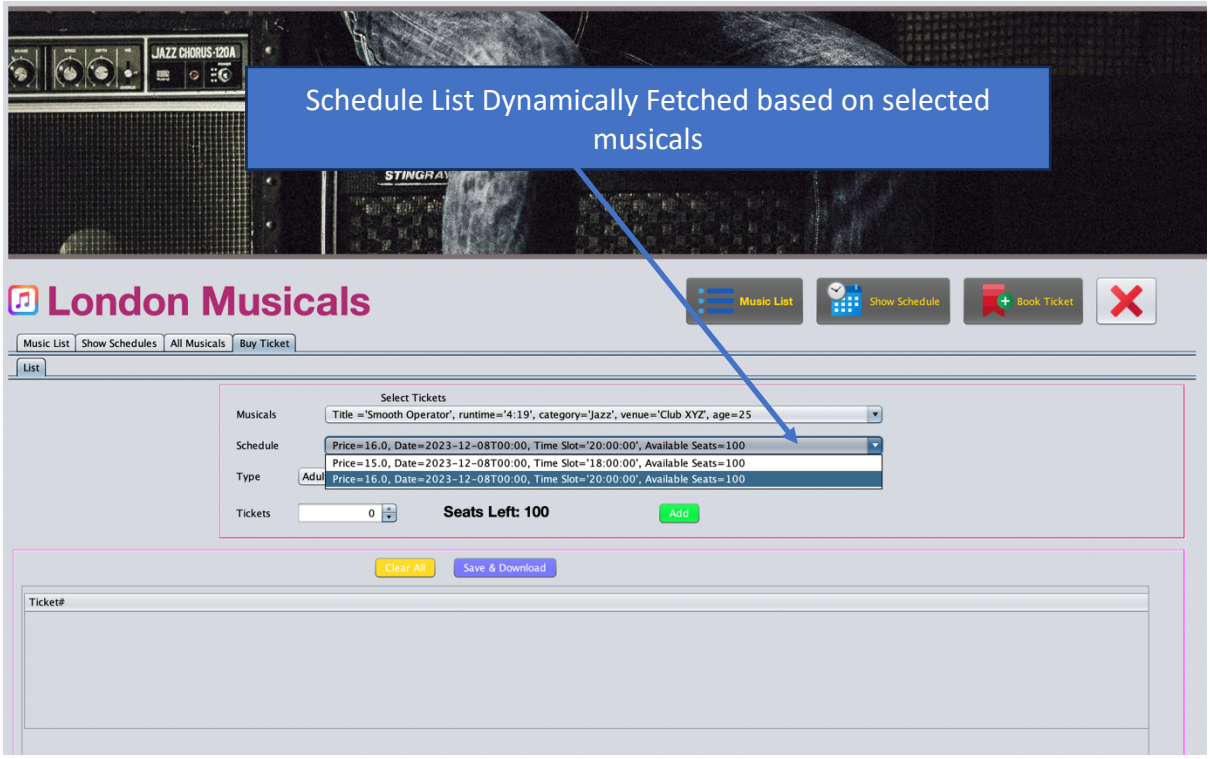


Figure 3:Fetching Schedules based on Selected Musicals

4. Users can add one or multiple tickets of multiple time at a time. All the tickets will the listed in the table below for saving it on the database and downloading the tickets PDF. The ticket number will be unique for every single ticket and the seat number will also be different.

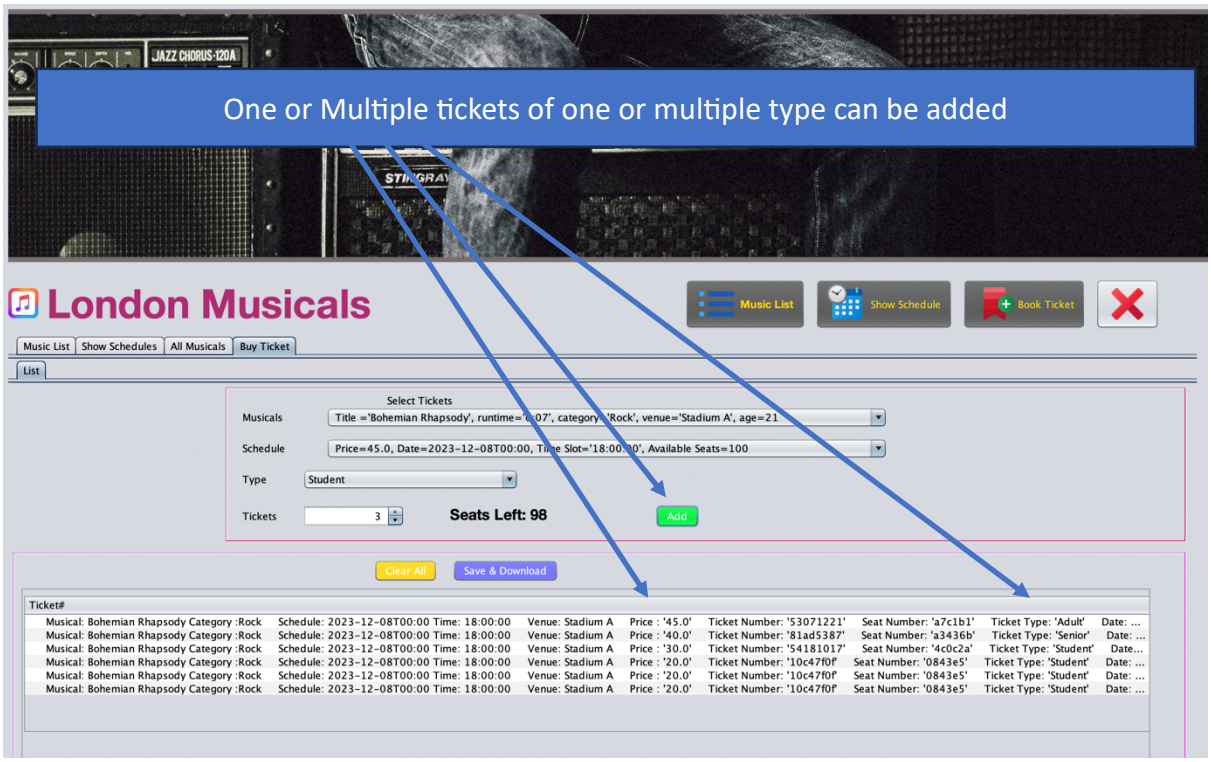


Figure 4: Users booking multiple tickets of multiple types.

5. The number of tickets gets saved on the database with ticket numbers, schedule info and music info. User can download PDF ticket from the system.

Ticket#
Musical: Bohemian Rhapsody Category :Rock Schedule: 2023-12-08T00:00 Time: 18:00:00 Venue: Stadium A Price : '45.0' Ticket Number: '53071221' Seat Number: 'a7c1b1' Ticket Type: 'Adult' Date: 2023-12-14T07:28:17.571286
Musical: Bohemian Rhapsody Category :Rock Schedule: 2023-12-08T00:00 Time: 18:00:00 Venue: Stadium A Price : '40.0' Ticket Number: '81ad5387' Seat Number: 'a3436b' Ticket Type: 'Senior' Date: 2023-12-14T07:28:27.874435
Musical: Bohemian Rhapsody Category :Rock Schedule: 2023-12-08T00:00 Time: 18:00:00 Venue: Stadium A Price : '30.0' Ticket Number: '54181017' Seat Number: '4c0c2a' Ticket Type: 'Student' Date: 2023-12-14T07:28:32.456262
Musical: Bohemian Rhapsody Category :Rock Schedule: 2023-12-08T00:00 Time: 18:00:00 Venue: Stadium A Price : '20.0' Ticket Number: '10c47f0f' Seat Number: '0843e5' Ticket Type: 'Student' Date: 2023-12-14T07:28:38.891250
Musical: Bohemian Rhapsody Category :Rock Schedule: 2023-12-08T00:00 Time: 18:00:00 Venue: Stadium A Price : '20.0' Ticket Number: '10c47f0f' Seat Number: '0843e5' Ticket Type: 'Student' Date: 2023-12-14T07:28:38.891250
Musical: Bohemian Rhapsody Category :Rock Schedule: 2023-12-08T00:00 Time: 18:00:00 Venue: Stadium A Price : '20.0' Ticket Number: '10c47f0f' Seat Number: '0843e5' Ticket Type: 'Student' Date: 2023-12-14T07:28:38.891250

Figure 5:Ticket Download PDF

Testing and Validation

To ensure the robustness of the code, extensive testing and validation processes were implemented. Input validation checks were added to handle bad input, such as non-numeric values or invalid entries. White box testing was conducted, and a test table was used to document various test cases and their outcomes. The testing phase aimed to identify and rectify any faults in the system.

1. While selecting number of tickets, the system is verifying the inputs with various validation along with checking available seats left. The feedback are very clear with proper massage to guide the users to correct their actions.

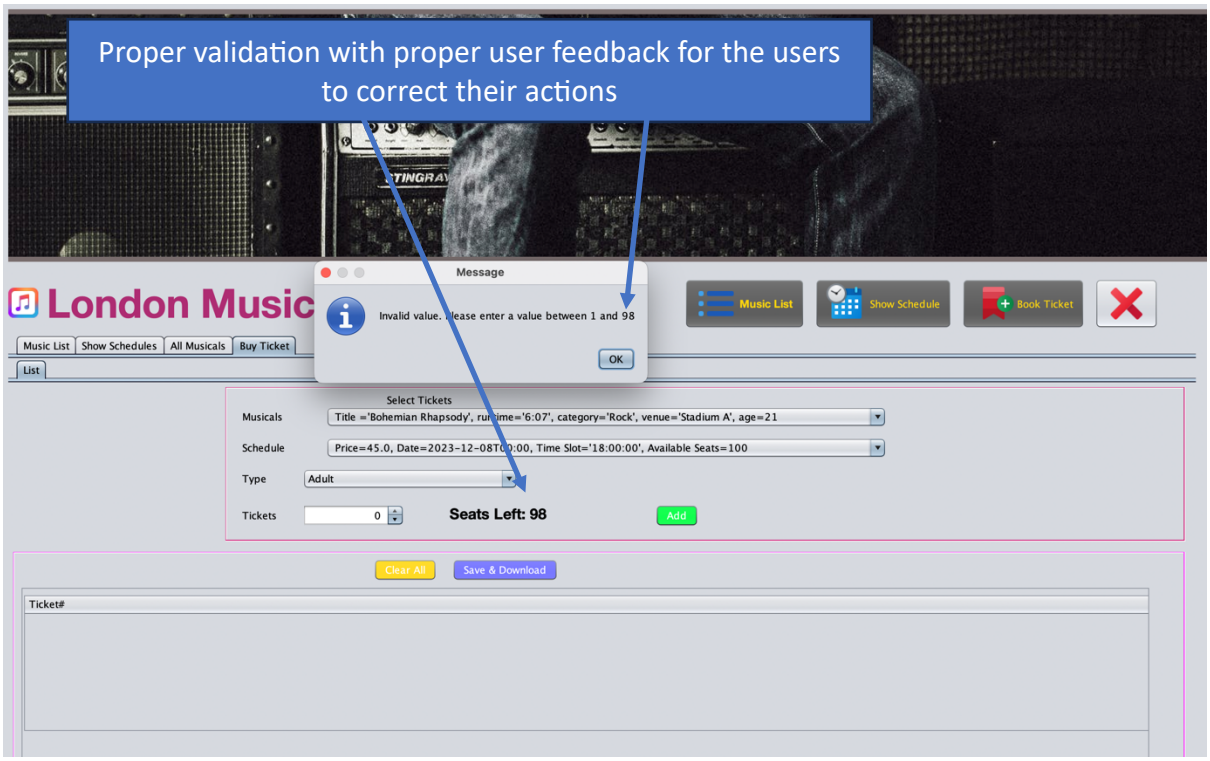


Figure 6:Validation with Proper Feedback

External Data Handling

The system was extended to save and load data externally using PDF and SQL files from the database. A method was introduced to manage the loading of musical data and the creation of text receipts for each transaction to the tickets table of the MySQL database. This allowed for the persistence of transaction details, including the number of tickets, total price, musical name, date/time, ticket type, and seat number.

Innovations

In the final stages, the system was further enhanced with innovative features. Users gained the ability to list and search for musicals by name or apply filters. Additionally, a database, specifically MySQL , was integrated to store and manage details. The GUI received enhancements through the incorporation of images.

The design and development process aimed to strike a balance between functionality, usability, and innovation, resulting in a comprehensive London Musical Ticket System. The following screenshots provide glimpses of the system in operation:

Testing and Faults

In this section, I provide an overview of the white box testing conducted on the London Musical Ticket System, along with a discussion of identified faults and failures. The detailed test table and results are included in Appendix B for reference.

White Box Testing Summary

White box testing, also known as structural or glass-box testing, was employed to scrutinize the internal logic and code structure of the London Musical Ticket System. This comprehensive testing approach aimed to ensure the reliability and robustness of the software.

Test Table: (Refer to Appendix B for the detailed test table and results.)

The test table outlines various test cases, covering different aspects of the system, including GUI functionality, data validation, and error handling. Each test case was meticulously designed to assess specific components and functionalities, ensuring a thorough examination.

Discussion of Faults and Failures:

1. **Data Validation Issues:**
    - Fault: During testing, certain scenarios revealed shortcomings in data validation, allowing users to input invalid values while using the spinner and selecting the number of tickets.
    - Resolution*:* Immediate corrective measures were taken to enhance data validation mechanisms and prevent the entry of incorrect or out-of-range values. Users will the prompt with warning along with the hints to correct their inputs. Such as, users can not select number of tickets more the available seats.
2. **GUI Responsiveness:**
    - Fault*:* In some instances, the GUI exhibited sluggish responsiveness, impacting the user experience.
    - Resolution*:* Code optimizations were implemented to enhance the overall responsiveness and ensure smoother navigation within the application.
3. **Incomplete Error Handling:**
    - Fault*:* A few error scenarios were not adequately addressed, leading to incomplete error handling.

- Resolution: Additional error-handling and expectation handling mechanisms were integrated to address previously overlooked scenarios, ensuring comprehensive coverage.

4. **Unresolved Issues:**
    - Several issues, particularly the screen and layout adjustment while resizing the screens and tabs. These issues necessitate further investigation and debugging to pinpoint the root cause and implement effective solutions.

Conclusion

White box testing has been instrumental in uncovering both glaring and subtle issues within the London Musical Ticket System. While many faults have been successfully addressed, persistent challenges highlight the need for continued testing, debugging, and refinement. Ongoing efforts are focused on resolving unresolved issues to enhance the system's stability and deliver a seamless user experience.

Conclusions

Summary of the Program

The London Musical Ticket System has been successfully designed and developed to provide users with an intuitive and functional platform for purchasing musical tickets. The program incorporates a well-designed graphical user interface (GUI), robust functionality, and innovative features such as external data handling and database integration. Through various stages, from initial design to implementation, testing, and enhancements, the program has evolved into a comprehensive solution for managing musical ticket transactions.

Reflection

Undertaking the development of the London Musical Ticket System has been a valuable learning experience. This project has significantly contributed to my knowledge and skills in Java programming, GUI design, data handling, and software testing. Reflecting on the journey, two key aspects stand out:

b) Development of Knowledge and Skills

Throughout this course component, my proficiency in Java programming has undergone significant improvement. The hands-on experience gained in designing and implementing Graphical User Interfaces (GUIs), managing external data, and integrating databases has been instrumental in enhancing my skill set. The development of the London Musical Ticket System has specifically contributed to the refinement of my problem-solving abilities, especially when dealing with challenges related to data validation and system testing.

Some of the key skills and knowledge areas I have developed include:

1. **Object-Oriented Programming Concepts:** The project provided ample opportunities to implement object-oriented programming (OOP) concepts effectively, showcasing their efficiency in software development.
2. **Data Structures:** Working on the project involved practical utilization of various data structures, including arrays, ArrayLists, Lists, and maps. This hands-on experience has deepened my understanding of data organization and manipulation.

3. **Solution Design:** Transitioning from requirements to designing a comprehensive solution with a well-defined workflow and GUI design was a crucial aspect of the learning process.
4. **Database Design:** Utilizing MySQL for database design and management, including structuring and organizing data effectively.
5. **Problem Solving and Error Handling:** Addressing challenges, implementing effective problem-solving strategies, and incorporating robust error handling and validation mechanisms.
6. **Software Testing:** The course emphasized the significance of thorough software testing in ensuring the development of robust, error-free applications. This includes designing and implementing test cases to validate the functionality of the London Musical Ticket System.

Long-Term Impact:

The skills acquired throughout this course component extend beyond immediate application, with broader implications for my future pursuits. The ability to conceive, design, and implement user-friendly applications has become a transferable skill applicable across diverse domains. The practical experience gained in managing databases, handling data, and interacting with external files is particularly valuable in scenarios where persistent data storage and retrieval play a pivotal role.

This course has provided me with a robust foundation in software development, paving the way for engagement in more intricate and advanced projects in the future. The amalgamation of theoretical knowledge and practical application has equipped me with the tools necessary for sustained success and growth in the dynamic field of software development.

Three More Months of Work

Given an additional three months to work on the program, several areas could be further refined and expanded. Potential areas of focus include:

1. **User Experience (UX) Enhancement:** Invest time in refining the GUI for an even more intuitive and visually appealing user experience. Incorporate user feedback to make navigational elements more user-friendly.

2. **Security Measures:** Implement additional security measures, especially when handling user data and financial transactions. This could involve encryption techniques and secure coding practices.
3. **Performance Optimization:** Optimize the program's performance, especially concerning database interactions and data retrieval. Identify and address any bottlenecks that could impact scalability.
4. **Advanced Features:** Explore the integration of advanced features, such as real-time seat availability updates, dynamic pricing based on demand, and integration with online payment gateways.
5. **Cross-Platform Compatibility:** Ensure the application's compatibility across different platforms and devices. This may involve refining the codebase for better adaptability.

In conclusion, the London Musical Ticket System project has been an enriching experience that has significantly contributed to my skills as a software developer. The lessons learned and the hands-on experience gained will undoubtedly prove valuable in my future endeavors within the field of software development.

Appendix A

Database Design and Entity Relationship diagram

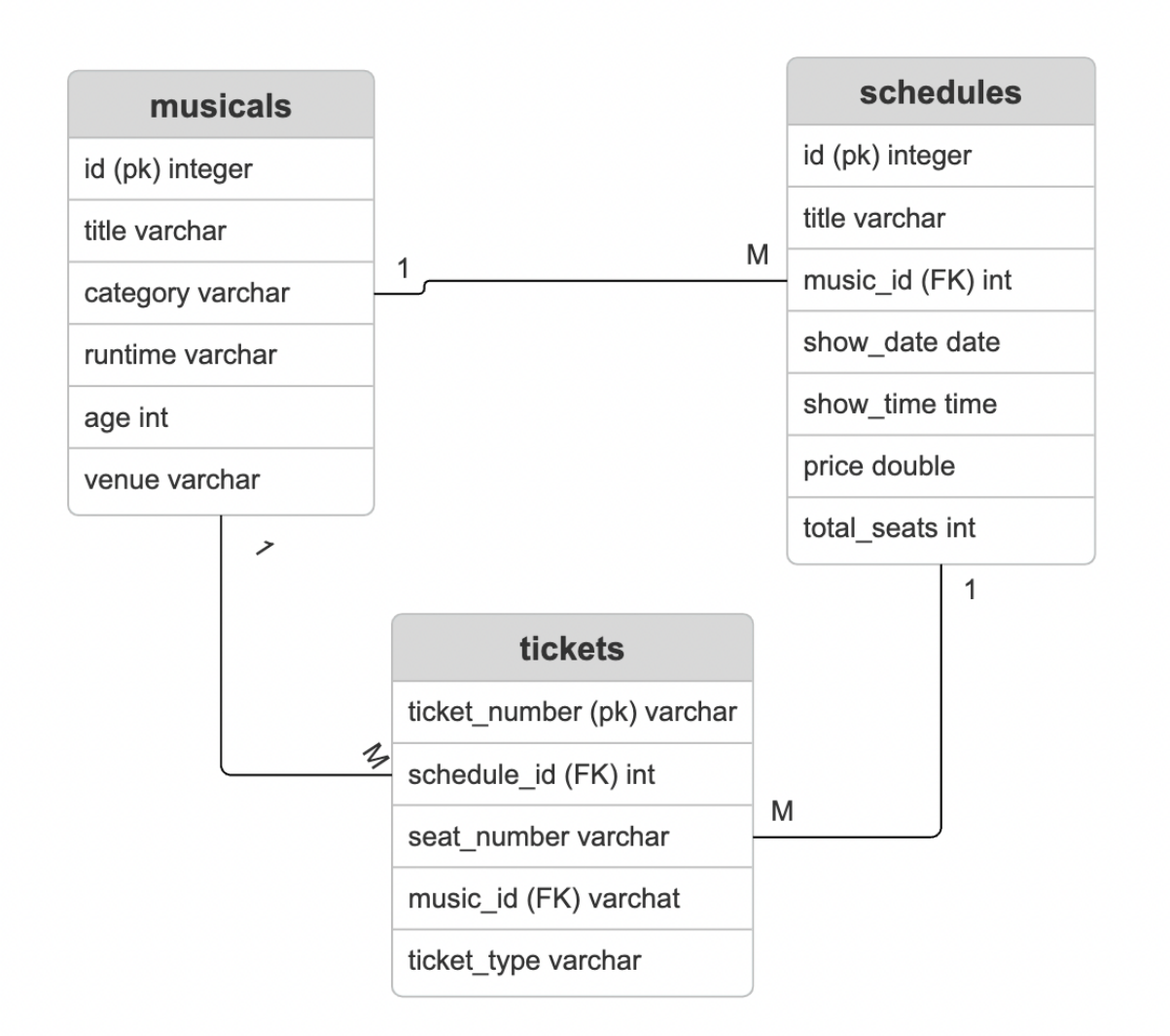


Figure 7: Entity Relationship Diagram

Ticket data structure with unique ticket number and seat number stored on database.

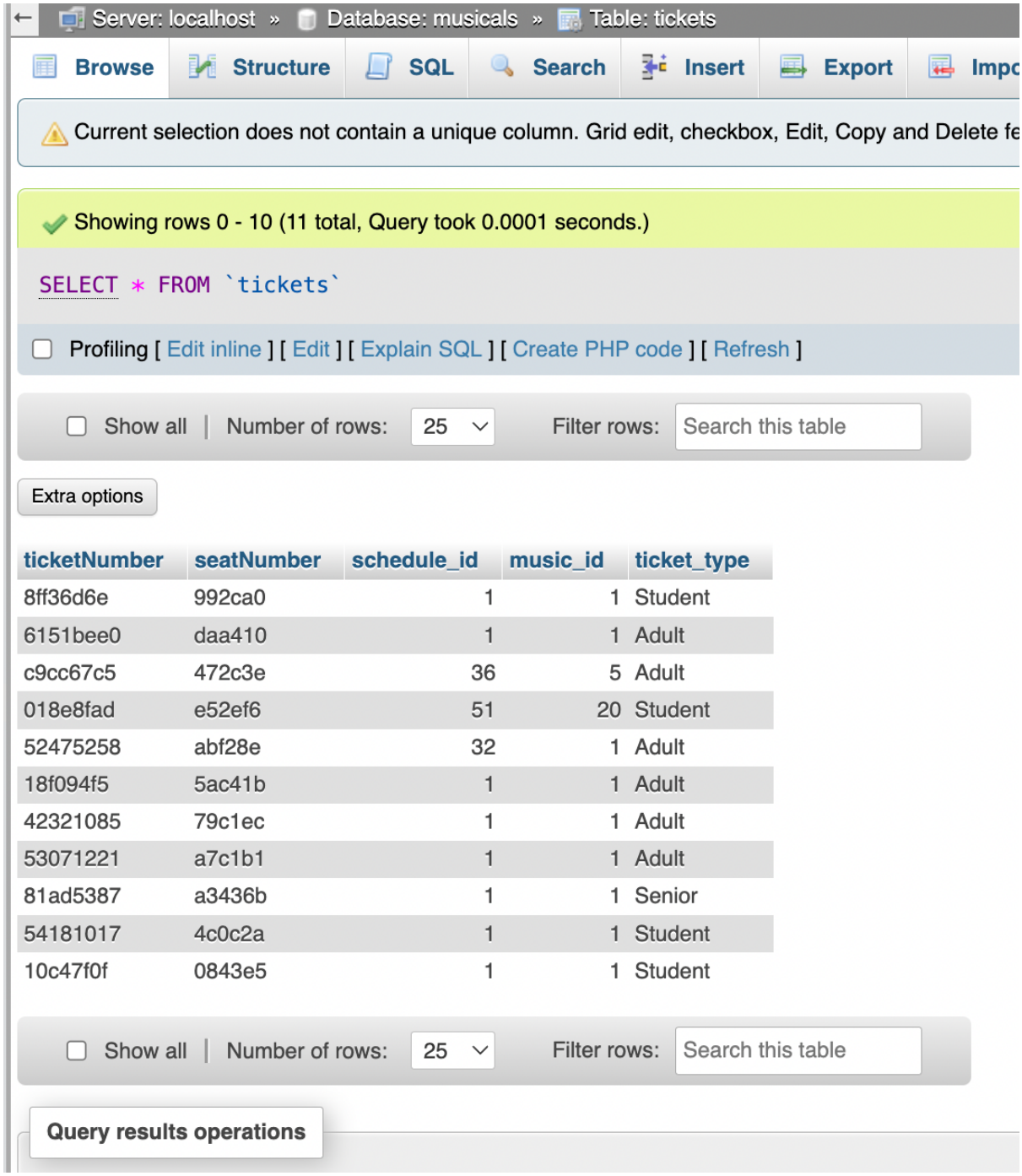


Figure 8: Storing transactions data for tickets

Appendix B

Test Table:

Test Case	Method Tested	Scenario	Input	Expected Output	Result
1	testSetSpinnerValue	Seats selected greater than seats left	Spinner Model: [11,0,11,1], Seats Left: 5	0 (Reset)	Passed
2	testSetSpinnerValue	Seats not selected (0)	Spinner Model: [0,0,8,1], Seats Left: 8	0 (Reset)	Passed
3	testSetSpinnerValue	Seats selected less than of equal to 0	Spinner Model: [-1,-1,8,1], Seats Left: 8	0 (Reset)	Passed
4	testSetSpinnerValue	Seats Selected within the valid range	Spinner Model: [5,9,8,1], Seats Left: 5	5	Passed
5	testSetSpinnerValue	Seats selected greater than seats left	Spinner Model: [11,0,11,1], Seats Left: 5	0	Failed
6	testCalculateTicketPrice	Calculate Adult ticket price	Schedule: [1, 1, LocalDateTime, "18:00", 100, 50.0], Ticket Type: "Adult"	50	Passed
7	testCalculateTicketPrice	Calculate Senior Ticket Price	Schedule: [1, 1, LocalDateTime, "18:00", 100, 50.0], Ticket Type: "Senior"	45	Passed
8	testCalculateTicketPrice	Calculate Student Ticket Price	Schedule: [1, 1, LocalDateTime, "18:00", 100, 50.0], Ticket Type: "Student"	40	Passed

Table 1: Test Table 1

Here's a test table for the **validateSpinnerValue** method:

Test Case	Input (ticketSpinner)	seatsLeft	Expected Result	Remarks
1	5	10	No error message, spinner value remains 5	Valid input within the range
2	15	10	Error message displayed, spinner value reset to 0	Entered value greater than seatsLeft
3	-5	10	Error message displayed, spinner value reset to 0	Entered value less than or equal to 0
4	“abc”	10	Error message displayed, spinner value reset to 0	Non-integer input
5	0	10	Error message displayed, spinner value reset to 0	Entered value less than or equal to 0
6	8	10	No error message, spinner value remains 8	Valid input within the range
7	20	0	Error message displayed,	seatsLeft is 0, any input is invalid

			spinner value reset to 0	
8	5	5	No error message, spinner value remains 5	Valid input within the range

Table 2: Test Table for Validation

Explanation:

- Test Case 1 checks a valid input within the specified range.
- Test Case 2 checks an input greater than the available seats.
- Test Case 3 checks an input less than or equal to 0.
- Test Case 4 checks non-integer input.
- Test Case 5 checks input equal to 0.
- Test Case 6 checks another valid input within the specified range.
- Test Case 7 checks when seatsLeft is 0, any input is invalid.
- Test Case 8 checks a valid input within the specified range.