# London Musical Ticket System

[London Musical Ticket System 1](#_Toc11070)

[Introduction 2](#_Toc14776)

[Design and Development 2](#_Toc26584)

[Testing and Faults 4](#_Toc22975)

[Conclusion 5](#_Toc13413)

[Further Development 6](#_Toc25557)

[Reflection 6](#_Toc26123)

[Appendix 7](#_Toc461)

[Appendix B 7](#_Toc2957)

## Introduction

The purpose of this project is to develop a Musical Ticket System using the knowledge and understanding of the Java programming language that has been developed along with this course.

The London Musical Ticket System is a Java Swing-based GUI project and it includes a number of GUI components such as JLabel, JTextField, JButton, JComboBox and JTable etc to name a few. In addition to these native GUI components, the GUI also incorporates capabilities to load external files (.CSV file for this project) using JFileChooser component and also writes .txt files. Additionally, the project also incorporates a working model of JavaDB (Derby) integration to save data into database table and also to fetch data from the tables.

The general flow or idea for the London Musical Ticket System is to allow users to easily make reservations for their favourite shows. These shows has their own day of month, time slots, run-time information, age-limit etc. After picking their favourite shows with other information such as date, time slot etc, the users also gets a receipts (in form of .txt) to confirm their bookings and also to have a proof about their reservations.

The London Musical Ticket System was developed by keeping in mind the ticket systems employed worldwide and many real ticketing systems were taken into consideration to design this ticket system. The GUI of London Musical Ticket System is very user-friendly. The GUI was focused around an idea which was to prevent user from typing more and more information. In London Musical Ticket System, only a handful textfields are added so that user will not have a burden to write more and more information to make their reservations. Instead a number of information is added in form of dropdowns (JCombobox) which gets filled automatically from the information provided to the system on run time.

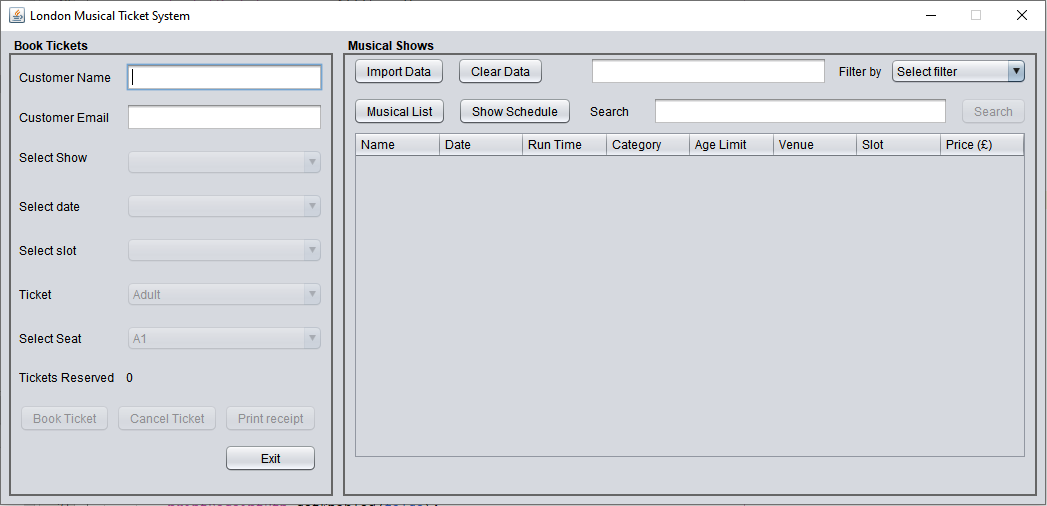
## Design and Development

The London Musical Ticket System was designed solely using Java programming language. The software stack includes NetBeans IDE (version 21) and JDK-21 is used for the Java development. For database, the GUI is integrated with Derby (version 10.17) which is a network-based JavaDB designed specifically for Java applications.

The GUI was designed using NetBeans IDE’s GUI components. The components are carefully placed keeping in mind the idea of a user-friendly environment and also about the real ticket systems employed worldwide. For (almost) every of the GUI component, a callback is also added so that relevant business logic can be handled when a GUI component triggers an action.

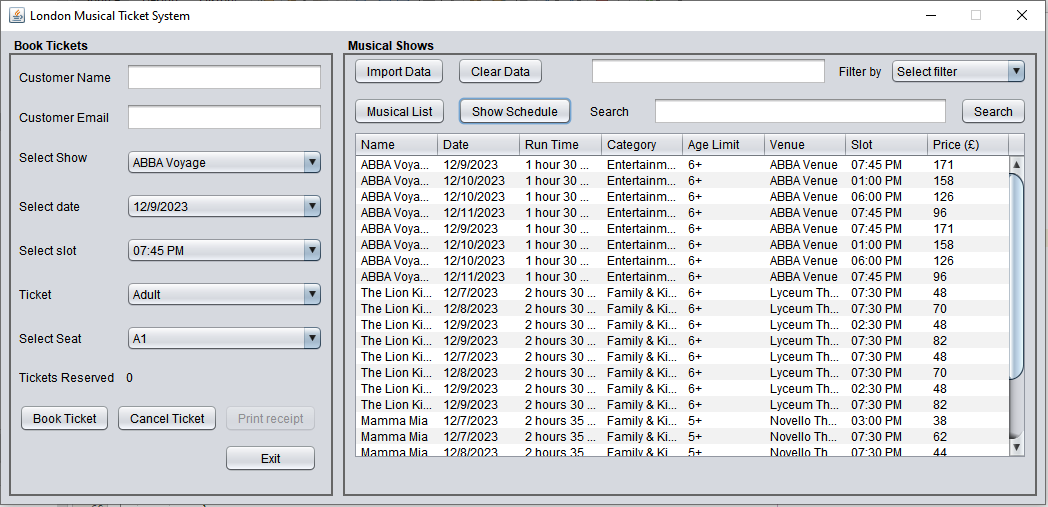
When application starts, a blank GUI is displayed to the user with a number of buttons. The buttons are self explanatory and user should not find any difficulty in understanding the reason for that specific button. In addition to buttons, a number of other GUI components such as Combobox, Table, TextFields and labels are also added in the GUI. On first run, since the GUI is blank and does not contains any information related to London Musical shows, a number of dropdows (comboboxes) and buttons are also disabled. The reason behind it is to prevent incorrect data transactions since there is no viable data in the GUI at the moment.

Below screenshot represents how the actual GUI looks like on first run:



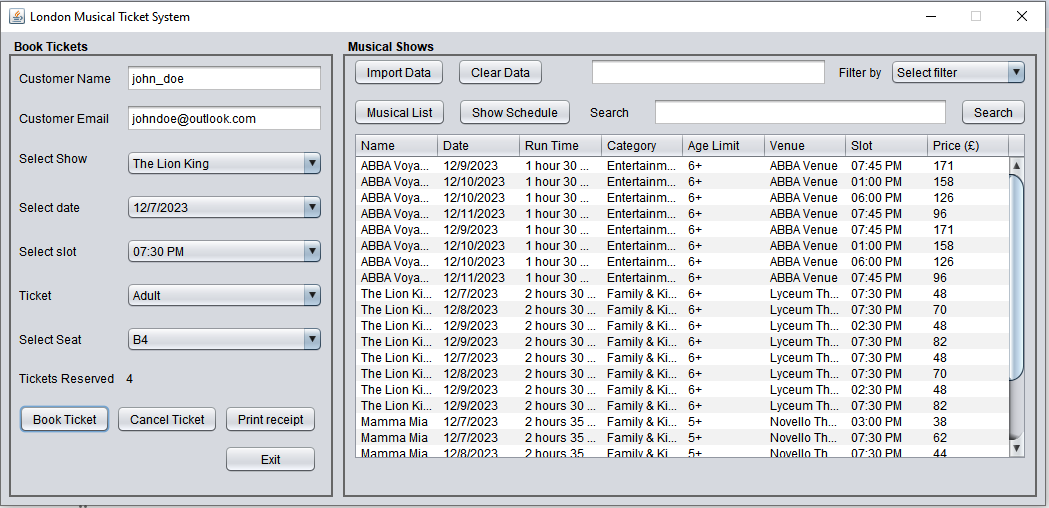
As mentioned above and as it can also be seen from the above screenshot, the GUI is blank and number of GUI components are disabled.

Now user can either import data (using Import Data button) or can see existing data (using either Musical List or Show Schedules buttons). When GUI is filled with relevant data of London Musical shows, the GUI looks a little different as shown in below screenshot.



The above screenshot represents that as soon as the London Musical Ticket System is provided with relevant data, every GUI component becomes enable except the Print receipt button. The reason behind Print Receipt button being disabled at the moment is because there is no ticket reservations made till now.

After making couple of tickets reservations, the GUI is updated and it looks like in below screenshot:



In the above screenshot, it is being displayed that a user **John Doe** has reserved four tickets. It can be seen that Print Receipt buttons is now enabled and user can now print the receipt to get the bill and total amount of his/her reservations along with reservation details.

When user click on Print Receipt button, the Musical Ticket System writes a text file (having .txt extension) containing all details about the reservations the user just made. Below is the example receipt file (in .txt format) of the reservations that we just made earlier.



The user can simply print this reservation file for record. It can be observed that the London Musical Ticket System is designed by taking in interest about the real world scenarios that it can be seen that receipt filename pattern is not a simple pattern. The pattern contains, along with customer name, also a timestamp. The reason behind adding timestamp is that it can kept in system for records to avoid any fraudulent activity. Not only to avoid frauds, it also allows same customer to make multiple reservations and all of the reservations will be kept in different receipts (as text files) and will not override previous receipt files.

## Testing and Faults

Below is the table which summarizes the white box testing of the London Musical Ticket System. The table includes many different aspects and lists a number of tests which were tested. These scenarios ranges from static GUI changes to tests including input fields validations.

The code is organized and developed in such a way that it notifies user about error or unwanted conditions at every part of making a ticket reservation.

## Conclusion

The London Musical Ticket System is developed to allow customers to easily book tickets of their favourite shows from London Musical. The GUI is robust, scale-able and incorporates modern UI design principles. The source-code is also well organized and can be easily understood.

The London Musical Ticket System handles many error cases and prompts user about any unpleasant condition. The GUI is also designed in a way so that user will have to do minimum amount of work.

## Further Development

This project can be extended and a lot of features and enhancements can be added. Some of the enhancements are discussed below:

1. The GUI can be incorporated with eye-catching images and audio/video visuals to make it more attractive for users.
2. The ticket system could be made autonomous so that it could load .CSV files on startup by intelligently identifying latest CSV files
3. The ticket system could be made further intelligent by incorporating REST API ecosystem so that it automatically fetches information about latest shows on its own.
4. The GUI could include images about Venues, Musical shows and celebrities etc.
5. The ticket system could integrate Google Map to show geographical coordinates of the venues where shows are going to be happen.
6. The ticket system could incorporate Email feature so that an email about reservation is also sent to the Customer.
7. The ticket system could include Email reminders so that it could be notified to Customer about their reservations.
8. The ticket system could also include email subscription feature so that subscribed customers could be notified about upcoming shows.

Aforementioned are some of the enhancements which could be added into this Musical Ticket System in order to make it more robust, user-friendly and business grower.

## Reflection

# Appendix

## Appendix B

White-Box Testing Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **T-No:** | **Test** | **Input** | **Condition** | **Results** |
| 1 | Disable GUI components | state = false | shows.isEmpty() == false | Couple of GUI components are disabled |
| 2 | Disable GUI components | state = false | shows.isEmpty() == true | No change in GUI as there is no data |
| 3 | Enable GUI components | state = true | shows.isEmpty() == true | No change in GUI as there is no data |
| 4 | Enable GUI components | state = true | shows.isEmpty() == false | GUI components are enabled |
| 5 | Print Receipt button | total\_tickets | total\_tickets > 0 | Print Receipt button is enabled |
| 6 | Print Receipt button | total\_tickets | total\_tickets < 0 | Print Receipt button is disabled |
| 7 | Populate Show Slots combo box | Array of shows | Shows is empty | Slots combo-box is not filled |
| 8 | Populate Show Slots combo box | Array of shows | Shows is not empty | Slots combo-box is filled with relevant information |
| 9 | Populate Show Dates combo box | Array of shows | Shows is empty | Dates combo-box is not filled |
| 10 | Populate Show Dates combo box | Array of shows | Shows is not empty | Dates combo-box is filled with relevant information |
| 11 | Cancel an already booked ticket | ticketsReserved | ticketsReserved is empty | A ticket cannot be canceled and removed from ticketsReserved array |
| 12 | Cancel an already booked ticket | ticketsReserved | ticketsReserved is not empty | A reserved ticket is canceled and removed from ticketsReserved array |
| 13 | JavaDB server | JavaDB connection | If server is not started | An error message is displayed |
| 14 | JavaDB server | JavaDB connection | If server is started | Connection with MusicalTicketSystem database is created |
| 15 | CSV file is invalid | CSV file | CSV file is empty | Data cannot be read and database cannot be populated |
| 16 | CSV file is invalid | CSV file | CSV file contains invalid number of columns | Data cannot be read and database cannot be populated |
| 17 | CSV file is valid | CSV file | CSV file contains valid | Data is read and database is also populated |
| 18 | Price filtering | Price filter text | Price is not numeric | Data is not filtered and error popup is displayed |
| 19 | Price filtering | Price filter text | Price is only numeric | Data is filtered and Shows table is updated |
| 20 | Customer Name | Customer name textbox | Customer name contains special characters | Ticket reservations fails and error dialog pop is displayed |
| 21 | Customer Name | Customer name textbox | Customer name contains only alphanumeric and underscore | Ticket reservation is successful |
| 22 | Customer Email address | Customer email textbox | Customer email does not follow correct format | Ticket reservations fails and error dialog pop is displayed |
| 23 | Customer Email address | Customer email textbox | Customer email follows correct format | Ticket reservation is successful |