# Java Entertainment Scheduling System

Group:
Iyana Taylor (2209566)
Lamar Dixon (2209591)
Devonic McDonald (2101569)
Antonio Goldson (2206840)
Raul Miller (2210179)

University of Technology, Jamaica
Faculty of Engineering and Computing
School of Computing and Information Technology

Tutor: Dr Julian Jarrett
CIT3009 Advanced Programming (UN2)

Due: March 23, 2025

# Table of Content

# Project Report

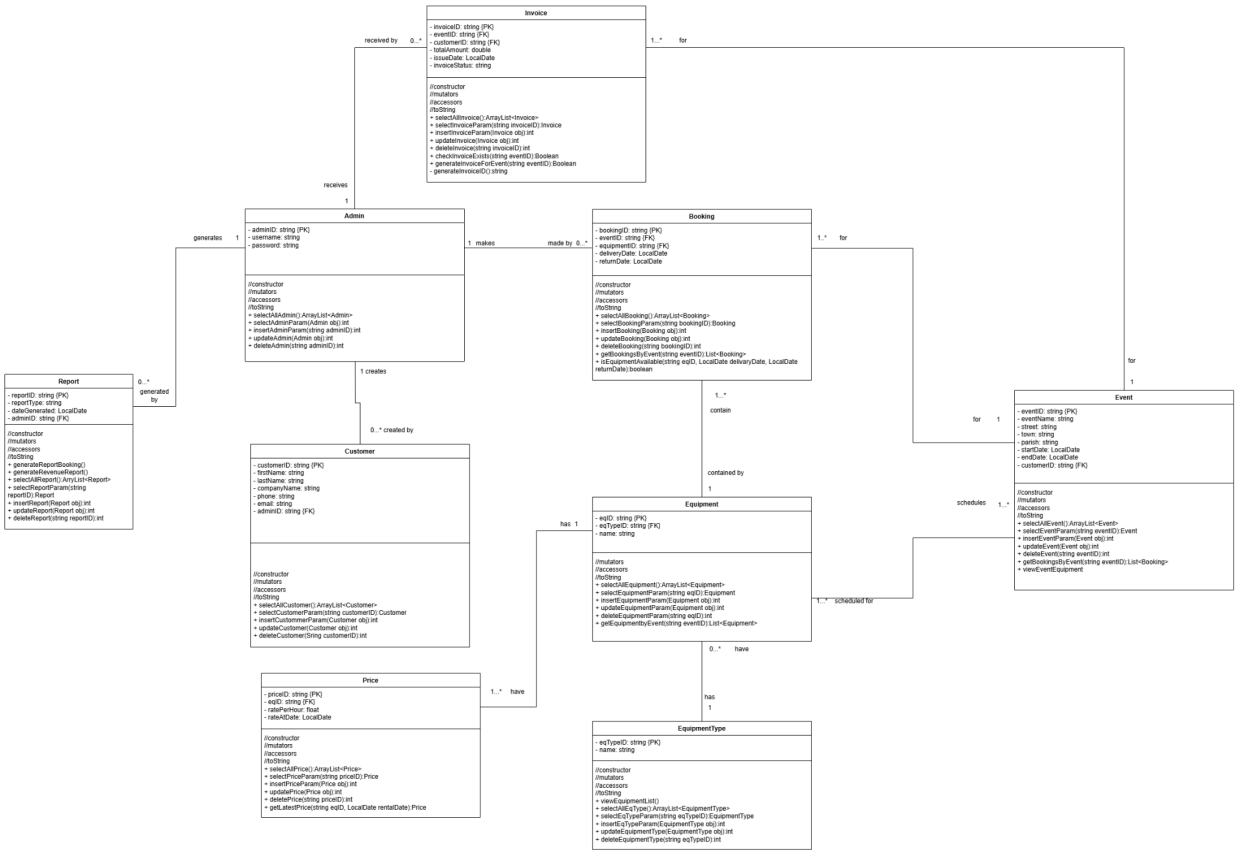
## Summarized Project Description

This project involves creating an Event Scheduling System for Java Entertainment, a company that rents equipment for events. The goal is to help them manage their inventory, schedule equipment without double-booking, handle billing, and optionally generate reports on bookings and revenue. The system will be built as a Java-based Client/Server application with a user-friendly interface, allowing clients to interact with a centralized database. Extra features include support for multiple users through a threaded server and the ability to export reports to PDF. The project should focus on clean database design, intuitive functionality, and scalability to meet the company's needs during the busy Spring Break season.

## Member Contributions

| ***Group Member*** | ***Contribution*** |
|---|---|
| Iyana Taylor | ERD, UI Design, Database Schema,<br>Client-Server Setup, Database Connection,<br>Editing & Testing<br>Models (Domains)<br>Views<br>Controllers<br>SQL Provider<br>Interfaces & SQL Providers per Domain |
| Lamar Dixon | ERD, UI Design, Database Schema<br>Views<br>Controllers |
| Devonic McDonald | ERD, UI Design, Database Schema<br>Views<br>Controllers |
| Antonio Goldson | UI Design, Database Schema |
| Raul Miller | Database Schema |

## Entity-Relationship Diagram



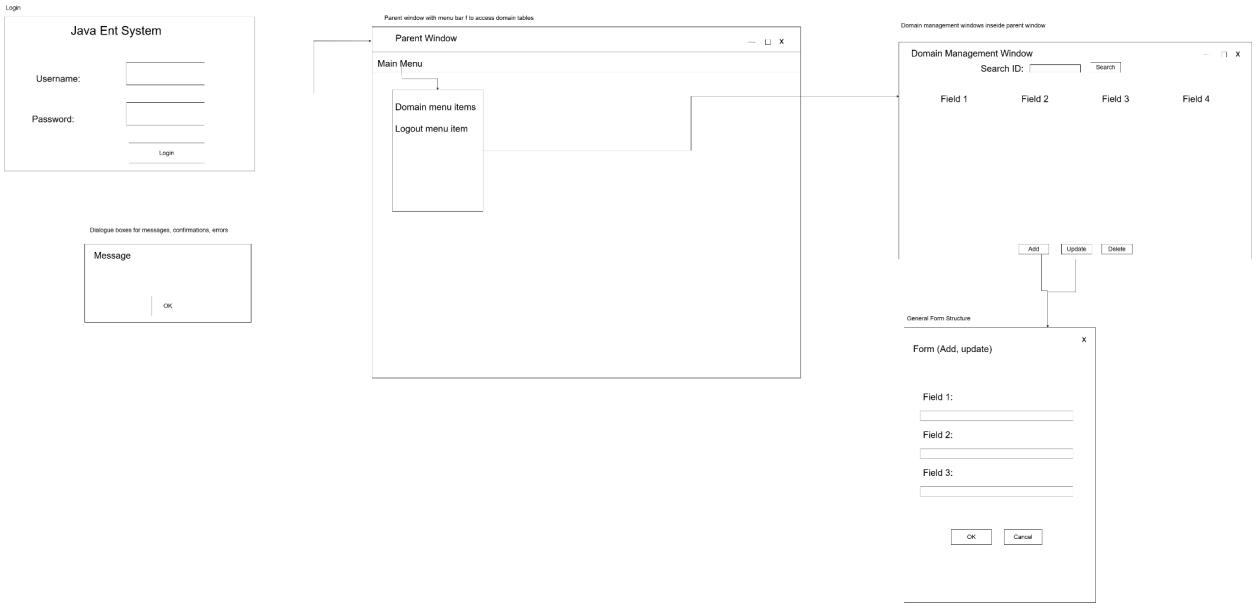

A PDF copy is included in the project ZIP for clearer viewing.

## 3NF Table Structure

| Table Name | Primary Key (PK) | Foreign Keys (FK) | Attributes |
|---|---|---|---|
| **Admin** | adminID | - | username, password |
| **Customer** | customerID | - | firstName, lastName, companyName, phone, email |
| **Event** | eventID | customerID | eventName, location, startDate, endDate |

| | | | |
|---|---|---|---|
| **Booking** | bookingID | eventID, equipmentID | deliveryDate, returnDate |
| **Invoice** | invoiceID | eventID, customerID | totalAmount, issueDate, invoiceStatus |
| **Equipment** | eqID | eqTypeID | name |
| **Price** | priceID | eqID | ratePerHour, rateAtDate |
| **EquipmentType** | eqTypeID | - | name |
| **Report** | reportID | - | reportType, dateGenerated |

## User Interface Design




A PDF copy is included in the project ZIP for clearer viewing.

# Developers' Notes

## Development Plan

### *Project Setup & Core Design (Entity Design, Database, Architecture)*

Goal: Lay the groundwork for the system

Tasks:

*Database & Class Design (20 marks)*

- ☐ Design **Entity-Relationship Diagram (ERD)** & **Class Diagram**
- ☐ Ensure **3rd Normal Form (3NF)** normalization
- ☐ Define foreign keys & many-to-many relationships
- ☐ Get approval if required
- ☐ Design UI mockups

*Project Setup*

- ☐ Set up a **GitHub repo** for version control
- ☐ Create the **Java project in NetBeans**
- ☐ Set up **database schema in MySQL/PostgreSQL**
- ☐ Create **base classes** for major entities
- ☐ Consider hashing the password

*Client-Server Architecture Design (15 marks)*

- ☐ Create the **server** (Java socket programming)
- ☐ Create the **client** (Java GUI with basic forms)
- ☐ Set up **network communication (TCP/IP sockets)**

### ***Database & Core Functionality Implementation***

Goal: Implement database connectivity & CRUD operations

Tasks:

*Database Connectivity (15 marks)*

- ☐ Set up **JDBC connection** to MySQL/PostgreSQL
    - ☐ add connector j and log core & API JARs to the project library

- ☐ Start with **domain entities first**
- ☐ Set up **SQLProvider**
- ☐ Interfaces for each entity (IDomainNameScv)

*Core Functions (15 marks)*

- ☐ SQLProvider for each entity (extends SQLProvider & implements the entity interface)
    - ☐ Implement **CRUD operations**
        - ☐ Ensure **no double bookings** (Constraints in DB)
- ☐ Import domains, interfaces and SQL providers in the server
- ☐ Setup the server
- ☐ Server driver

*Exception Handling & Logging (10 marks)*

- ☐ Implement **try-catch blocks** for database & network errors
- ☐ Use **Java logging framework** (e.g., Log4J) for error logs
    - ☐ add logging for dbtools SQLProvider class in the Server project
    - ☐ logging for server class in the Server project

***User Interface + Advanced Features***

Goal: Build the GUI & implement scheduling and billing

Tasks:

*Graphical User Interface (15 marks)*

- ☐ Set up the client
- ☐ The models for the GUI are the domains, copy them from the Server to the Client project
- ☐ Build **Java Swing GUI**
    - ☐ Do the views for each domain
        - ☐ Implement **Table Views** for data display
        - ☐ Add **forms for data entry**
        - ☐ Add buttons for crud operations
- ☐ Do the controllers for each view (thread the server earlier than intended - for testing)
- ☐ Complete parent window

*Billing Module (Invoices, Receipts)*

- ☐ Implement invoice creation (Client requests invoice → Server fetches data → Generates total cost)
    - ☐ Implement invoice generation based on event & booking
    - ☐ Download the invoice from the invoice table
- ☐ View equipment stored under a particular equipment type (it's just a method in EventSQLProvider, did not set a way to access it via the UI)

*Generics & Collections (5 marks)*

- ☐ Use **ArrayLists, HashMaps, and generics** to manage objects efficiently

***Threading, Reports & Finalizing Features***

Goal: Add multi-user support & reporting features

Tasks:

*Threading (5 marks)*

- ☐ Convert **server into a multi-threaded** server
- ☐ Handle **multiple client requests simultaneously**

*Reporting Module (5 marks)*

- ☐ Implement **date-based booking reports**
- ☐ Implement **revenue reports** (total revenue, per-day revenue)
- ☐ Use **JasperReports or iText for PDF generation** (extra credit)
- ☐ Downloading reports between generated between certain dates

*User Management*

- ☐ Implement **Login & User Roles (Admin, Staff, Customer) - only Admin for now**
- ☐ Implement logout logic
    - ☐ disconnect the client
    - ☐ Go back to the login window

***Final Testing, Documentation & Submission***

Goal: Test everything & prepare documentation

Tasks:

*Test Cases & Debugging*

- ☐ Write test cases (about 15-20)
  - ☐ **Unit test core functions** (JUnit for CRUD operations)
  - ☐ **Integration test Client-Server communication**
  - ☐ **Edge case testing** (e.g., invalid bookings, DB connection loss)

*Code Cleanup & Documentation (5 marks)*

- ☐ Ensure **proper indentation & naming conventions**
- ☐ Remove the plain selections and inserts from the interfaces and specific SQL providers (only keep the parameterized ones)
- ☐ Remove unnecessary functions and comments
- ☐ Add **comments** for all functions
- ☐ Write a **README file** (Setup Instructions, Features) - documentation

*Prepare Submission*

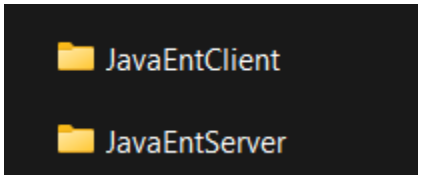
- ☐ Double-check the **grading rubric** for completion
- ☐ Finalize & **submit project (zipped folder with client, server and documentation)**
- ☐ Save files, screenshots etc to the outer project folder (the submission folder will be separate)
- ☐ Prep for demo

**User Manual**

For testing

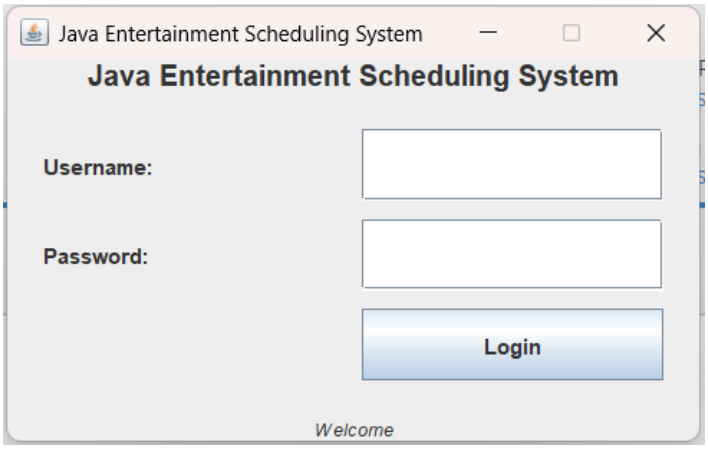
Part A - General

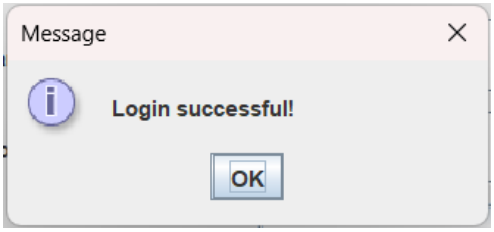
1. Ensure Client and Server project folders are saved to your computer.



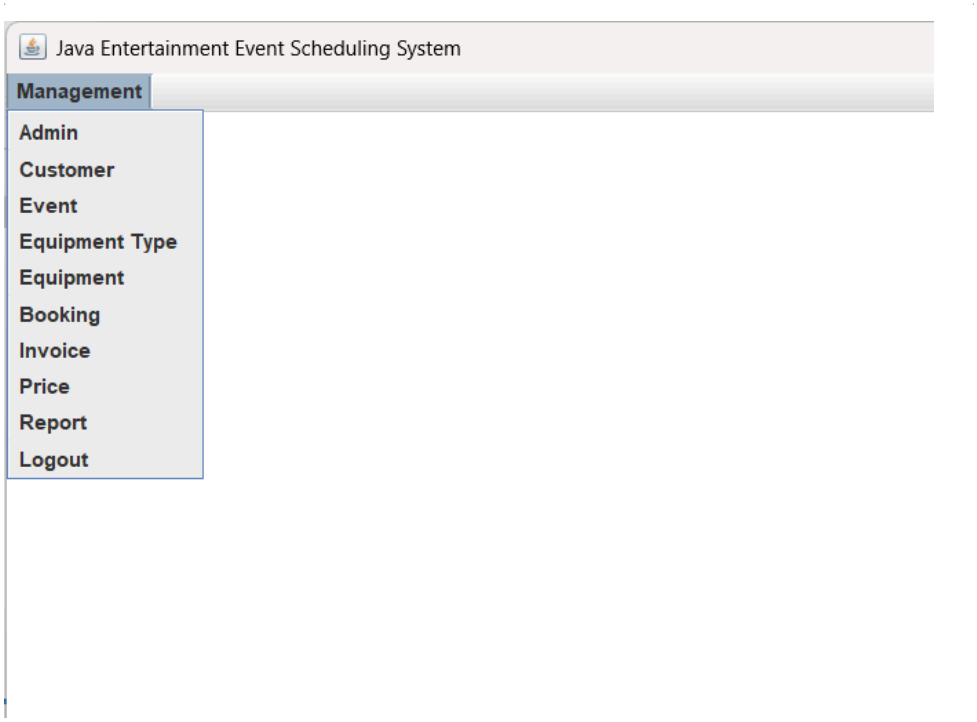

2. Do not remove anything, removals/tampering with the project structures might cause them to break.
3. Open/ Import both projects into your IDE of choice.
4. Clean and Build both before running.
5. Upon successful completion of clean and build, run the server first. It needs to be live to accept client connection when necessary.
6. Keep the server running then run the client.

7. After the network connection is established between the client and server, the login window of the system will appear.

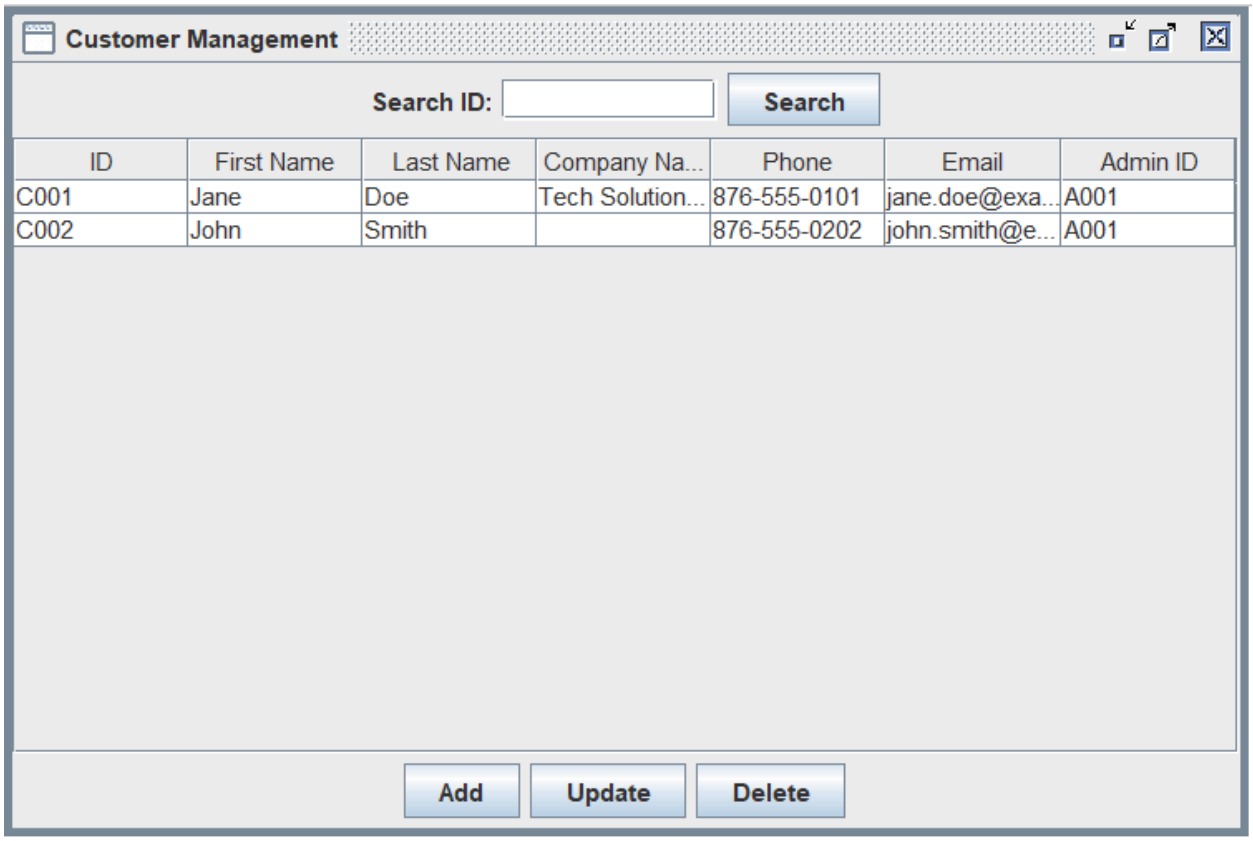


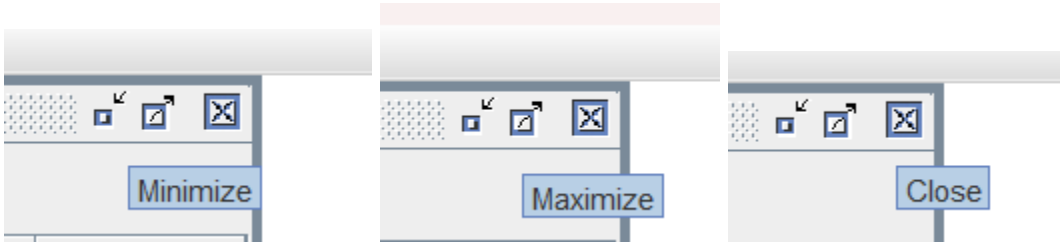

8. Login credentials are read from the database (script provided in the project folder). To authenticate users the system matches the username and password entered with the admin table in the database. The only admin record in the database table at this time is: adminID: A001, username: **adminUser**, password: **securePass123.** Enter these credentials for your first successful login.
9. After successful login, the login window and dialogue will be replaced by the parent window of the system. This window has a single “Management” tab with menu items to access the tables.


10. Simply click a menu item to see its corresponding table. These are sub-management windows, more than one can be open at once. Click anywhere in the window to make it the active window. And move them around as you open one without closing the others as the most recently opened one appears in the top left corner. More than one instance of the same window can be opened, be double-check to ensure you are operating in the correct window.

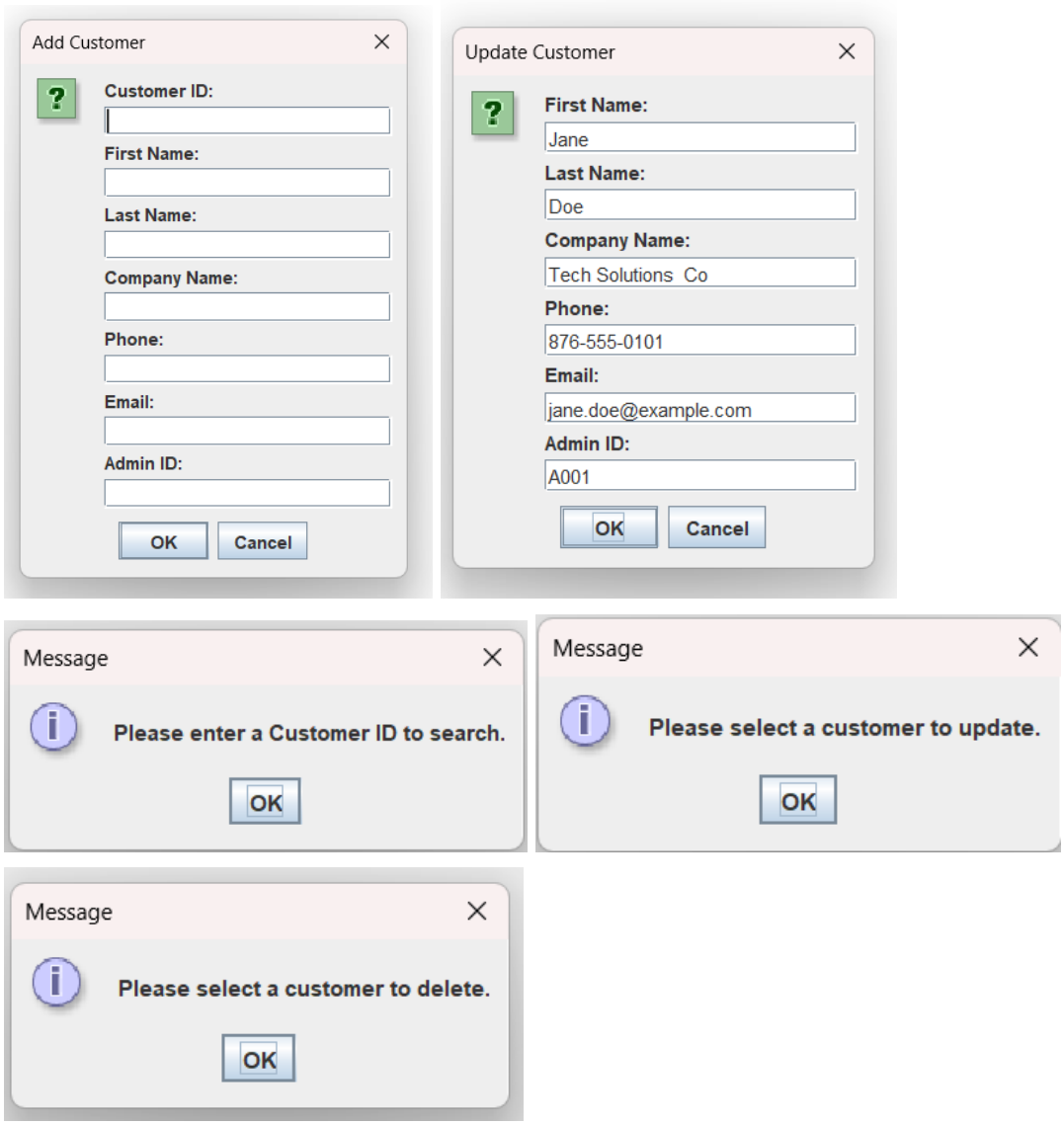



11. All windows can be iconified and maximized depending on your viewing needs.

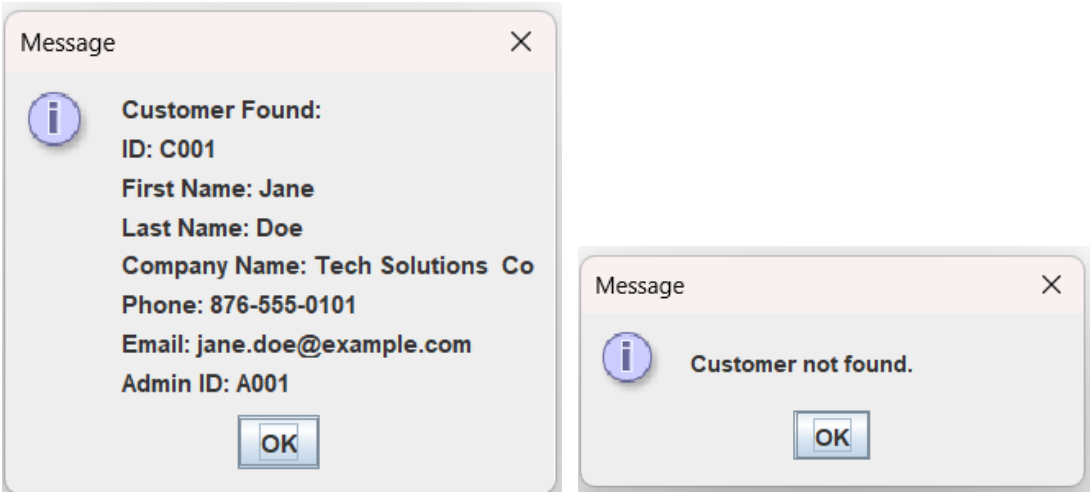




12. All tables have basic ADD, UPDATE, DELETE and SEARCH operations. Note that a record must be selected before you can update or delete it, and an ID must be entered into the text field before you can search.

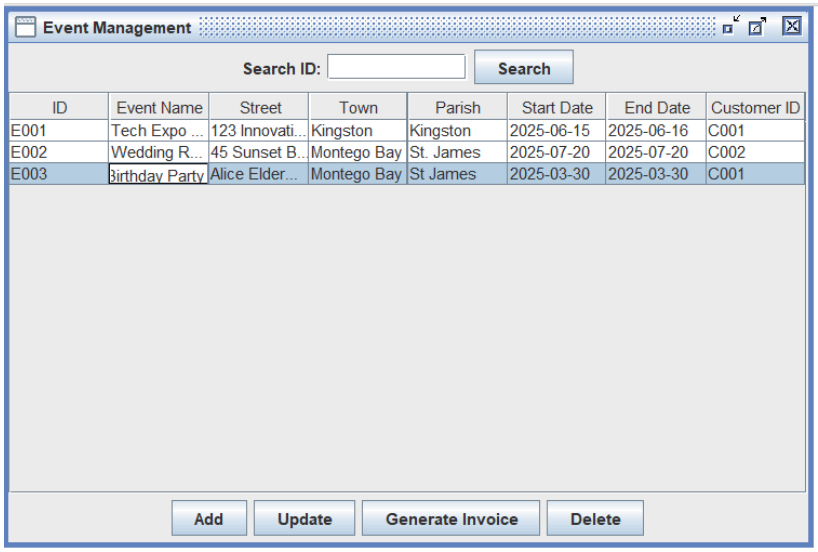



13. Be careful in your attempt to delete records, a dialogue box appears to confirm deletion.
14. Selections will be cleared after update and delete operations are carried out.
15. When searching you will either of two dialogue boxes based on the results of your search.

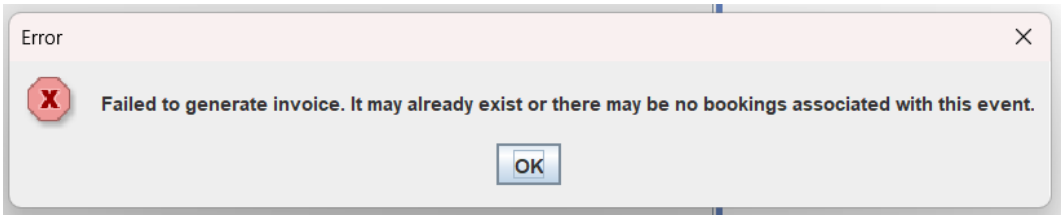



Part B - Event Invoice Generation

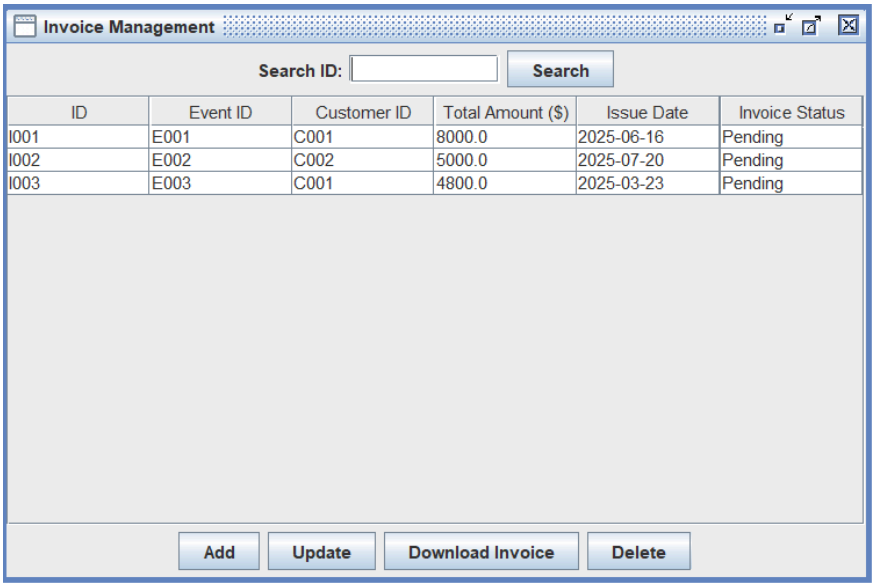
16. When generating invoices a row must be selected before the invoice generation button is pressed.



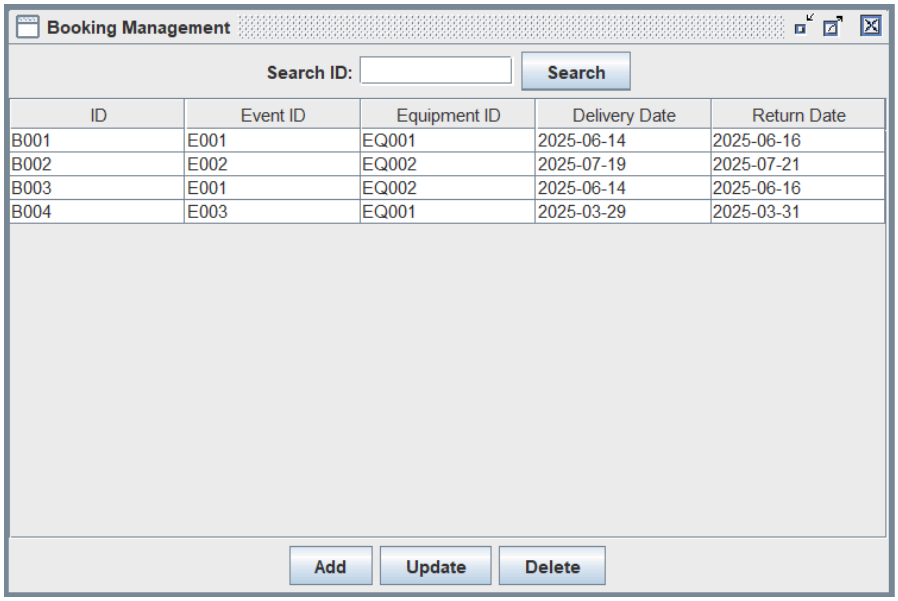

a. If an invoice for that Event already exists OR there are no bookings for that event this dialogue box will come up:




An invoice generation attempt was made for Event ‘E003’ but there is already an Invoice for that event:

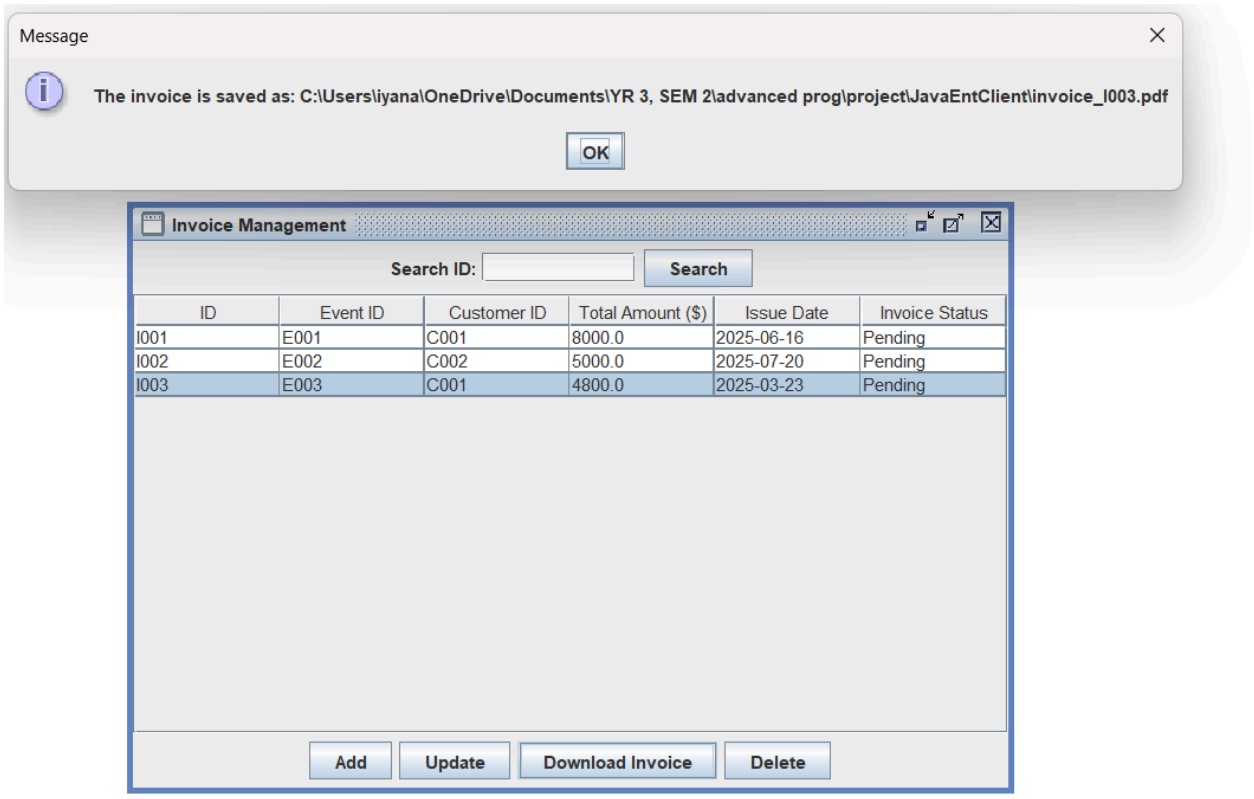



17. Do ensure that after adding a new event, bookings are made for that event if you wish to generate an invoice for it ('it' being the event).



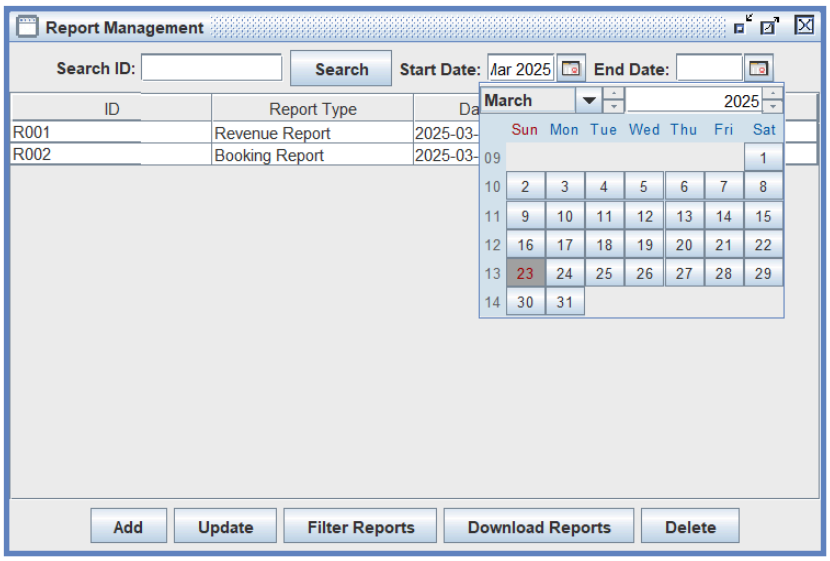

Part C - Downloading Invoices

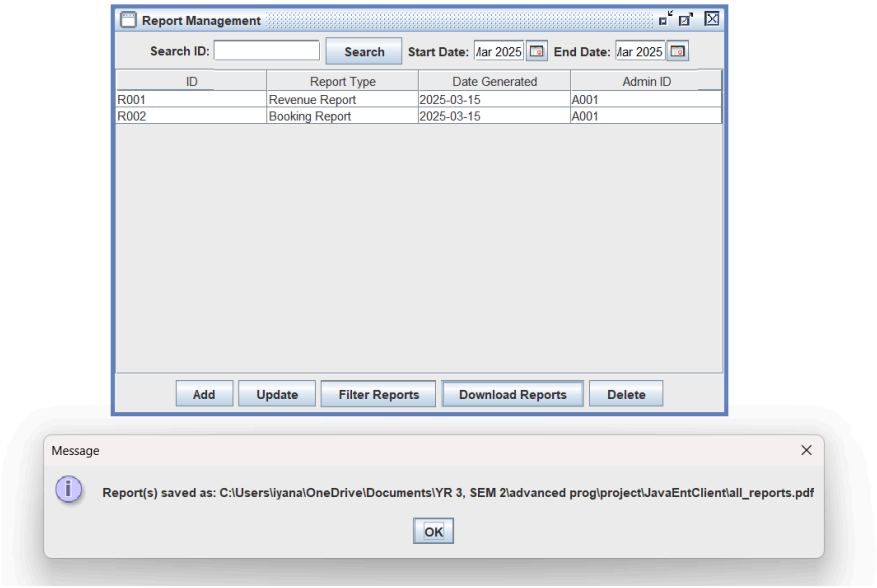
18. To download an invoice ensure that invoice record is selected. The invoice will be saved as a PDF (typically in the same folder as the Client Project files and folders), pay attention to the path to see exactly where the invoice is stored or make a note of the name (at the very end of the path) and search for it in your devices file explorer.



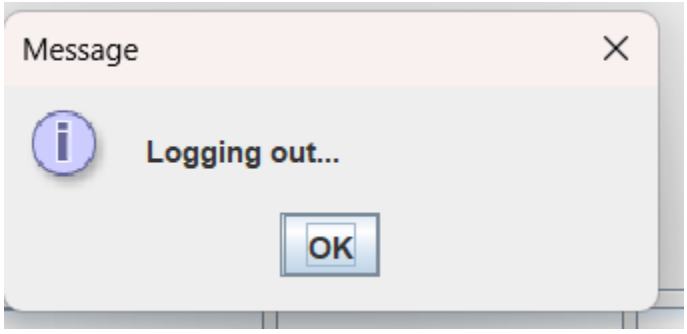

Part D - Reports

19. Reports generated between certain dates can be filtered and downloaded all together or a single record selected and downloaded.






20. Upon logging out you will be taken back to the login window and the client will close the connection (clicking the ‘x’ also closes the client connection).




21. The server may be stopped via the IDE or the server tray icon (blue cabinet icon here).