Kingdom of Saudi Arabia
Qassim University
College of Computer
Computer Science

EVENT REGISTRATION AND TICKET MANAGEMENT SYSTEM

PRESENTED TO	PRESENTED BY
Dr. Rafef Alsuhaibani	**LAMA ALGHOFAILI (TEAM LEADER)** **MEMBER 2** **MEMBER 3** **MEMBER 4** **MEMBER 5**

Table of Contents

Application Overview

What Is the Application?

The Event Management System is a desktop application built using Java Swing and MySQL that allows different types of users to manage and participate in events through a secure and user-friendly interface. The system supports three main roles: Administrator, Organizer, and Attendee, each with specific permissions and functionalities.

The application provides user authentication, allowing users to securely sign up, log in, and reset their password. Administrators can manage system data, including viewing users, deleting users, managing events, and generating global system reports. Organizers can create, edit, and delete events, monitor registrations, and view event statistics. Attendees can browse available events, register for an event, receive a unique ticket, and view their registered events.

The system integrates directly with a MySQL database, handling event creation, seat availability, ticket generation, user data, and registration records. It also includes built-in transaction management to ensure safe seat updates and prevent over-booking.

Overall, the application provides a complete workflow for managing events— from creating events, registering attendees, generating tickets, and analyzing system activity through advanced reports—making it a full event-management solution.

Project Purpose

The purpose of this application is to provide a complete and efficient system for managing events, where administrators, organizers, and attendees can interact through a secure and user-friendly interface. The system aims to simplify event creation, registration, ticket generation, and reporting, while ensuring data accuracy, security, and smooth communication between all users

Application Overview

Core Functionalities of the Application

1. User Authentication & Authorization
• Users can create accounts (Admin, Organizer, Attendee).
• Secure login using hashed passwords and salts.
• Access level depends on user role (RBAC).
2. Event Management (Organizer / Admin)
• Create new events with title, date, time, category, location, and capacity.
• Edit existing events.
• Delete events.
• View all events in the system.
3. Event Registration (Attendees)
• Users can view a list of available events.
• Register for an event if seats are available.
• System updates seats automatically using transaction handling.
4. Ticket Generation
• System generates a unique ticket ID for each registration.
• QR-code-like data is stored for ticket verification.
5. Reports & Statistics
• Event statistics (capacity usage, registrations, remaining seats).
• User distribution by role (Admin, Organizer, Attendee).
• Most popular categories.
• Most registered events.
• Full system report (users, events, registrations).
• Organizer reports for specific events.
6. Database Integration
• MySQL database for storing users, events, tickets, and registrations.
• Uses DAO pattern for clean and structured database operations.
7. Transaction Management
• Ensures that ticket creation, registration, and seat reduction happen safely together.
• Rolls back in case of any error.

Application Overview

Key Points of the System

1- Secure Login & User Management
The system relies on secure authentication with role-based access for different user types.

2-Complete Event Management
Events can be added, edited, deleted, and viewed, with automatic seat updates.

3- Reliable Registration Process
Ensures seat availability, prevents duplicate registrations, and generates unique tickets.

4- Database Integration with Safe Transactions
All data is stored in MySQL with transaction safety to prevent over-booking or data inconsistency.

5- Useful Reports & Statistics
Provides system insights such as total events, registrations, users, and capacity usage

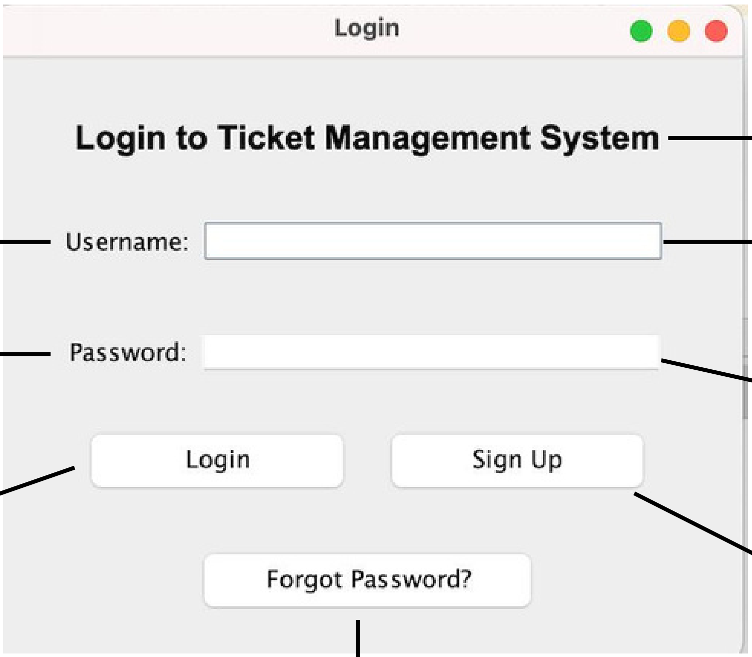
GUI Elements

The graphical user interface (GUI) of the Ticket Management System is designed to provide a clear, structured, and user-friendly experience for all system roles: Admin, Organizer, and Attendee. Each interface is carefully organized to ensure smooth navigation and efficient task completion.

The GUI includes dedicated screens for user authentication, account creation, password recovery, dashboards for each user role, event management operations (add, edit, delete), event registration, ticket viewing, and detailed system reports. Labels, text fields, buttons, combo boxes, and tables are used consistently across the application to guide users, support data entry, and present system information clearly.

This section showcases all GUI components in the system, explaining the purpose of each screen and describing the function of its main elements to demonstrate how users interact with the application effectively

GUI Elements



Type: Label
Purpose: Indicates the username input field.

Type: TextField
Purpose: Allows the user to enter their username.

Type: Password Field
Purpose: Allows secure password entry.

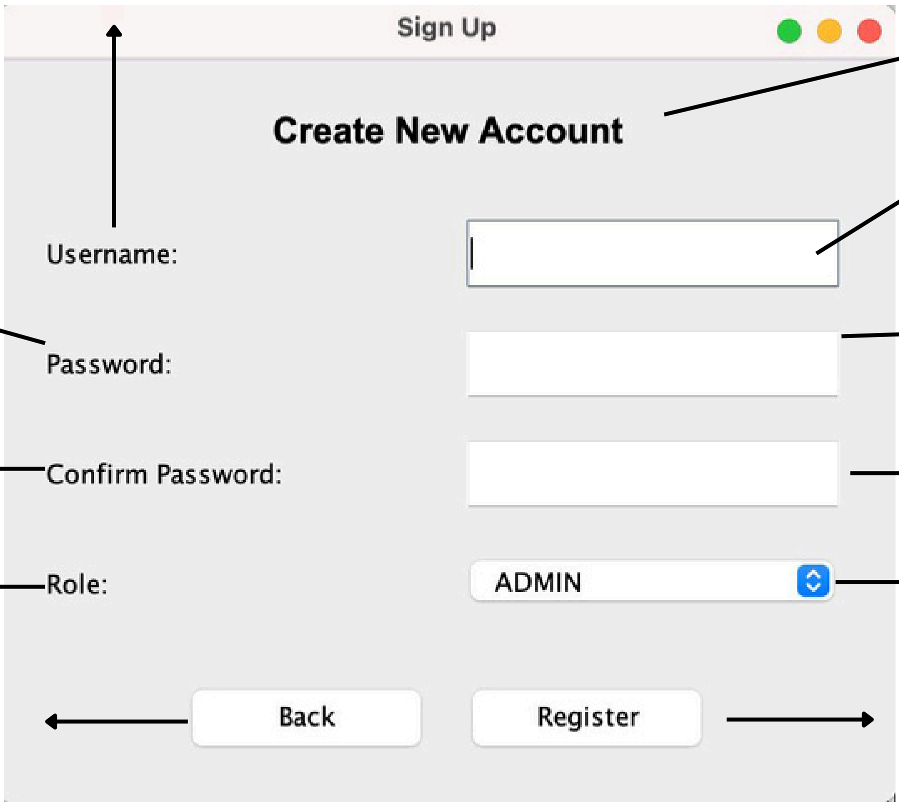
Type: Button
Purpose: Opens the registration screen to create a new account.

Type: Label
Purpose: Indicates the username input field.

Type: Label
Purpose: Indicates the password input field.

Type: Button
Purpose: Authenticates the user and logs them in.

Type: Button
Purpose: Starts the password recovery process.



Type: Label
Purpose: Indicates the username input field.

Type: Label
Purpose: Displays the main title of the Sign-Up page.

Type: Text Field
Purpose: Allows the user to enter their username.

Type: Label
Purpose: Indicates the password input field.

Type: Password Field
Purpose: Allows secure entry of the user's password.

Type: Label
Purpose: Indicates the field for re-typing the password.

Type: Password Field
Purpose: Allows the user to re-enter the password for confirmation

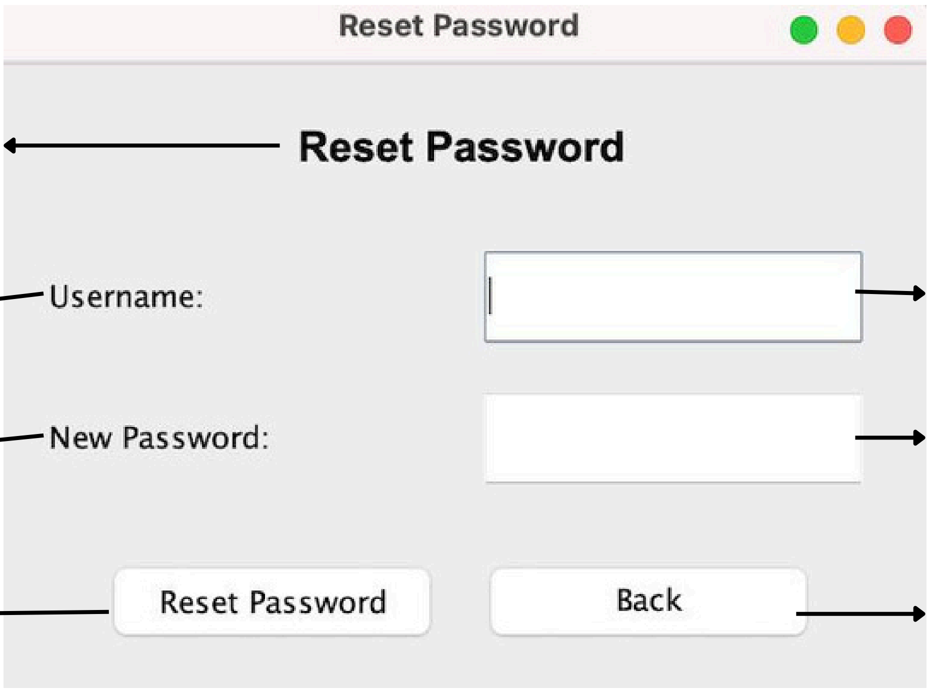
Type: Label
Purpose: Indicates the selection of user role.

Type: Combo Box
Purpose: Allows choosing the account role.

Type: Button
Purpose: Returns to the previous screen without registering.

Type: Button
Purpose: Creates the new user account and saves it to the database.

GUI Elements


Reset Password

Reset Password

Username:

New Password:

Reset Password

Back

Type: Label
Purpose: Displays the title of the reset password screen.

Type: Label
Purpose: Indicates the field for entering the username.

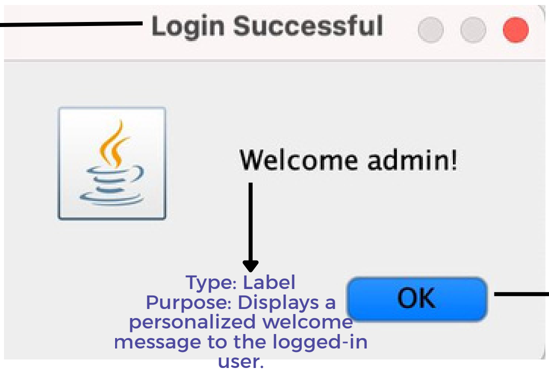
Type: Label
Purpose: Indicates the field for entering the new password.

Type: Button
Purpose: Updates the user's password in the system.

Type: Text Field
Purpose: Allows the user to enter their username.

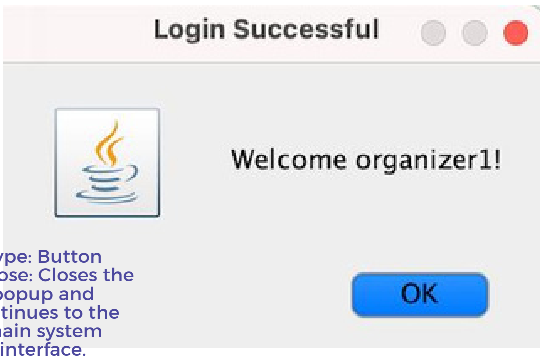
Type: Password Field
Purpose: Allows the user to enter a new password.

Type: Button
Purpose: Returns the user to the login screen.


Login Successful

Welcome admin!

OK


Login Successful

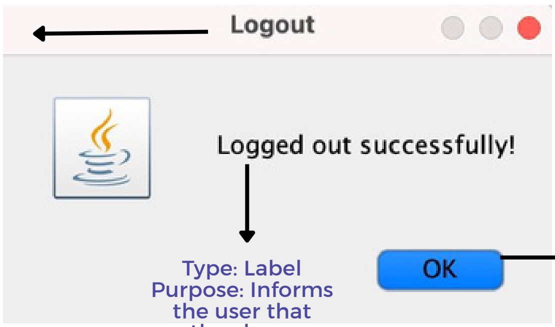
Welcome organizer1!

OK

Type: Label
Purpose: Shows the success message after the user logs in.

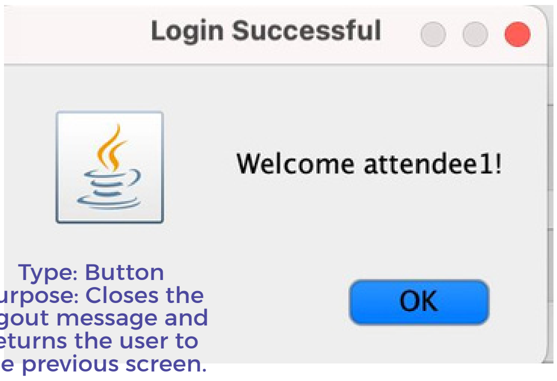
Type: Label
Purpose: Displays a personalized welcome message to the logged-in user.

Type: Button
Purpose: Closes the popup and continues to the main system interface.


Logout

Logged out successfully!

OK


Login Successful

Welcome attendee1!

OK

Type: Title Label
Purpose: Shows the title of the logout confirmation window.

Type: Label
Purpose: Informs the user that they have successfully logged out.

Type: Button
Purpose: Closes the logout message and returns the user to the previous screen.

GUI Elements

Type: Label
Purpose: Shows the logged-in admin's ID information.

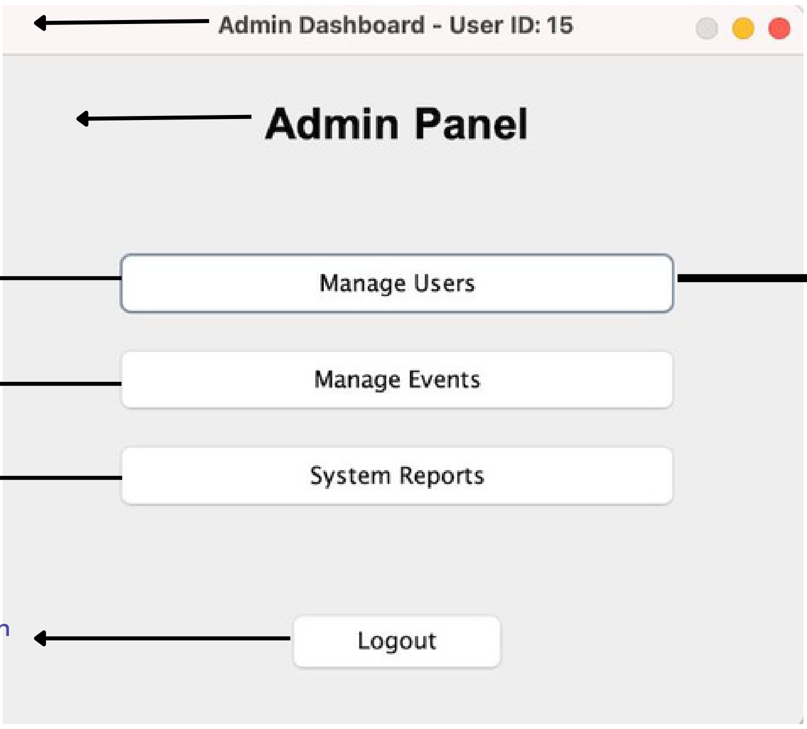
Type: Label
Purpose: Displays the main title of the admin dashboard.

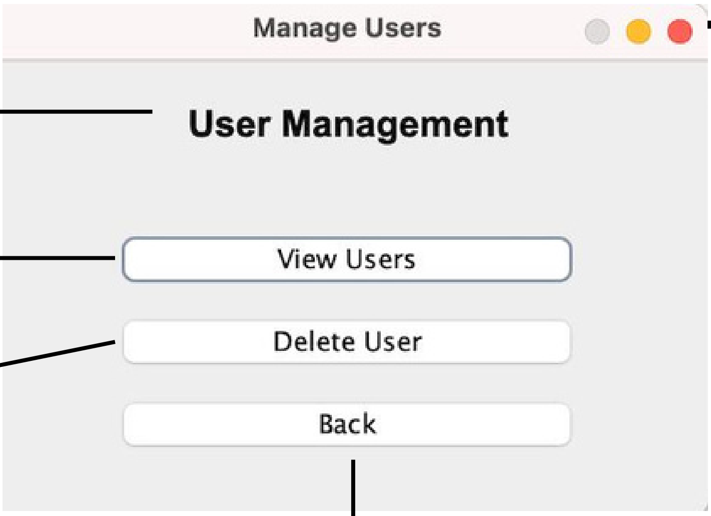
Type: Button
Purpose: Opens the interface for managing system users.

Type: Button
Purpose: Opens the event management section.

Type: Button
Purpose: Displays analytics and system reports.

Type: Button
Purpose: Logs out the admin and returns to the login screen.





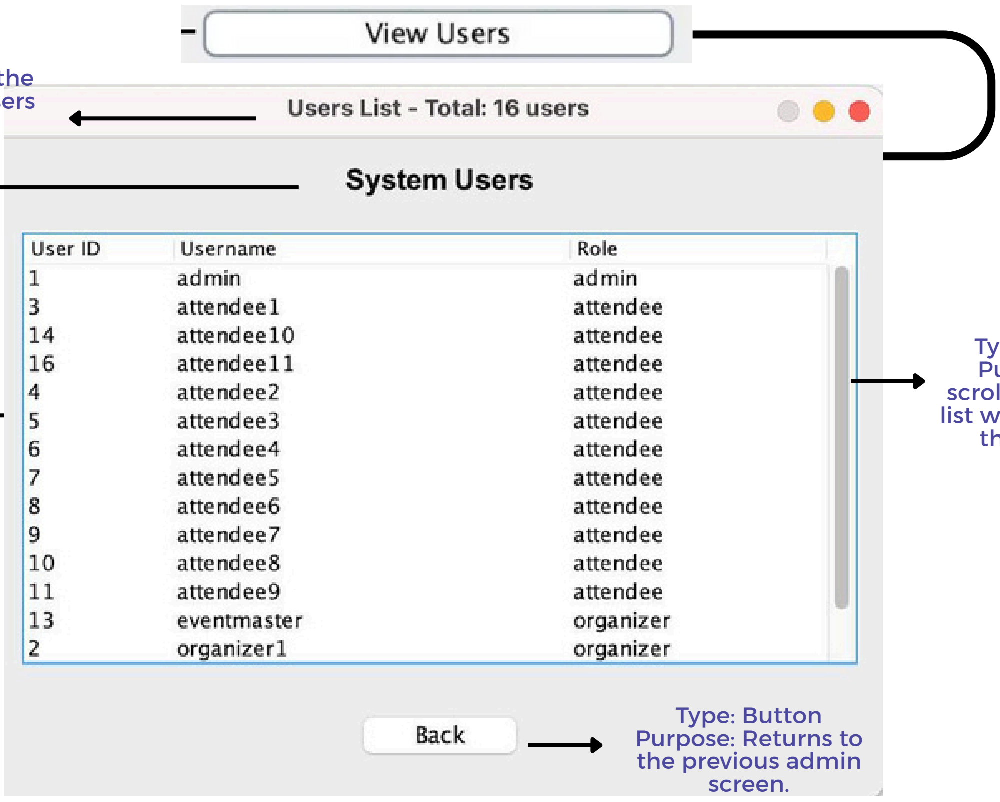
Type: Label
Purpose: Displays the main title of the page ("User Management").

Type: Button
Purpose: Opens a window showing all registered users in the system.

Type: Button
Purpose: Allows the admin to delete a user by selecting them from the list.

Type: Button
Purpose: Returns to the previous Admin dashboard screen.

GUI Elements



User ID	Username	Role
1	admin	admin
3	attendee1	attendee
14	attendee10	attendee
16	attendee11	attendee
4	attendee2	attendee
5	attendee3	attendee
6	attendee4	attendee
7	attendee5	attendee
8	attendee6	attendee
9	attendee7	attendee
10	attendee8	attendee
11	attendee9	attendee
13	eventmaster	organizer
2	organizer1	organizer

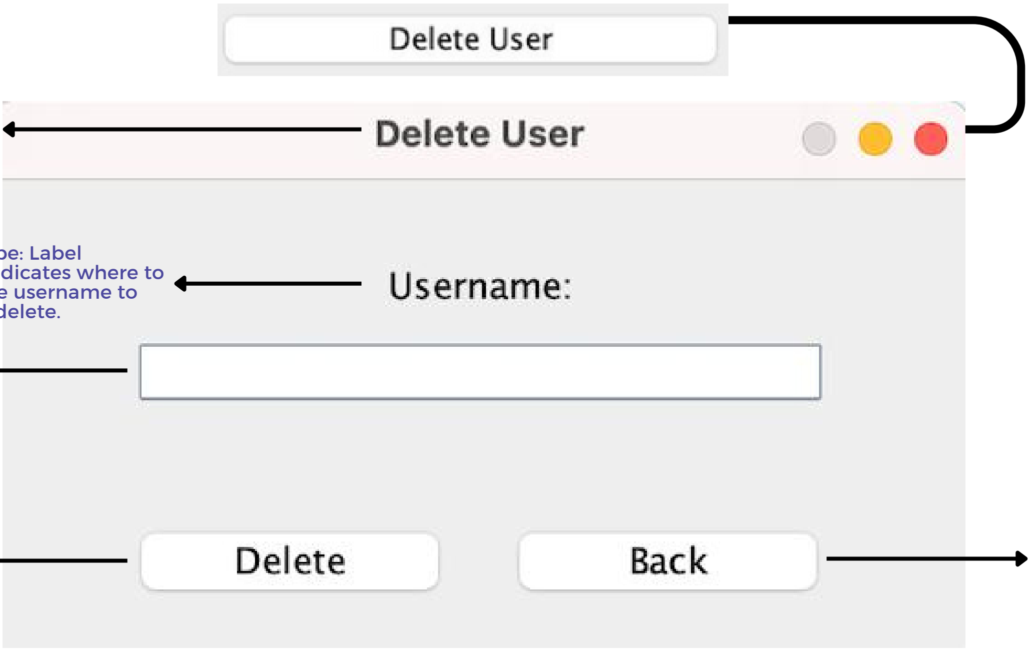
Type: Label
Purpose: Displays the total number of users in the system.

Type: Label
Purpose: Shows the section title ("System Users").

Type: Table (JTable)
Purpose: Displays all system users with their details.

Type: ScrollPane
Purpose: Allows scrolling through the list when users exceed the visible area.

Type: Button
Purpose: Returns to the previous admin screen.



Type: Label
Purpose: Displays the title of the delete user page.

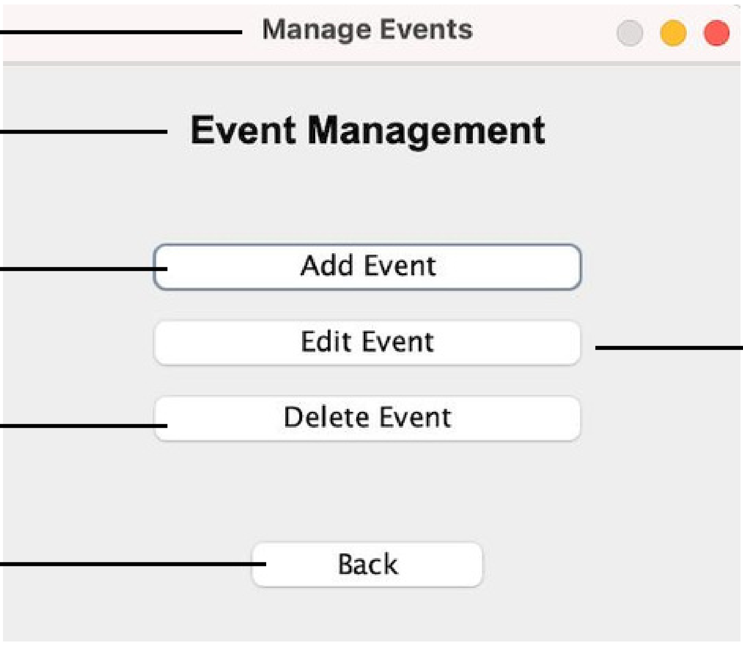
Type: Label
Purpose: Indicates where to enter the username to delete.

Type: Text Field
Purpose: Receives the username of the account to be deleted.

Type: Button
Purpose: Removes the specified user from the system.

Type: Button
Purpose: Returns to the previous admin menu.

GUI Elements



Type: Label
Purpose: Shows the title of the window (“Manage Events”).

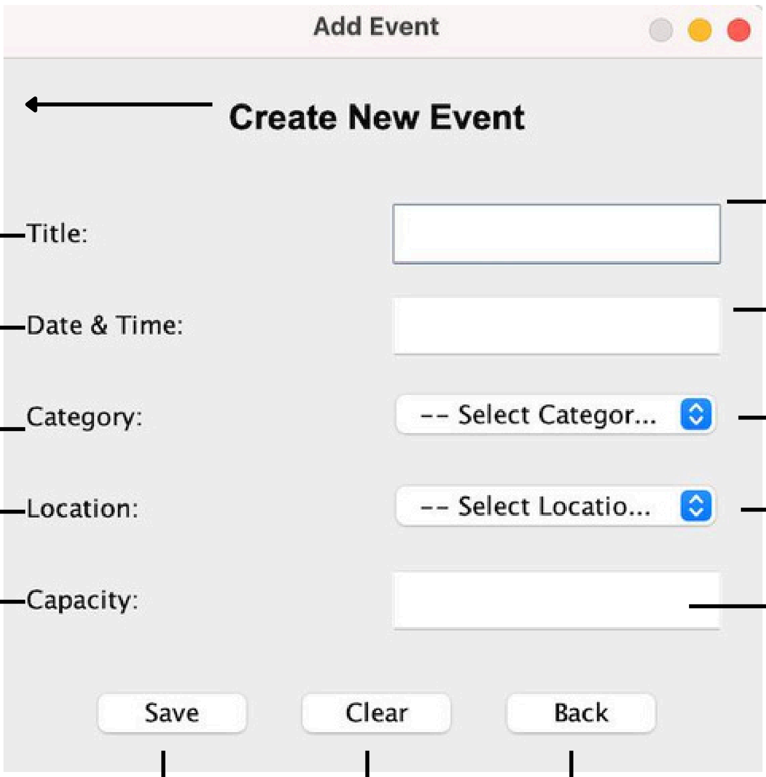
Type: Label
Purpose: Indicates that the screen is for event management.

Type: Button
Purpose: Opens the interface to create a new event.

Type: Button
Purpose: Opens the interface to modify an existing event.

Type: Button
Purpose: Allows the admin/organizer to remove an event.

Type: Button
Purpose: Returns to the previous screen.



Type: Label
Purpose: Displays the page title for adding a new event.

Type: Label
Purpose: Indicates where to enter the event title.

Type: Text Field
Purpose: Allows entering the event name.

Type: Label
Purpose: Shows the required field for choosing the event’s date and time.

Type: Text Field
Purpose: Allows entering the event date and time.

Type: Label
Purpose: Indicates where to select the event category.

Type: Combo Box
Purpose: Allows choosing the event’s category.

Type: Label
Purpose: Indicates where to select the event location.

Type: Combo Box
Purpose: Allows choosing the location.

Type: Label
Purpose: Indicates where to enter event capacity

Type: Text Field
Purpose: Allows entering the maximum number of attendees.

Type: Button
Purpose: Saves the new event into the system.

Type: Button
Purpose: Resets all input fields.

Type: Button
Purpose: Returns to the previous page.

GUI Elements

Type: Label
Purpose: Displays the page title ("Edit Event").

Type: Label
Purpose: Informs the user to select an event to edit.

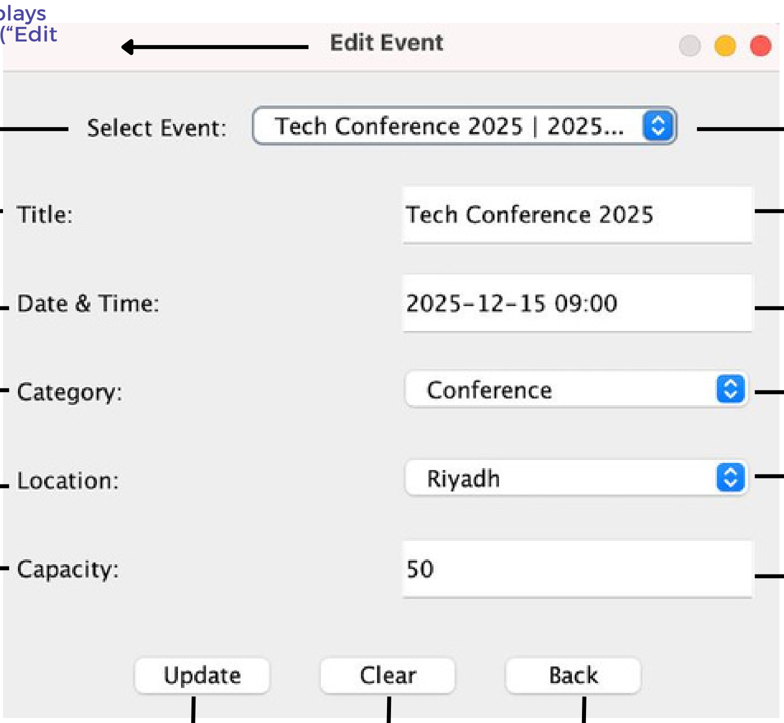
Type: Label
Purpose: Indicates the event title input field.

Type: Label
Purpose: Indicates where to edit the event date and time.

Type: Label
Purpose: Indicates the category selection field.

Type: Label
Purpose: Indicates the event location field.

Type: Label
Purpose: Indicates where to edit event capacity.



Select Event: Tech Conference 2025 | 2025...

Title: Tech Conference 2025

Date & Time: 2025-12-15 09:00

Category: Conference

Location: Riyadh

Capacity: 50

Type: ComboBox
Purpose: Allows choosing an existing event to edit.

Type: TextField
Purpose: Allows the user to edit the event title.

Type: TextField
Purpose: Allows entering the new date and time.

Type: ComboBox
Purpose: Allows choosing a category for the event.

Type: ComboBox
Purpose: Allows selecting a location.

Type: TextField
Purpose: Allows entering a new number of seats

Type: Button
Purpose: Saves the updated event information.

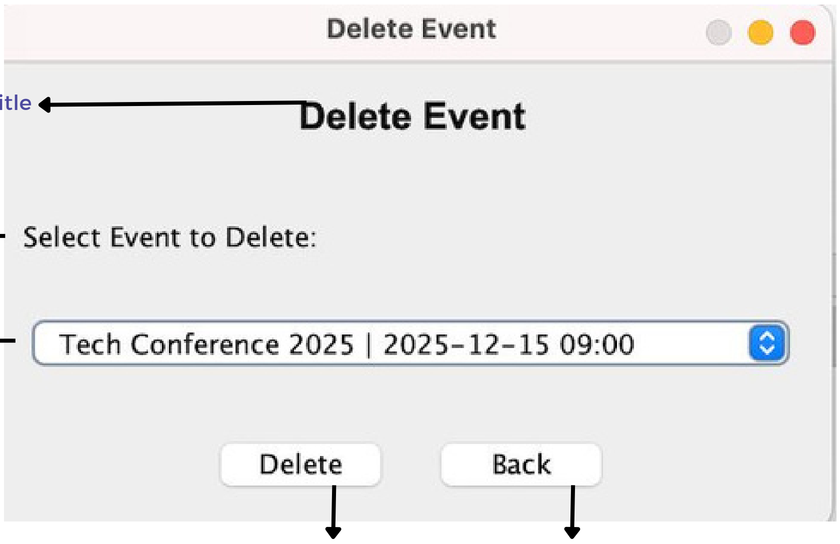
Type: Button
Purpose: Clears all fields for re-editing.

Type: Button
Purpose: Returns to the previous screen.

Type: Label
Purpose: Displays the page title for deleting an event.

Type: Label
Purpose: Tells the user to choose which event will be deleted.

Type: Combo Box
Purpose: Allows selecting the specific event to be removed.



Select Event to Delete:

Tech Conference 2025 | 2025-12-15 09:00

Type: Button
Purpose: Deletes the selected event from the system.

Type: Button
Purpose: Returns to the previous screen.

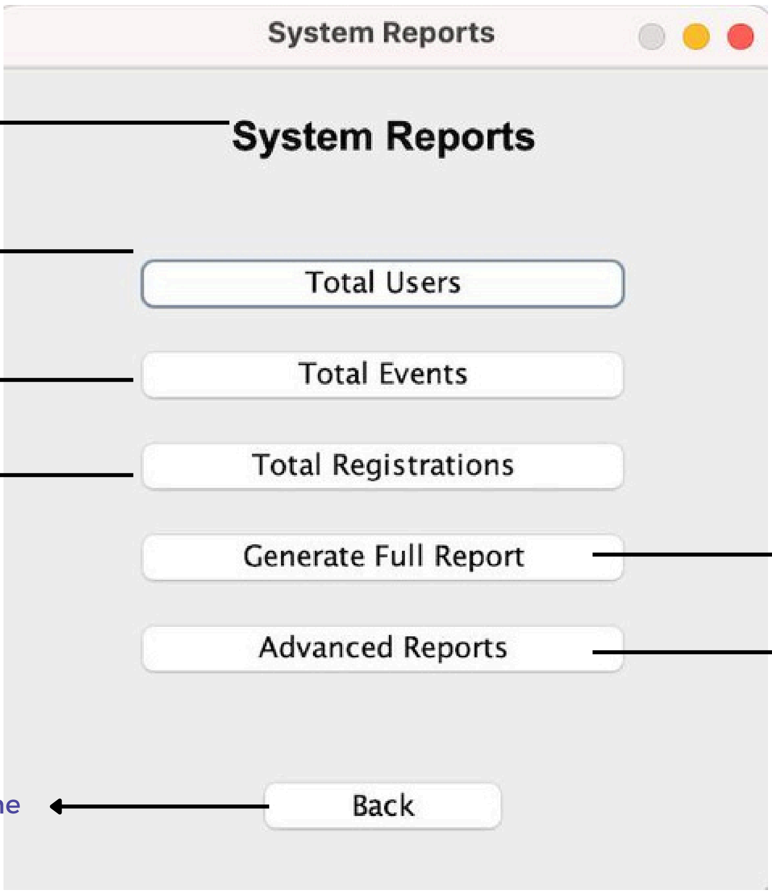
GUI Elements

Type: Label
Purpose: Displays the main title of the reports section.

Type: Button
Purpose: Shows the total number of registered users.

Type: Button
Purpose: Displays the total number of events in the system.

Type: Button
Purpose: Shows how many users registered for events.

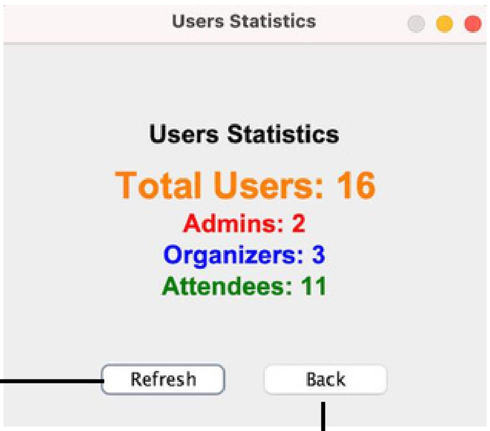


Type: Button
Purpose: Produces a complete system report.

Type: Button
Purpose: Opens detailed analytics and advanced reporting options.

Type: Button
Purpose: Returns to the previous dashboard.

Shows the total number of users in the system and how many are admins, organizers, and attendees.



Users Statistics

Total Users: 16

Admins: 2

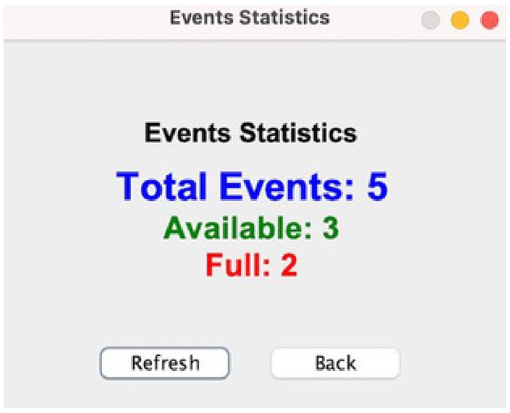
Organizers: 3

Attendees: 11

Type: Button
Purpose: Updates the screen to show the latest data

Type: Button
Purpose: Returns the user to the previous page.

Displays the total number of events and how many are available or fully booked.



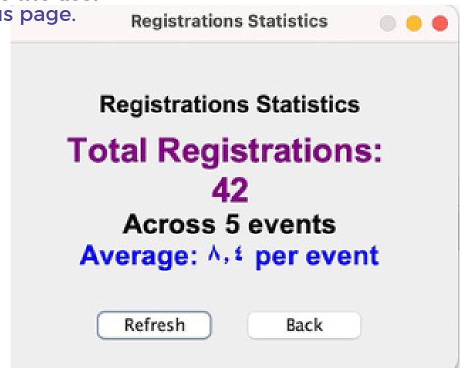
Events Statistics

Total Events: 5

Available: 3

Full: 2

Shows the total number of registrations and the average number of registrations per event.



Registrations Statistics

Total Registrations: 42

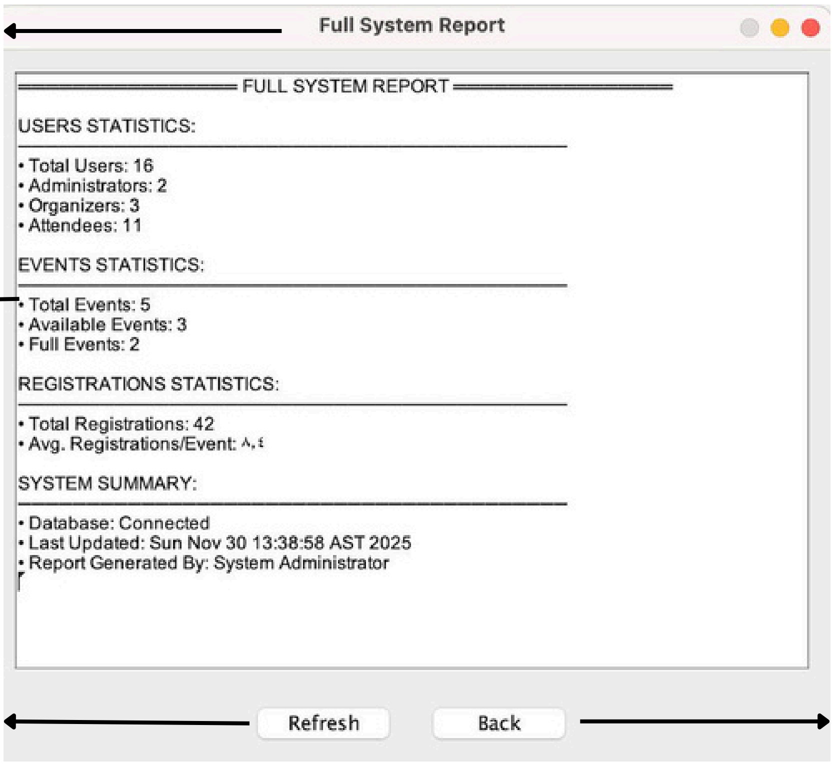
Across 5 events

Average: ٨,٤ per event

GUI Elements

Type: Label
Purpose: Displays the title of the full system report page.

Type: Text Area
Purpose: Shows detailed system statistics for users, events, registrations, and summary.



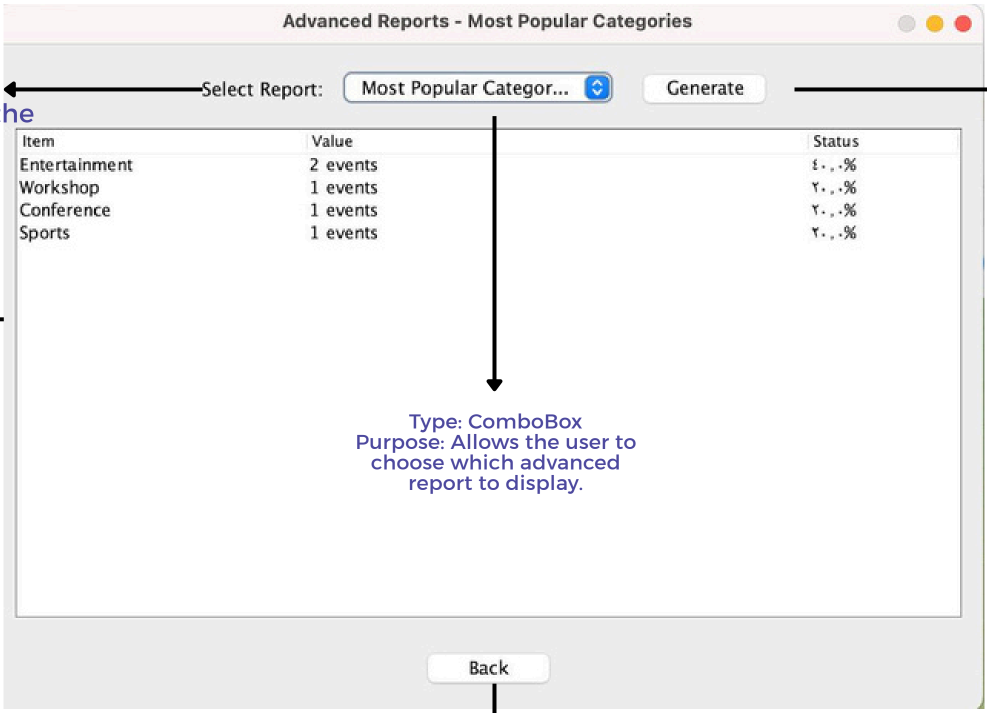
```
═══════════════ FULL SYSTEM REPORT ═══════════════

USERS STATISTICS:
──────────────────────────────────────────
• Total Users: 16
• Administrators: 2
• Organizers: 3
• Attendees: 11

EVENTS STATISTICS:
──────────────────────────────────────────
• Total Events: 5
• Available Events: 3
• Full Events: 2

REGISTRATIONS STATISTICS:
──────────────────────────────────────────
• Total Registrations: 42
• Avg. Registrations/Event: ٨,٤

SYSTEM SUMMARY:
──────────────────────────────────────────
• Database: Connected
• Last Updated: Sun Nov 30 13:38:58 AST 2025
• Report Generated By: System Administrator
```

Type: Button
Purpose: Reloads the report data to show the latest system statistics.

Type: Button
Purpose: Returns to the reports menu.

Type: Label
Purpose: Indicates the ropdown for selecting the report type.

Type: Button
Purpose: Loads and displays the selected report in the table.



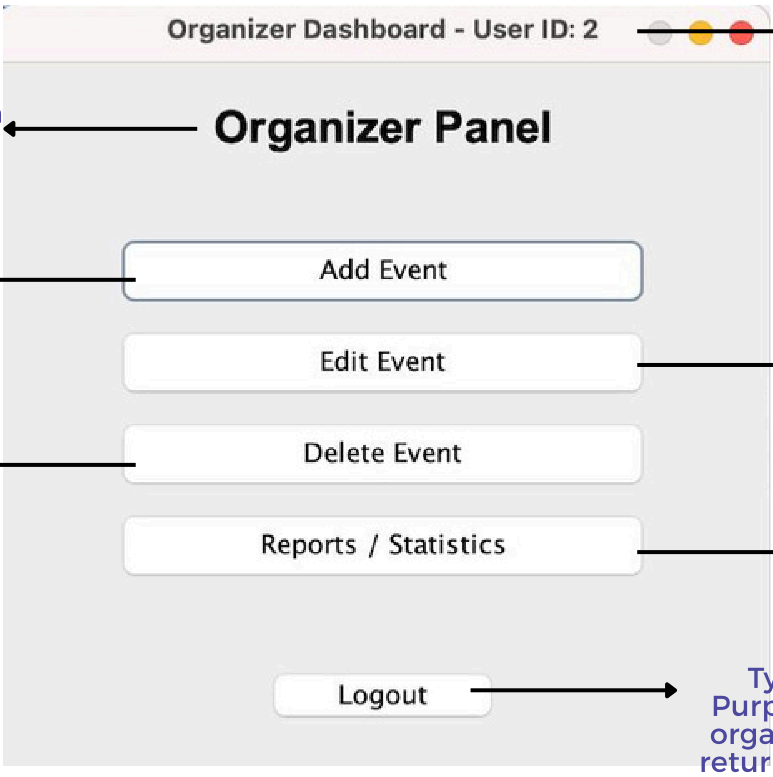
Item	Value	Status
Entertainment	2 events	٤٠,٠%
Workshop	1 events	٢٠,٠%
Conference	1 events	٢٠,٠%
Sports	1 events	٢٠,٠%

Type: Table
Purpose: Displays the report items (category, value, status).

Type: ComboBox
Purpose: Allows the user to choose which advanced report to display.

Type: Button
Purpose: Returns the user to the previous reports menu.

GUI Elements



Type: Label
Purpose: Shows the logged-in organizer's ID.

Type: Label
Purpose: Displays the main title of the organizer dashboard.

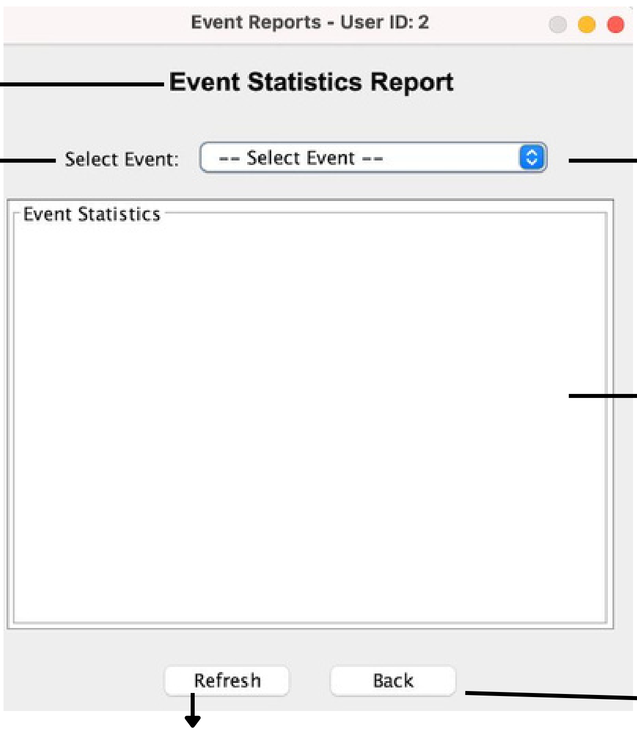
Type: Button
Purpose: Opens the form for creating a new event.

Type: Button
Purpose: Opens the screen for modifying an existing event.

Type: Button
Purpose: Allows the organizer to remove an event from the system.

Type: Button
Purpose: Displays event-related analytics and summary reports.

Type: Button
Purpose: Logs the organizer out and returns to the login screen.



Label – "Event Statistics Report"
Purpose: Displays the title of the report screen.

Label – "Select Event:"
Purpose: Indicates the event selection field.

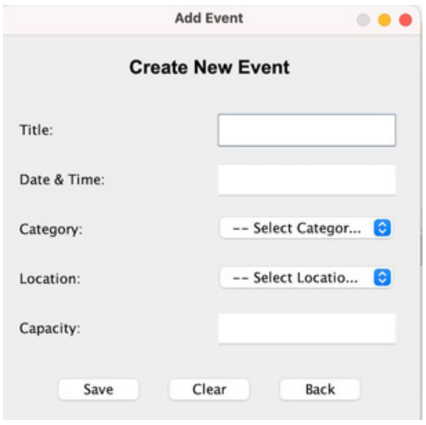
Combo Box – Event Selector
Purpose: Allows the user to choose an event to view its statistics.

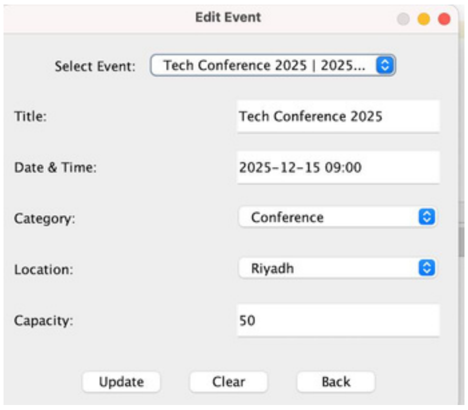
Text Area – Event Statistics Display
Purpose: Shows detailed statistics for the selected event.

Button – "Back"
Purpose: Returns to the previous dashboard.

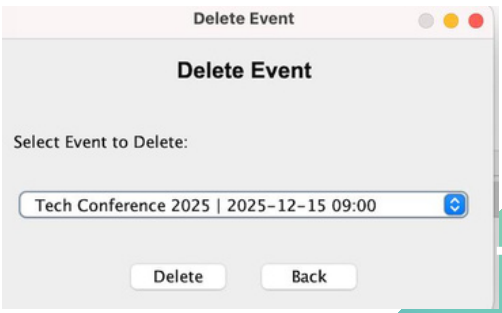
Button – "Refresh"
Purpose: Reloads and updates the statistics.

The Add Event, Edit Event, and Delete Event screens in the Organizer section are identical to the ones in the Admin panel. Since these interfaces were already explained earlier, they will not be repeated here.





Select Event: Tech Conference 2025 | 2025...
Title: Tech Conference 2025
Date & Time: 2025-12-15 09:00
Category: Conference
Location: Riyadh
Capacity: 50



Select Event to Delete: Tech Conference 2025 | 2025-12-15 09:00

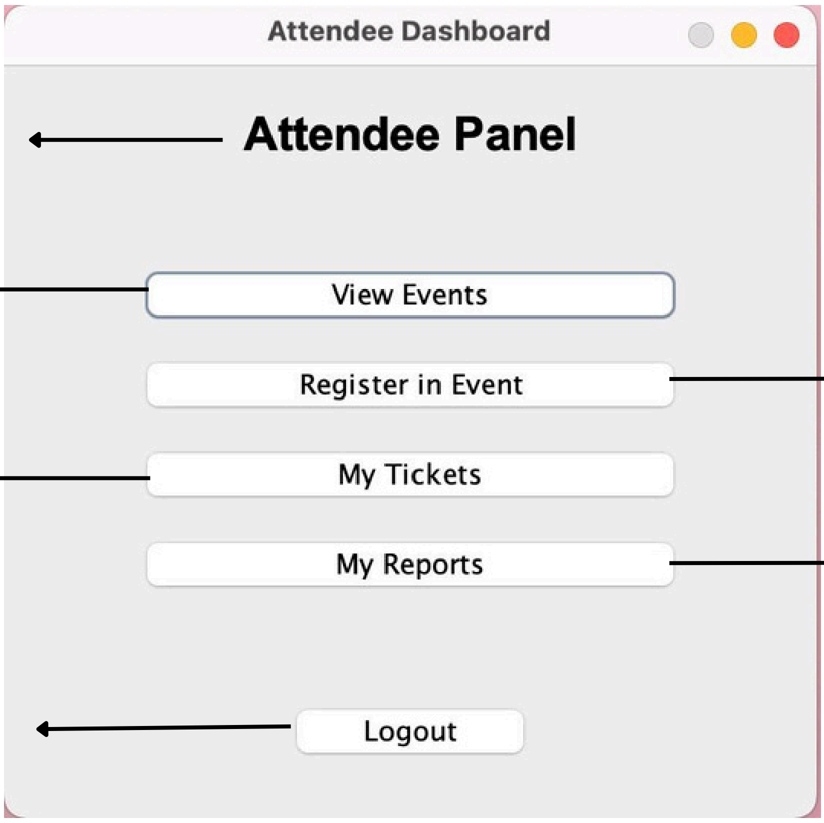
GUI Elements

Type: Label
Purpose: Displays the main title of the attendee dashboard.

Type: Button
Purpose: Opens the page showing available events.

Type: Button
Purpose: Shows all tickets the attendee has registered for.

Type: Button
Purpose: Logs out the attendee and returns to the login page.



Type: Button
Purpose: Allows the attendee to register for a selected event.

Type: Button
Purpose: Displays reports related to the attendee's event participation.

Type: Label
Purpose: Indicates the search input field.

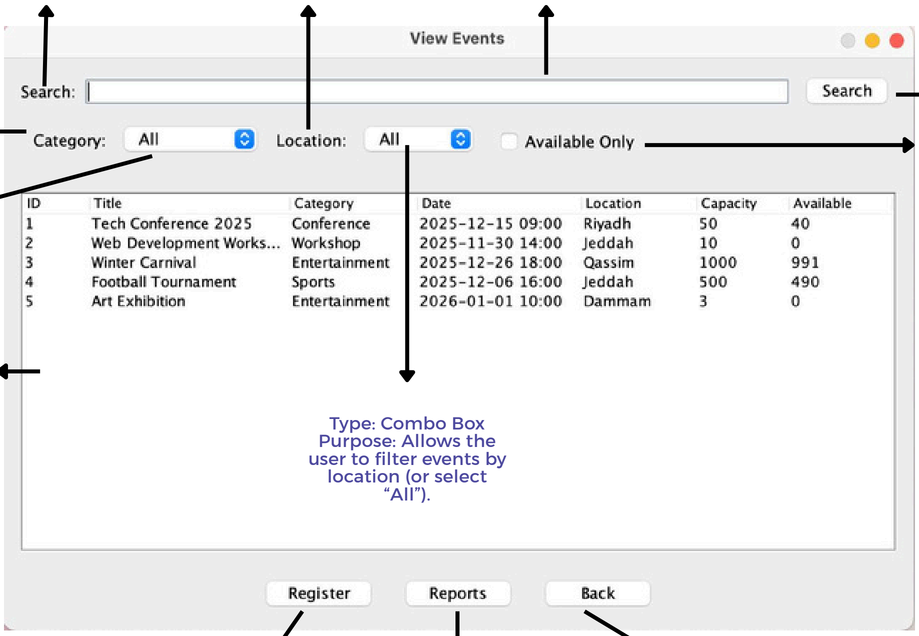
Type: Label
Purpose: Describes the location filter.

Type: Text Field
Purpose: Lets the user type keywords to search for events.

Type: Label
Purpose: Describes the category filter.

Type: Combo Box
Purpose: Allows the user to filter events by category (or select "All").

Type: Table
Purpose: Displays the list of events with ID, title, category, date, location, capacity, and available seats.



ID	Title	Category	Date	Location	Capacity	Available
1	Tech Conference 2025	Conference	2025-12-15 09:00	Riyadh	50	40
2	Web Development Works...	Workshop	2025-11-30 14:00	Jeddah	10	0
3	Winter Carnival	Entertainment	2025-12-26 18:00	Qassim	1000	991
4	Football Tournament	Sports	2025-12-06 16:00	Jeddah	500	490
5	Art Exhibition	Entertainment	2026-01-01 10:00	Dammam	3	0

Type: Combo Box
Purpose: Allows the user to filter events by location (or select "All").

Type: Button
Purpose: Starts the search using the entered keywords.

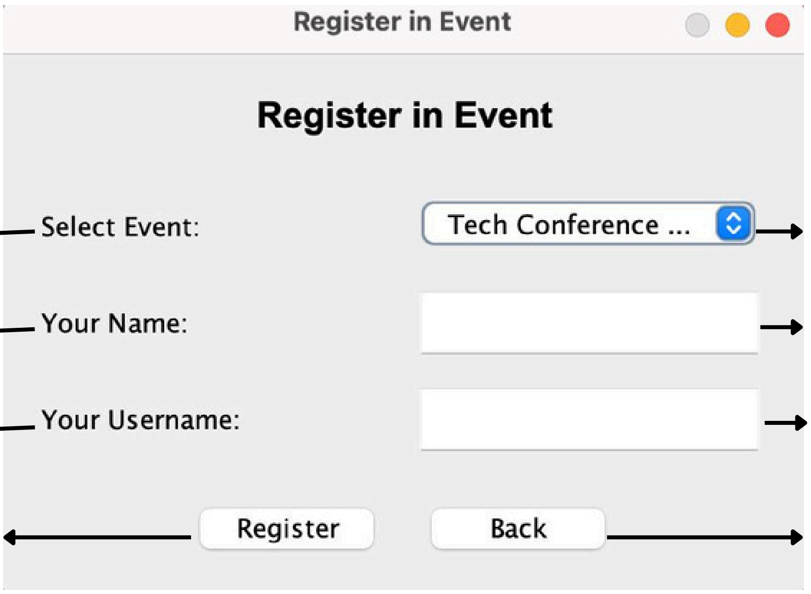
Type: Check Box
Purpose: When checked, shows only events that still have available seats.

Type: Button
Purpose: Opens the registration window for the selected event.

Type: Button
Purpose: Opens the attendee reports window.

Type: Button
Purpose: Returns to the previous attendee main screen.

GUI Elements



Type: Label
Purpose: Shows the event selection label (“Select Event:”).

Type: Combo Box
Purpose: Allows the attendee to choose an event to register in

Type: Label
Purpose: Indicates the field for entering the attendee’s real name.

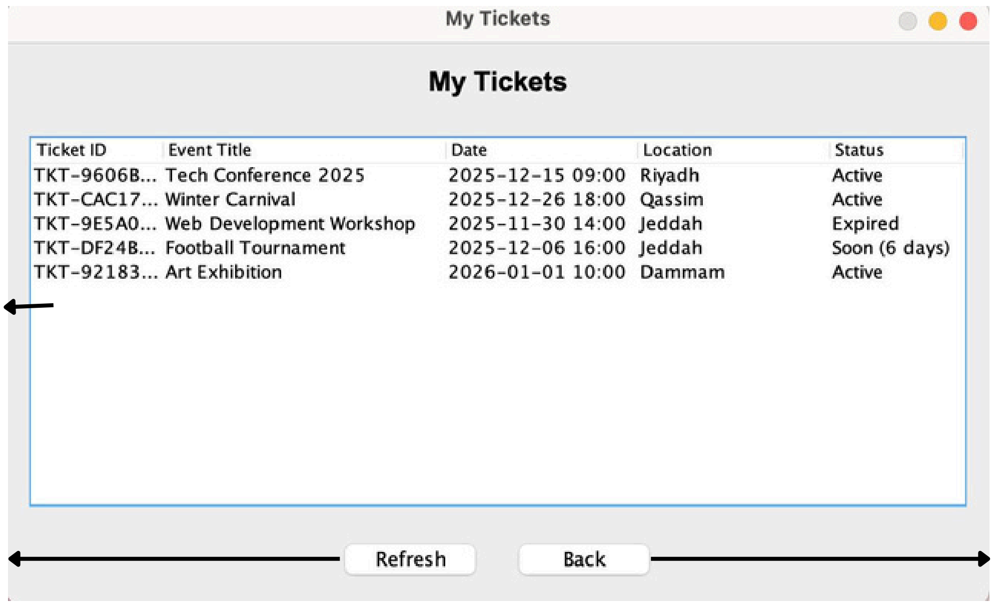
Type: Text Field
Purpose: Allows the attendee to enter their name.

Type: Label
Purpose: Indicates the field for entering the attendee’s username.

Type: Text Field
Purpose: Allows the attendee to enter their username.

Type: Button
Purpose: Submits the registration request for the selected event.

Type: Button
Purpose: Returns to the previous screen.



Ticket ID	Event Title	Date	Location	Status
TKT-9606B...	Tech Conference 2025	2025-12-15 09:00	Riyadh	Active
TKT-CAC17...	Winter Carnival	2025-12-26 18:00	Qassim	Active
TKT-9E5A0...	Web Development Workshop	2025-11-30 14:00	Jeddah	Expired
TKT-DF24B...	Football Tournament	2025-12-06 16:00	Jeddah	Soon (6 days)
TKT-92183...	Art Exhibition	2026-01-01 10:00	Dammam	Active

Type: Table
Purpose: Displays the attendee’s tickets, including ticket ID, event title, date, location, and ticket status.

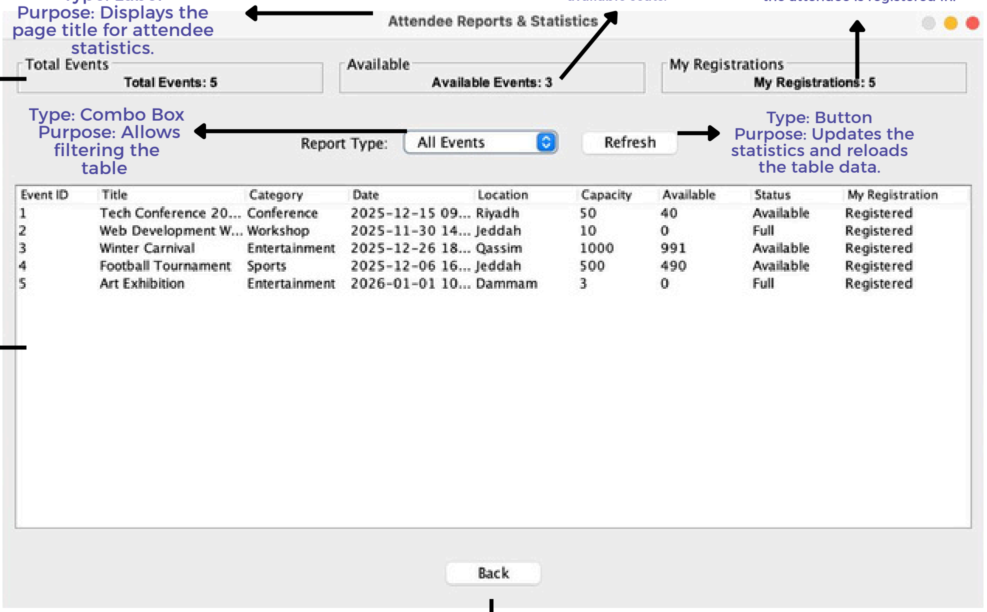
Type: Button
Purpose: Refreshes the table to show the latest ticket updates.

Type: Button
Purpose: Returns to the previous screen.

Type: Label
Purpose: Displays the page title for attendee statistics.

Type: Label
Purpose: Displays how many events still have available seats.

Type: Label
Purpose: Shows how many events the attendee is registered in.



Event ID	Title	Category	Date	Location	Capacity	Available	Status	My Registration
1	Tech Conference 20...	Conference	2025-12-15 09...	Riyadh	50	40	Available	Registered
2	Web Development W...	Workshop	2025-11-30 14...	Jeddah	10	0	Full	Registered
3	Winter Carnival	Entertainment	2025-12-26 18...	Qassim	1000	991	Available	Registered
4	Football Tournament	Sports	2025-12-06 16...	Jeddah	500	490	Available	Registered
5	Art Exhibition	Entertainment	2026-01-01 10...	Dammam	3	0	Full	Registered

Type: Label
Purpose: Shows the total number of events in the system.

Type: Combo Box
Purpose: Allows filtering the table

Type: Button
Purpose: Updates the statistics and reloads the table data.

Type: Table
Purpose: Displays event details such as ID, title, date, location, capacity, availability, and registration status.

Type: Button
Purpose: Returns to the attendee dashboard.

Database Connection

Integrating a database into the application is a fundamental step in developing any robust software. For our Event Registration and Ticket Management System, we used MySQL as the database management system and connected it through the Server tab in NetBeans. This approach streamlined the development process by providing visual database management tools and simplifying debugging.

MySQL Setup:

Before connecting to the database, we ensured the following:

1. MySQL Installation and Operation: The MySQL server was installed and running on our machine.

2. Database Creation: Our database named ticket_management_database was created, containing the required tables.

3. User Configuration: A MySQL user account (root) with appropriate privileges was configured for accessing the database.

Database Connection Class (DBConnection):
A central class named `DBConnection` was created to manage all database connection operations:

· It contains a static getConnection() method that is used to obtain a database connection from any part of the application.

· This Method connects to the MySQL database using the following code:

```
return DriverManager.getConnection(url, username, password);
```

where:

· url = "jdbc:mysql://localhost:3306/ticket_management_database"
· username = "root"
· password = "**********"

Database Tables:

The database was designed to include the following tables:

1. users: Stores user authentication and profile information (username, password, password_salt, role).

2. events: Stores event details and capacity management (title, category, date, location, capacity, seats_available).

3. registrations: Tracks user registrations for events (user_id, event_id).

4. tickets: Manages ticket issuance and tracking (user_id, event_id, ticket_unique_id, ticket_status, qr_code_data).

Exception Handling

1. Admin Access & User Management Exceptions:

1.1 Exception when checking admin access
Occurs inside checkAdminAccess() when calling:

- userDAO.getUserRole(userId)

If an error happens, message shown:

- "Error checking permissions: " + e.getMessage()

1.2 Access denied (handled scenario)
If user role is not “admin”:

- System displays "Access Denied"
- User is returned to login screen
- (Not an exception.)

1.3 Exception while retrieving user information
Triggered during:

- userDAO.getUserByUsername(username)

Displayed as:

- "Error deleting user: " + ex.getMessage()

1.4 Exception while checking admin count
Occurs during:

- userDAO.getAdminCount()

Handled inside the same exception block.

1.5 Exception when deleting a user
Triggered during:

- userDAO.deleteUser(user.getId())

Shown in the deletion error message.

1.6 User not found (handled scenario)
If getUserByUsername() returns null:

- "User '___' not found!"

1.7 Missing username input (handled scenario)
If text field is empty:

- "Please enter a username"

Exception Handling

2. Data Loading Exceptions

2.1 Exception loading users
In loadUsersData() when calling:
- userDAO.getAllUsers()

Error shown:
- "Error loading users: " + ex.getMessage()

2.2 Exception loading reports
Occurs in loadReports() during:
- eventDAO.getAllEvents()
- ticketDAO.hasTicket(currentUserId, event.getEventId())

Shown as:
- "Error loading reports: " + ex.getMessage()

2.3 Exception loading events
Occurs when calling:
- eventDAO.getAllEvents()

Shown as:
- "Error loading events: " + ex.getMessage()

Exception Handling

3. Registration Process Exceptions

Exceptions may occur during:

- userDAO.getUserByUsername(username)
- eventDAO.getEventById(...)
- ticketDAO.hasTicket(...)
- registrationDAO.register(...)
- eventDAO.updateSeatsAvailable(...)
- ticketDAO.createTicket(...)

Displayed as:

- "Registration failed: " + ex.getMessage()

4. Ticket Loading & Event Parsing Exceptions

4.1 Exception loading tickets
Occurs in loadTicketsFromDB() during:

- ticketDAO.getTicketsForUser(currentUserId)

Shown as:

- "Error loading tickets: " + ex.getMessage()

4.2 Exception parsing event date
Occurs in getTicketStatus() when:

- sdf.parse(eventDate)

If parsing fails:

- Status returned as "Unknown"

4.3 Exception loading events (duplicate section)
Occurs when calling:

- eventDAO.getAllEvents()

Shown as:

- "Error loading events: " + ex.getMessage()

Exception Handling

5. Login & Authentication Exceptions

5.1 SQL/database exceptions in login()
Triggered when calling:
- DBConnection.getConnection()
- ps.executeQuery()

Method signature includes throws Exception → exception propagates upward.

5.2 Exception verifying password
Occurs during:
- PasswordUtility.verifyPassword(...)

Not caught → propagated upward.

5.3 Exception from DAO calls
These may throw exceptions:
- userDAO.getUserByUsername(username)
- userDAO.verifyUserPassword(username, password)

5.4 Exception during handleLogin()
Occurs when calling:
- authService.login(username, password)

Shown as:
- "An unexpected error occurred.\n" + ex.getMessage()

6. Password Handling Exceptions

6.1 Exception hashing password
Occurs in:
- MessageDigest.getInstance("SHA-256")

If algorithm missing:
- NoSuchAlgorithmException → wrapped in RuntimeException.

6.2 Exception resetting password
During:
- userDAO.getUserByUsername(username)
- userDAO.updatePassword(user.getId(), newpass)

Shown as:
- "Error updating password: " + ex.getMessage()

7. User Creation Exceptions

7.1 Exception creating a new user
Occurs in:
- userDAO.addUser(newUser)

Shown as:
- "Error creating user: " + ex.getMessage()

Exception Handling

6. Password Handling Exceptions

6.1 Exception hashing password
Occurs in:
- MessageDigest.getInstance("SHA-256")

If algorithm missing:
- NoSuchAlgorithmException → wrapped in RuntimeException.

6.2 Exception resetting password
During:
- userDAO.getUserByUsername(username)
- userDAO.updatePassword(user.getId(), newpass)

Shown as:
- "Error updating password: " + ex.getMessage()

7. User Creation Exceptions

7.1 Exception creating a new user
Occurs in:
- userDAO.addUser(newUser)

Shown as:
- "Error creating user: " + ex.getMessage()

8. DAO-Level SQL Exceptions (General)

All DAO methods use:
- DBConnection.getConnection()
- ps.executeUpdate()
- ps.executeQuery()

And all declare:
- throws Exception

Thus, any SQL error is thrown to caller.

Exception Handling

9. Event DAO Exceptions

9.1 Exception inserting an event
Occurs in:
- addEvent(Event event)

9.2 Exception deleting an event
Occurs in:
- deleteEvent(int eventId)

9.3 Exception updating an event
Occurs in:
- updateEvent(Event event)

9.4 Exception updating available seats
Occurs in:
- updateSeatsAvailable(eventId, changeBy)

9.5 Exception checking available seats
Occurs in:
- hasAvailableSeats(eventId)

9.6 Exception fetching event by ID
Occurs in:
- getEventById(eventId)

9.7 Exception fetching all events
Occurs in:
- getAllEvents()

9.8 Exception counting events
Occurs in:
- getTotalEventsCount()
- getAvailableEventsCount()
- getFullEventsCount()

Exception Handling

10. Registration DAO Exceptions

10.1 Exception during registration insertion
Occurs in:
- register(userId, eventId)

10.2 Exception counting registrations
Occurs in:
- countRegistrations(eventId)

10.3 Exception fetching registered users
Occurs in:
- getRegisteredUsersForEvent(eventId)

10.4 Exception counting all registrations
Occurs in:
- getTotalRegistrationsCount()

11. Ticket DAO Exceptions

11.1 Exception creating a ticket
Occurs in:
- createTicket(userId, eventId)

11.2 Exception retrieving tickets
Occurs in:
- getTicketsForUser(userId)

11.3 Exception checking if user already has ticket
Occurs in:
- hasTicket(userId, eventId)

11.4 Exception retrieving detailed ticket information
Occurs in:
- getTicketDetails(ticketUniqueId)

Exception Handling

12. User DAO Exceptions

12.1 Exception adding a new user
Occurs in:
- addUser(User user)

12.2 Exception deleting a user
Occurs in:
- deleteUser(int userId)

12.3 Exception updating a user's password
Occurs in:
- updatePassword(int userId, newPassword)

12.4 Exception verifying password
Occurs in:
- verifyUserPassword(username, password)

12.5 Exception retrieving one user
Occurs in:
- getUserByUsername(username)

12.6 Exception retrieving all users
Occurs in:
- getAllUsers()

12.7 Exception retrieving user role
Occurs in:
- getUserRole(userId)

12.8 Exception counting users
Occurs in:
- getTotalUsersCount()

12.9 Exception counting admins
Occurs in:
- getAdminCount()

12.10 Exception counting by role
Occurs in:
- getUsersCountByRole(String role)

Exception Handling

13. Database Connection Exceptions

13.1 Exception loading JDBC driver
Occurs when:
- Class.forName("com.mysql.cj.jdbc.Driver")

Throws:
- ClassNotFoundException

13.2 Exception establishing connection
Occurs when:
- DriverManager.getConnection(url, username, password)

Throws:
- SQLException

14. Event Form (UI) Exceptions

14.1 Number format exception
Occurs when parsing:
- Integer.parseInt(capacityStr)

14.2 Date parse exception
Occurs when:
- sdf.parse(dateTime)

Shown as:
- "Invalid date format. Use: yyyy-MM-dd HH:mm"

14.3 Exception saving event
Occurs when calling:
- eventDAO.addEvent(event)

Shown as:
- "Error saving event: " + ex.getMessage()

14.4 Exception deleting event
Occurs in:
- handleDelete() → eventDAO.deleteEvent()

14.5 Exception updating event
Occurs when calling:
- eventDAO.updateEvent(updatedEvent)

Shown as:
- "Error updating event: " + ex.getMessage()

Exception Handling

15. Organizer Access Exceptions

15.1 Exception checking organizer access
Occurs during:
- userDAO.getUserRole(userId)

Shown as:
- "Error checking permissions: " + e.getMessage()

16. Report Generation Exceptions

16.1 Error loading events list
Occurs during:
- eventDAO.getAllEvents()

Shown as:
- "Error loading events: " + ex.getMessage()"

16.2 Error loading event statistics
Occurs during calls to:
- registrationDAO.countRegistrations(...)

Shown as:
- "Error loading statistics: " + ex.getMessage()"

16.3 Errors in generating full reports
Occurs during calls to:
- userDAO.getTotalUsersCount()
- userDAO.getUsersCountByRole(...)
- eventDAO.getTotalEventsCount()
- eventDAO.getAvailableEventsCount()
- eventDAO.getFullEventsCount()
- registrationDAO.getTotalRegistrationsCount()

Shown as:
- "Error loading report data: " + message

16.4 Errors loading specific report types
Examples:
- Most popular categories
- Capacity utilization
- Most registered events

All caught in generateReport() as:
- "Error generating report: " + ex.getMessage()"

Exception Handling

17. Popular Categories, Capacity & Registrations Reports

17.1 Error generating popular categories
Possible causes:
- DB failure
- Null category
- Empty event list
- HashMap issues

17.2 Error calculating capacity
Occurs when:
- event capacity or seats available invalid

Division guarded by:
- (totalCapacity > 0)

17.3 Error generating most registered events
Possible causes:
- Invalid eventId
- Registration table read failure
- Zero-capacity events

18. Ticket ID & QR Code Exceptions

18.1 Error generating ticket ID
Occurs during:
- UUID.randomUUID().toString().substring(0, 8)

Very rare.
18.2 Error generating QR code data
Occurs during:
- String.format(...)

Possible causes:
- Null ticketId
- Null eventTitle
- Null eventDate

Exception Handling

19. Transaction Exceptions (Registration Process)

19.1 Error establishing DB connection
Occurs in:
- DBConnection.getConnection()

Leads to rollback.
19.2 Error setting autocommit false
Occurs in:
- conn.setAutoCommit(false)

Triggers catch → rollback.
19.3 No available seats (explicit exception)
If eventDAO.hasAvailableSeats(eventId) returns false:
- Throws: new Exception("No available seats")
- → rollback

19.4 Error creating ticket
Occurs in:
- ticketDAO.createTicket(...)

Triggers rollback.
19.5 Error registering user
Occurs in:
- registrationDAO.register(...)

Triggers rollback.
19.6 Error updating seats
Occurs when:
- eventDAO.updateSeatsAvailable(...) returns false

Throws:
- "Failed to update seats"
  rollback

19.7 Commit failure
Occurs during:
- conn.commit()

Triggers rollback.
19.8 Rollback failure
Occurs in:
- conn.rollback()

Printed but not thrown.
19.9 Error restoring autocommit / closing connection
Occurs in:
- conn.setAutoCommit(true)
- conn.close()

Printed but not thrown.

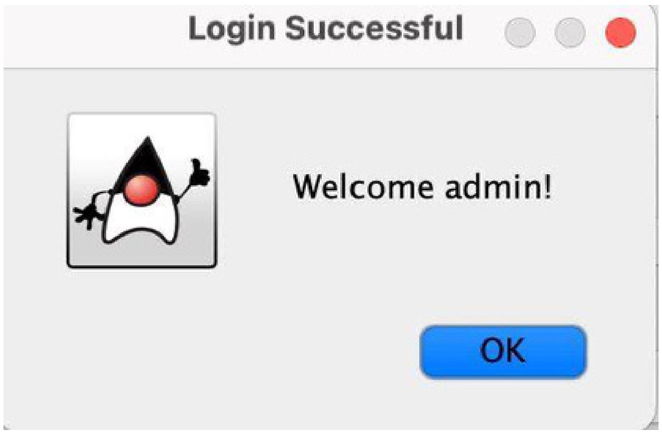
Testing and Validation

To validate the Ticket Management System, we performed manual end-to-end testing for all main workflows: authentication, user management, event management, registration, and reporting. For each feature, we executed both valid and invalid scenarios and captured screenshots as evidence. The following test cases summarize how the system was tested and how we ensured that it works correctly for different users (Admin, Organizer, and Attendee).

1) Log in

Test Case 1 : Successful Login

We entered a valid username and password for an existing user. The system authenticated the credentials and redirected the user to the appropriate dashboard (Admin Panel, Organizer Panel, or Attendee Panel). This confirms that the login process works correctly with valid data.



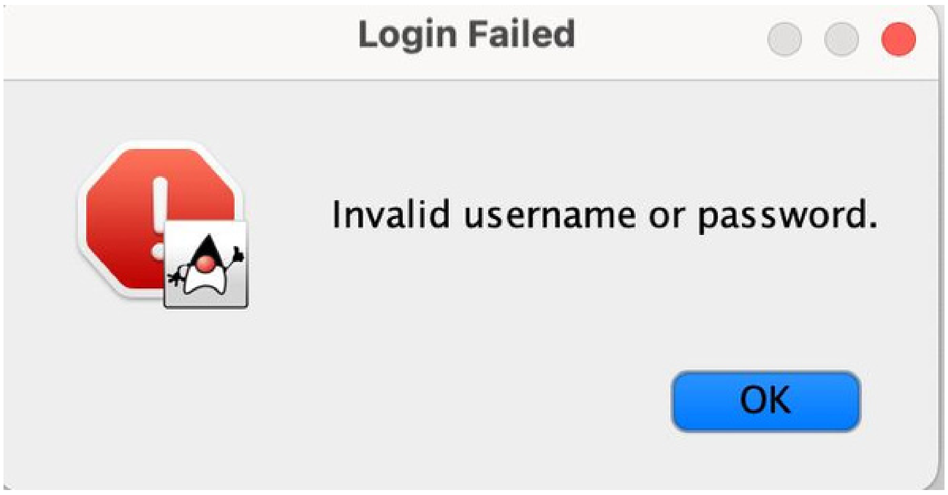
Welcome admin!

Successful login with valid credentials, leading to the main dashboard.

Testing and Validation

Test Case 2 : Failed Login with Invalid Credentials

We intentionally entered an incorrect password for a valid username. The system rejected the login attempt and displayed an error message instead of granting access. This verifies that invalid credentials are handled correctly and that unauthorized users cannot enter the system.



Invalid username or password.

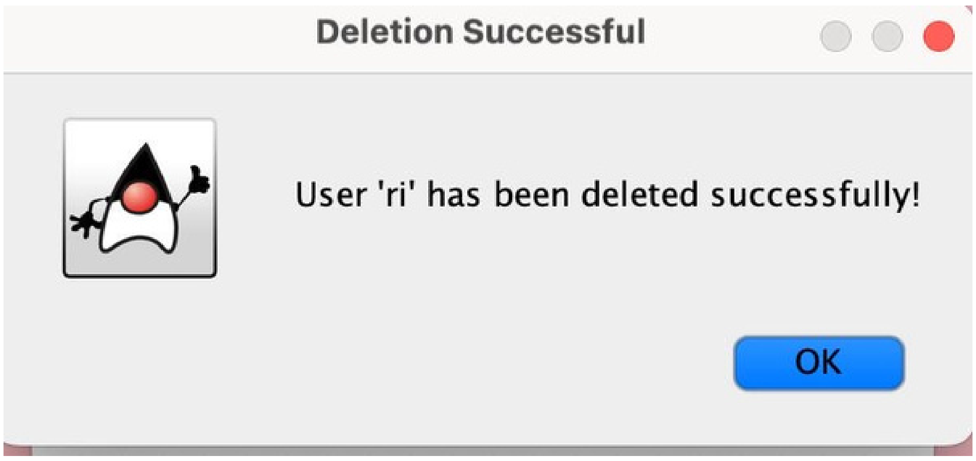
Failed login attempt with wrong password, showing an error message.

Testing and Validation

2) Delete User

Test Case 1 : Successful Delete User

We opened the Delete User screen from the Admin Panel, entered a valid username of an existing user, and clicked Delete. The system removed the user from the database, and the user no longer appeared in the user list or system reports. This verifies that the delete operation works correctly when the username exists.


Deletion Successful

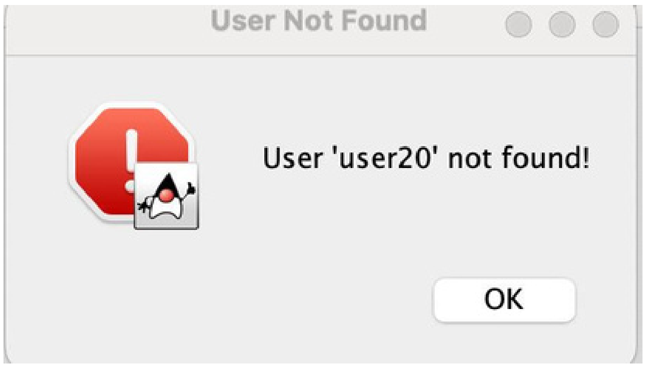
User 'ri' has been deleted successfully!

Admin successfully deleted an existing user.

Testing and Validation

Test Case 2 : Deleting a User That Does Not Exist (Failure)

To test validation, we attempted to delete a username that does not exist in the system. After clicking Delete, the system displayed an error message indicating that the user could not be found. No changes were made to the database. This confirms that invalid delete operations are safely handled.

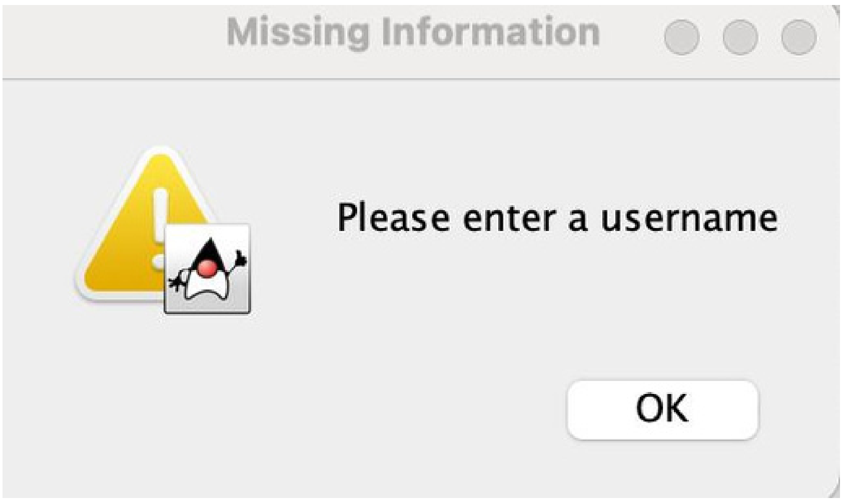

User Not Found

User 'user20' not found!

Failed delete attempt with a non-existing username, showing an error message.

Test Case 3 : Attempting to Delete a User Without Entering a Username

In this scenario, we tested how the system handles a missing input. We opened the Delete User form, left the username field empty, and clicked the Delete button. The system detected the missing information and displayed a warning message: “Please enter a username”. No operation was performed, and no changes were made to the database.



Please enter a username

Error message displayed when attempting to delete a user with an empty username.

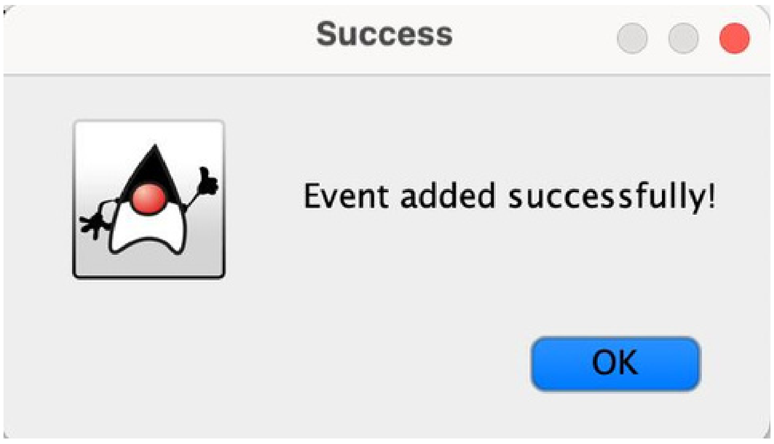
Testing and Validation

3) Add Event

Test Case 1 :

Successfully Adding a New Event (Success)

We opened the Add Event screen and entered all required fields correctly (Title, Date/Time, Category, Location, and Capacity). After clicking Save, the system added the event to the events list successfully and displayed a confirmation message.
This confirms that the add operation works properly with valid input.



Success

Event added successfully!

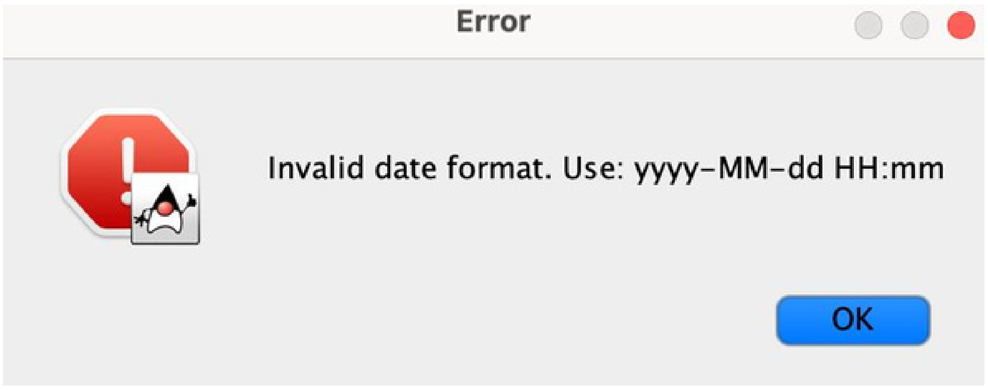
Event is saved and appears in the events table.

Testing and Validation

Test Case 2:

Adding an Event with Invalid Date Format (Failure)

In this scenario, we tested how the system handles an incorrectly formatted date when adding a new event.
We entered an invalid date format (e.g., “2025/12/15 9 AM” or “15-12-2025 09:00”) instead of the required format “YYYY-MM-dd HH:mm”, and then clicked Save.


Error

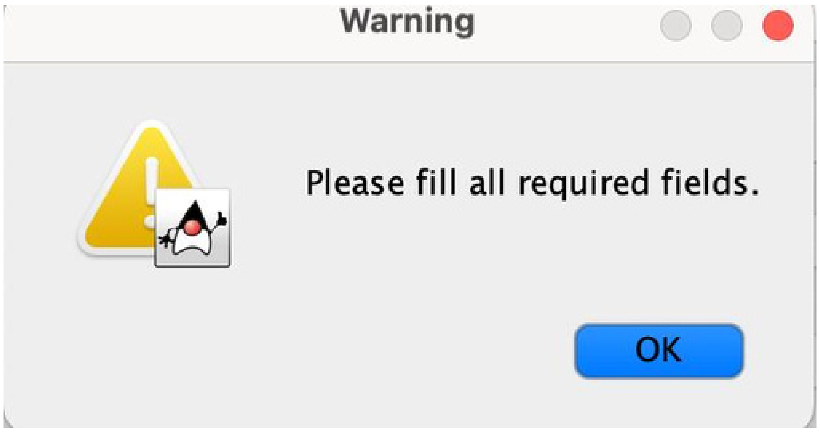
Invalid date format. Use: yyyy-MM-dd HH:mm

Error displayed for invalid date format during event creation.

Test Case 3:

Attempting to Add an Event with Missing Fields (Failure)

To test validation, we left one or more fields empty (e.g., Title or Capacity) and clicked Save.
The system detected the missing information and displayed a warning message asking the user to complete all required fields.
No event was added.


Warning

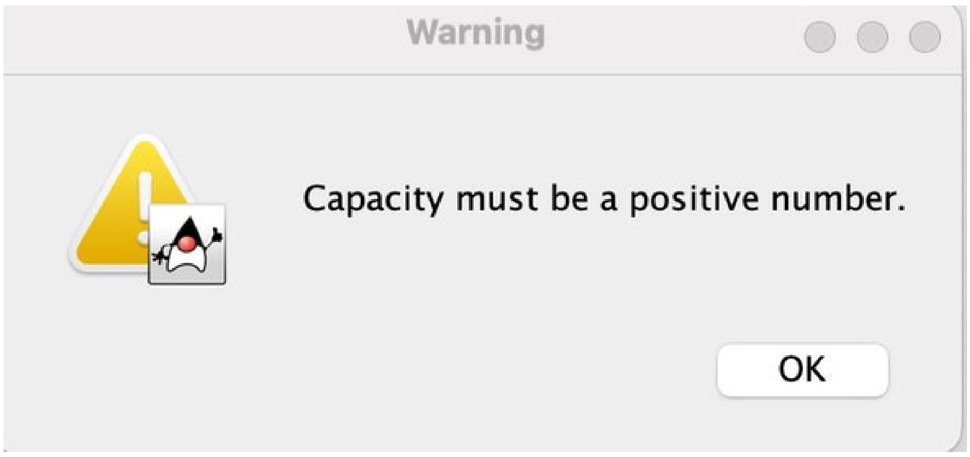
Please fill all required fields.

Warning message appears. No changes made to the database.

Testing and Validation

Test Case 4:
Attempting to Add an Event With Invalid Capacity (Negative Number)

In this scenario, we tested how the system validates event capacity input.
We opened the Add Event form, filled the required fields, and entered a negative capacity value (e.g., –10).
When we clicked the Save button, the system detected the invalid input and displayed a warning message:
“Capacity must be a positive number.”
No event was created, and no data was saved to the database, confirming that the input validation works correctly


Warning

Capacity must be a positive number.

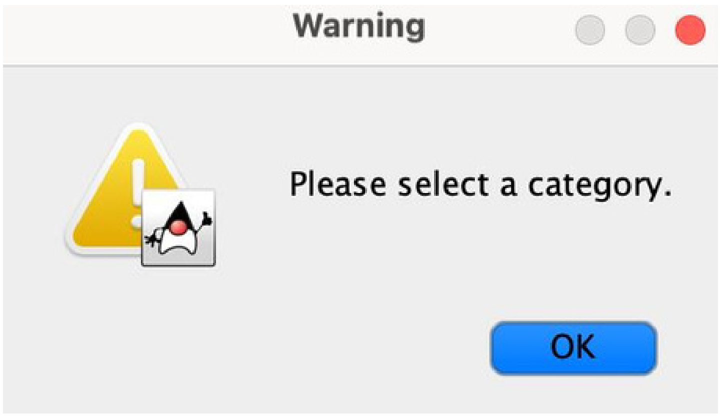
Error message displayed when entering a negative event capacity.

Testing and Validation

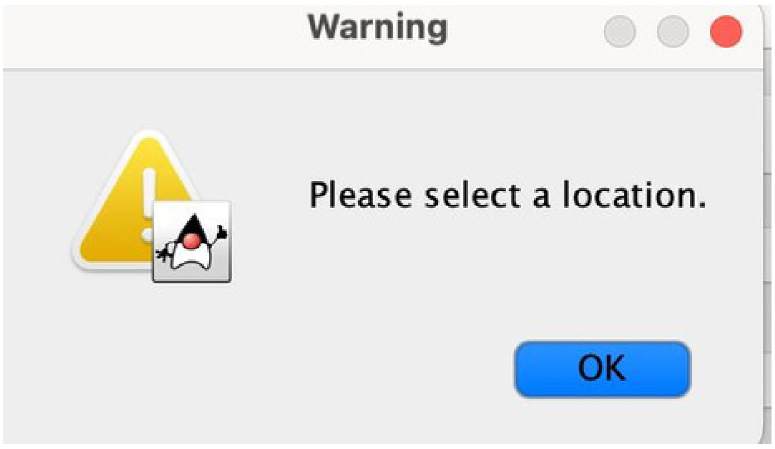
Test Case 5 :

Attempting to Add an Event Without Selecting Category or Location (Failure)

In this test case, we attempted to add a new event while leaving the Category and Location combo boxes unselected. After filling the other fields (Title, Date & Time, Capacity) and clicking the Save button, the system detected that mandatory dropdown fields were missing.


Warning

Please select a category.


Warning

Please select a location.

The system displays a warning message when Category or Location is not selected, ensuring required fields are filled before adding an event.

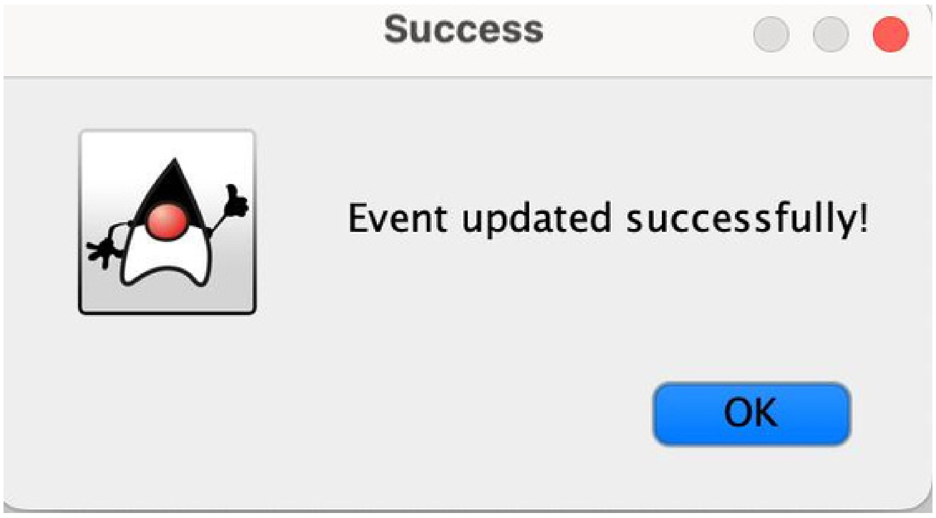
4) Edit Event

Test Case 1 :

Successfully Editing an Existing Event (Success)

We opened the Edit Event screen, selected an existing event from the dropdown, updated its information (e.g., new date, new capacity), and clicked Update.

The system saved the changes and updated the event details in the events table.



Event updated successfully!

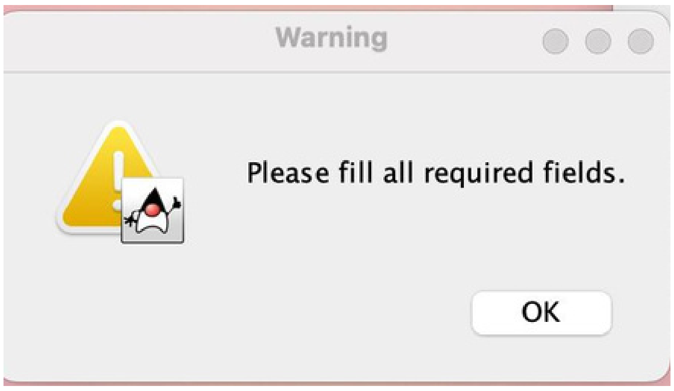
Event details updated.

Test Case 2 :

Editing an Event with Missing/Invalid Data (Failure)

We selected an event but left required fields empty (e.g., deleting the title), or entered invalid capacity.

After clicking Update, the system displayed a validation error and prevented the update.



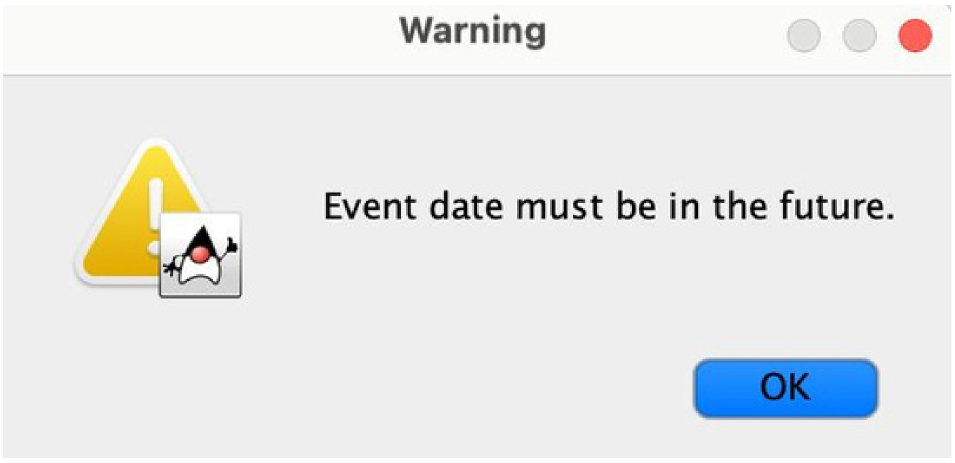
Please fill all required fields.

Validation error. Event not updated.

Test Case 3:

Editing an Event with a Past Date (Failure)

While editing an existing event, a past date was entered in the Date & Time field.
When clicking Update, the system detected the invalid date and displayed a warning message stating that the event date must be in the future.
No changes were applied to the event.



Event date must be in the future.

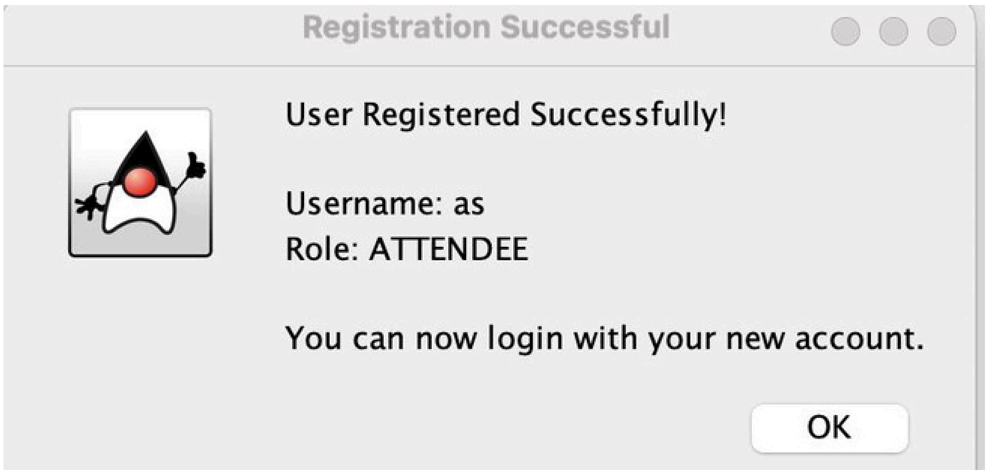
Warning shown — the system prevented saving an event with an invalid (past) date.

Testing and Validation

4) Sign up

Test Case 1:
Successful User Registration

We tested the Sign-Up process by entering valid information for a new user and submitting the registration form.
The system successfully created the account and displayed a confirmation message showing the username and assigned role.



User Registered Successfully!

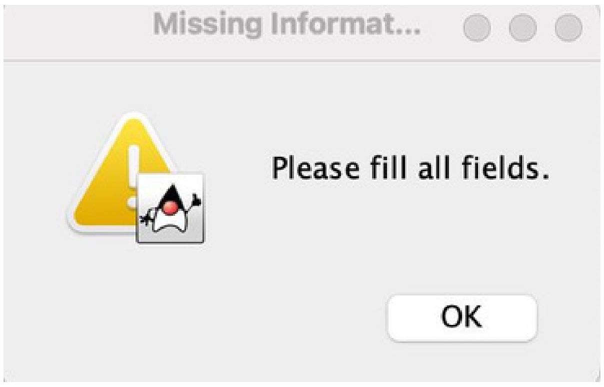
Username: as
Role: ATTENDEE

You can now login with your new account.

User account created successfully, and the system informed the user that they can now log in using their new credentials

Test Case 2:
Attempting to Register with Missing Fields (Failure)

While completing the Sign-Up form, we intentionally left one or more required fields empty and clicked the Register button.
The system detected the missing information and displayed a warning message instructing the user to fill all fields.



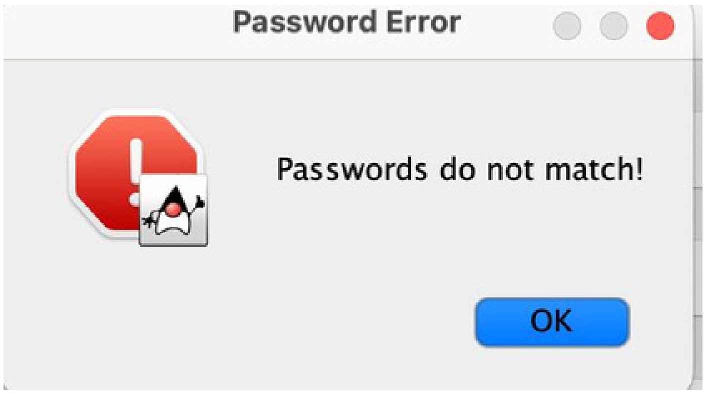
Please fill all fields.

Registration was not completed, and no user account was created until all fields were properly filled.

Test Case 3:

Password and Confirm Password Do Not Match (Failure)

During the Sign-Up process, we entered two different values in the Password and Confirm Password fields. When clicking the Register button, the system detected the mismatch and displayed an error message indicating that the passwords do not match.



Passwords do not match!

Registration was not completed until both password fields matched correctly.

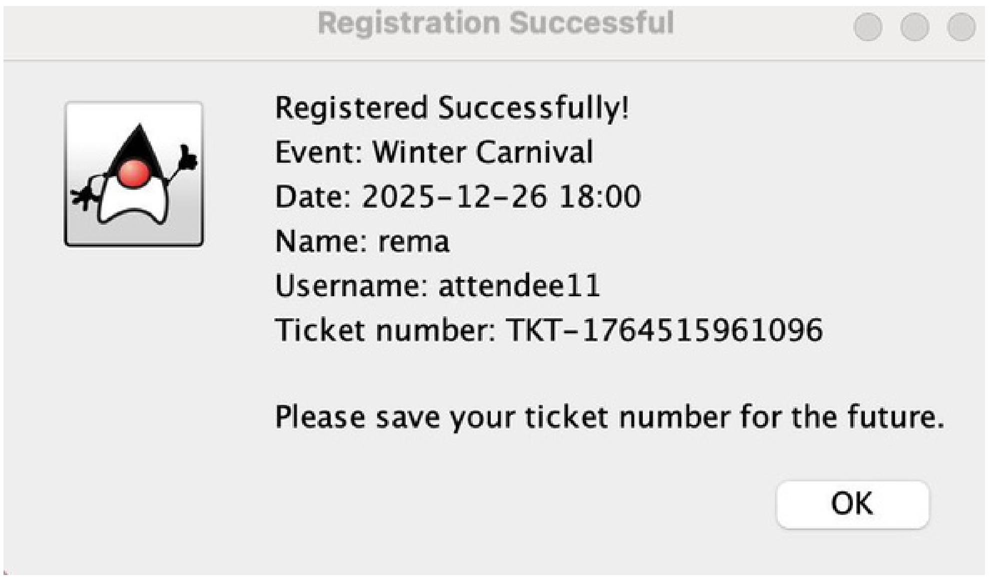
5) Register in event

Test Case 1:

Successful Event Registration

As an attendee, we registered for an event by selecting Winter Carnival and clicking Register.

The system processed the request successfully and displayed a confirmation message showing full event details, including the event name, date, attendee name, username, and a unique ticket number.



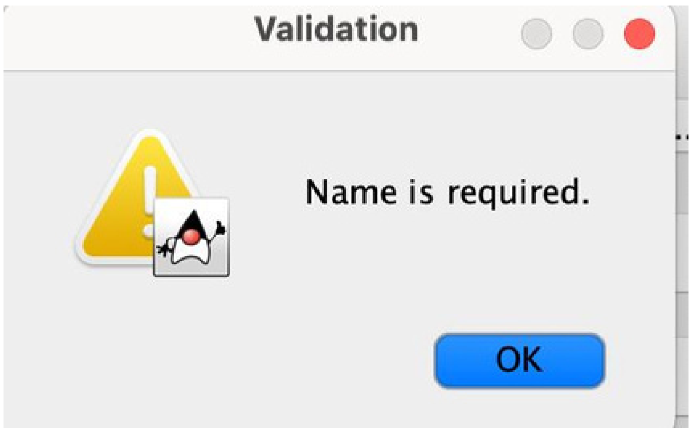
```
Registered Successfully!
Event: Winter Carnival
Date: 2025-12-26 18:00
Name: rema
Username: attendee11
Ticket number: TKT-1764515961096

Please save your ticket number for the future.
```

Registration completed successfully, and the attendee received a ticket number for future reference.

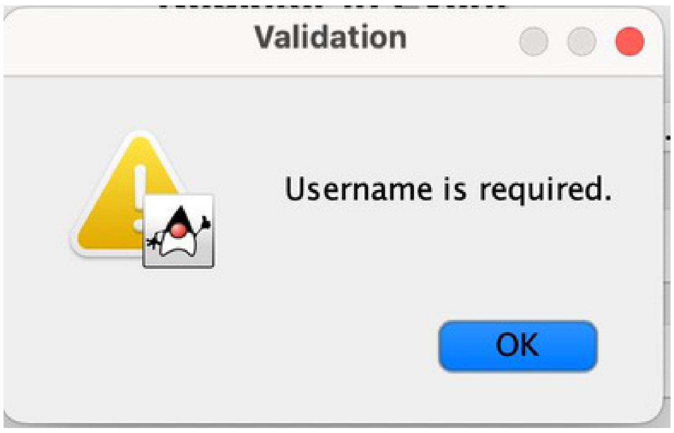
Test Case 2:

Missing Name or Username When Adding an Event (Failure)

While attempting to add a new event, the attendee left one of the required text fields empty (either the Name field or the Username field). After clicking the Add Event button, the system detected the missing information and displayed a validation warning message ("Name is required." or "Username is required.")


Validation

Name is required.


Validation

Username is required.

The event was not added, and the system ensured that both fields must be filled before processing the request.

Test Case 3 :

Attempting to Register in an Event with a Non-Existing Username

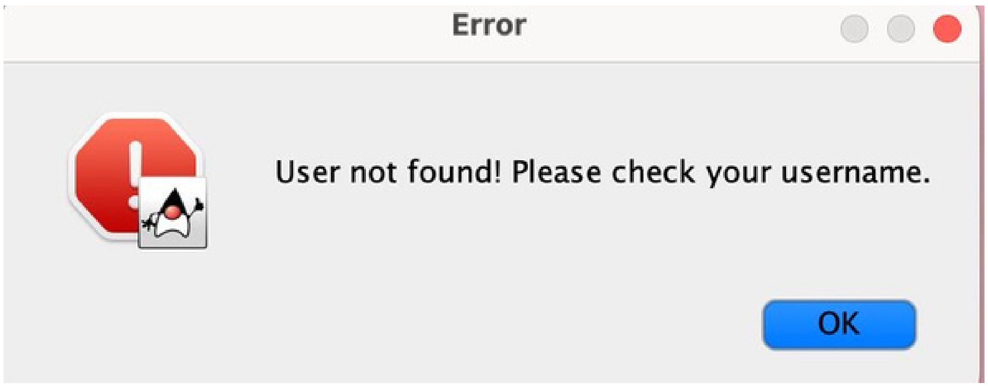
In this scenario, we tested how the system responds when a user attempts to register for an event using a username that does not exist in the database.

We opened the Register in Event form, selected an event, entered a valid name, and typed an invalid/non-existing username in the “Your Username” field.

When we clicked the Register button, the system checked the database and could not find the entered username. As expected, the system displayed an error message:

“User not found! Please check your username.”

The registration process was stopped, and no changes were made to the event or the database.

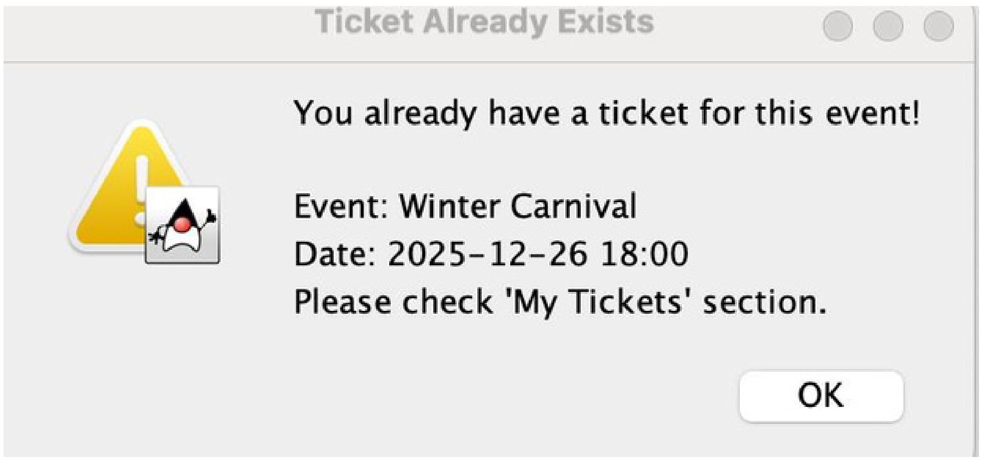

Error

User not found! Please check your username.

Error message displayed when attempting to register using a non-existing username.

Test Case 4 :
Attempting to Register for an Event Already Registered (Failure)

The attendee attempted to register for an event they had already registered for earlier. After clicking the Register button, the system detected an existing ticket and displayed a warning message informing the user that they already have a ticket for this event.



You already have a ticket for this event!

Event: Winter Carnival
Date: 2025-12-26 18:00
Please check 'My Tickets' section.

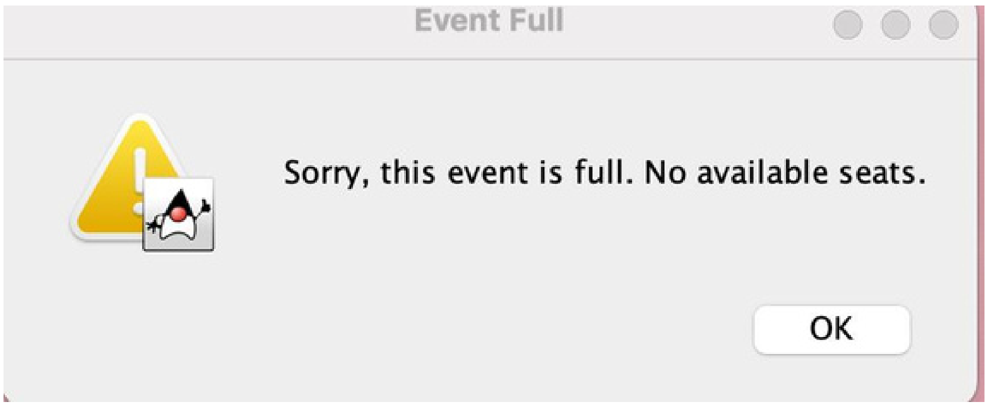
No new ticket was created. The system prevented duplicate registration and directed the user to check the My Tickets section.

Testing and Validation

Test Case 5 :

Attempting to Register for a Full Event (Failure)

The attendee tried to register for an event that had already reached its maximum capacity. After clicking the Register button, the system detected that there were no available seats and displayed a warning message informing the user that the event is full.


Event Full

Sorry, this event is full. No available seats.

The registration was blocked, and no ticket was created. This confirms that the system correctly prevents registration when event capacity is reached.