Restaurant Management System

GithubURL

https://github.com/LiShangcheng/oop_final_project_2025.git

Leo Li (sl10449)

Doris Zhu(ez2334)

Date of Submission: August 14, 2025

Table of Work

(Please write x in the boxes to mention what each student achieved in this project)

	Leo Li	Doris Zhu	
Project Description	X	X	
Uses Cases Diagram(s) and discription	X	X	
Sequence Diagrams	X	X	
Class diagram(s)	X	X	
Implementation	X	X	
Conclusion	X	X	

System Analysis

This system is a graphical restaurant table management application designed to streamline front-of-house operations such as managing table availability, waitlists, and party assignments. The system simulates a real-world dining environment where waitstaff and customers interact through intuitive GUI screens.

General Description, Goals and Benefits

Main Goals:

- To allow customers to join a queue and check their status.
- To let staff view queue and table statuses, assign or release tables, and monitor party progress.

Key Benefits:

- Improved efficiency in managing dining flow.
- User-friendly interfaces for both customers and staff.
- Encourages fair table assignment based on queue order and party size.

Special Requirements

Performance:

- Handles multiple user interactions without significant delay.

Interfaces:

- MainGUI.java acts as the central hub for all user and admin actions.
- JoinQueueGUI: For customers joining the waitlist.
- QueueStatusGUI and CheckStatusGUI: For viewing queue and party status.
- WaiterDashBoardGUI: For waiters to view and manage tables.
- ReleaseTableGUI: For releasing tables post-dining.

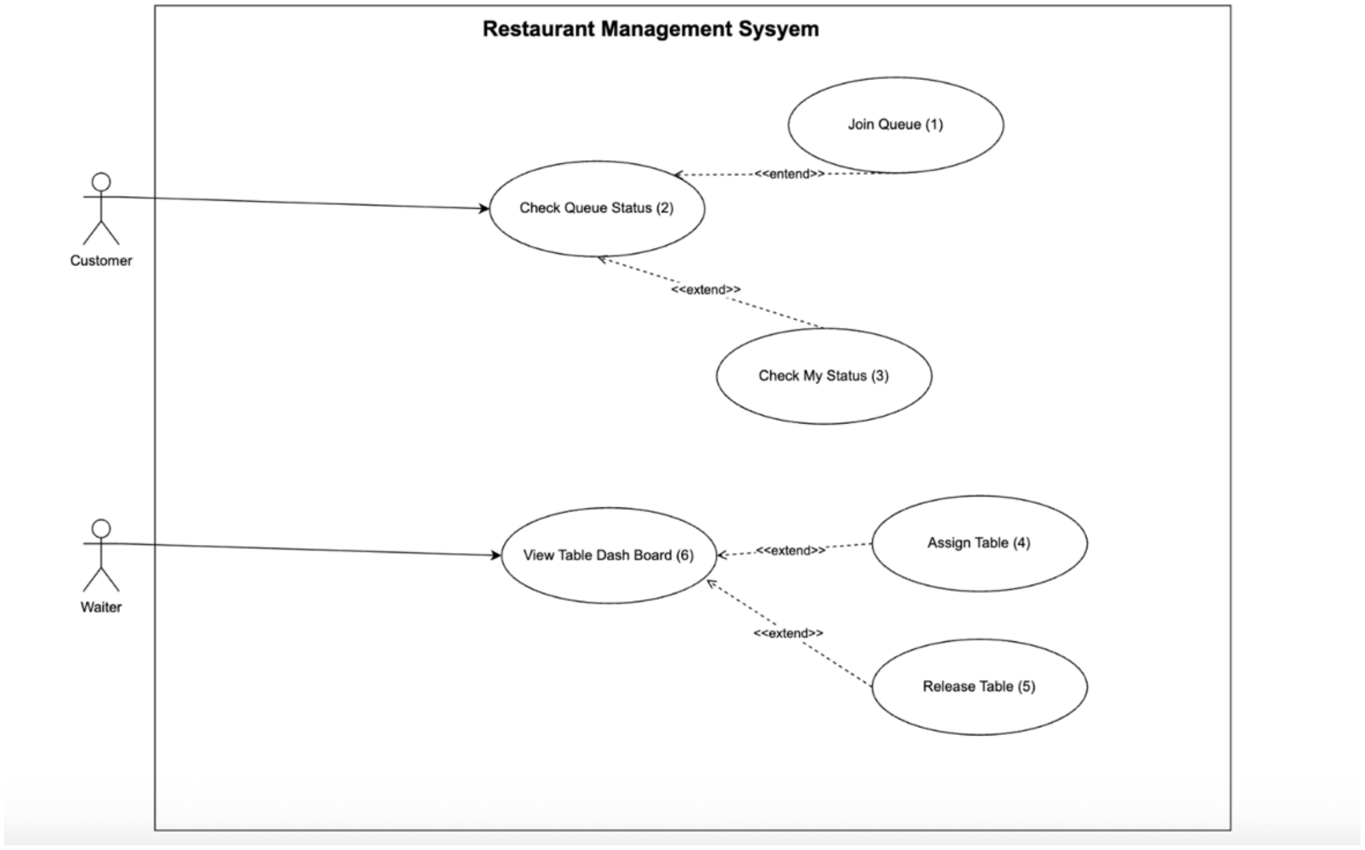
Constraints:

- Designed for local desktop environments, not web or mobile platforms.

Reliability:

- Error handling is included for empty inputs and invalid states.
- TableManager ensures tables cannot be double-assigned or released if not occupied.

Use Cases Diagrams

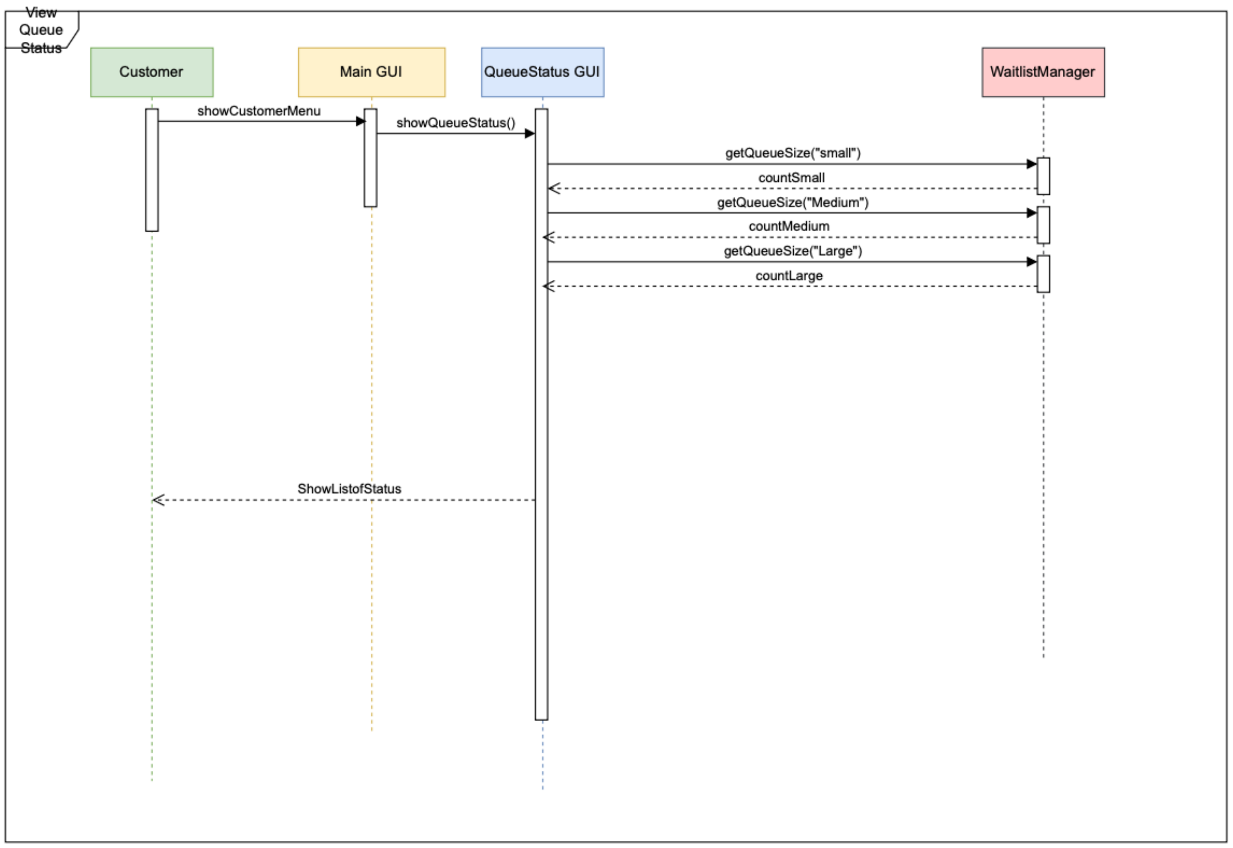


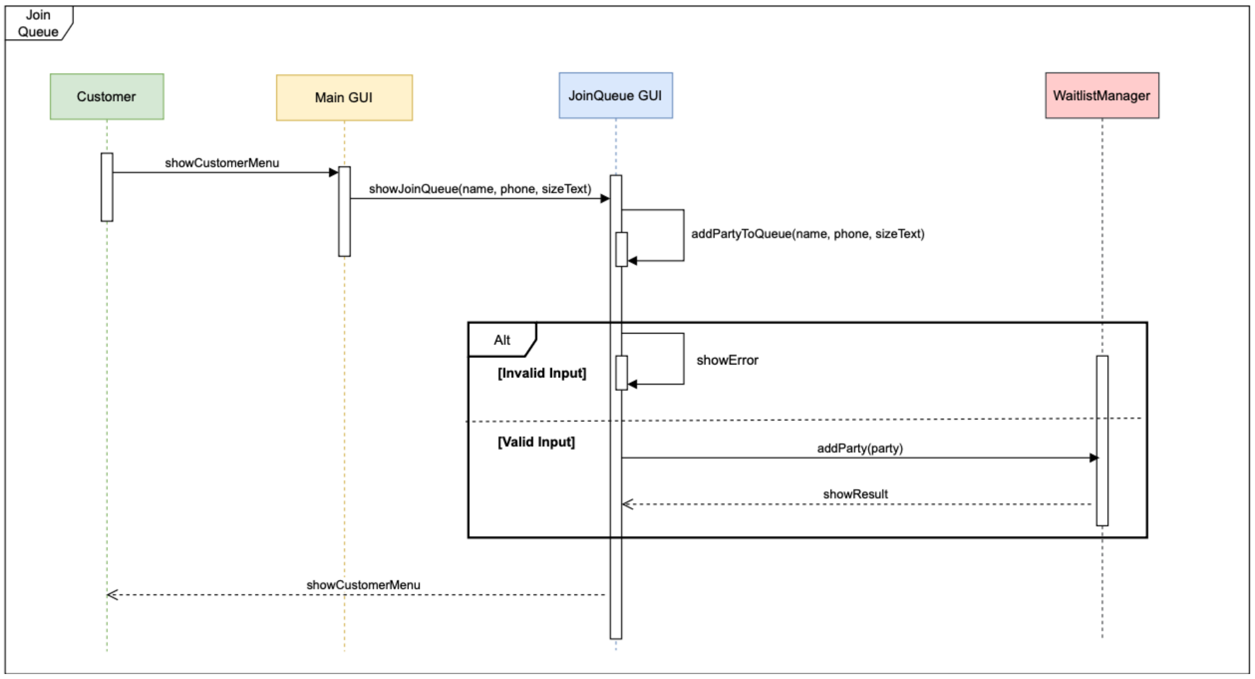
Use Case Diagrams Description

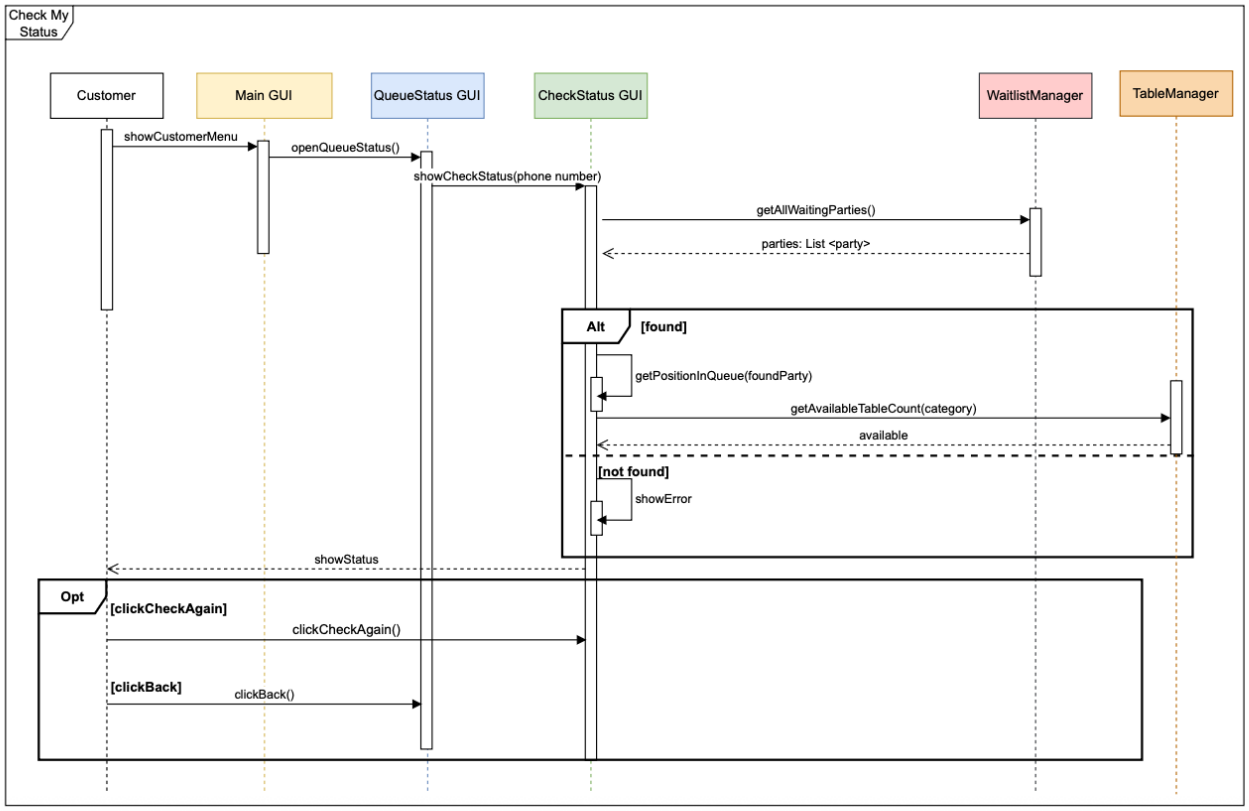
UC Reference Name/Number: Join Queue (1)
Overview: Customer provides personal information (name, phone number) and party size to join the restaurant waitlist. System validates input, creates Party object, and adds to appropriate size category queue with confirmation message.
Related use cases: UC-2 (Check Queue Status)
Actors: Customer
UC Reference Name/Number: Check Queue Status (2)
Overview: Customer views current waitlist information showing number of people waiting for each table size category (Small 1-2 people, Medium 3-4 people, Large 5+ people). This is the main customer interface with navigation options.
Related use cases: UC 1 (Join Queue) and UC 3 (Check My Status)
Actors: Customer
UC Reference Name/Number: Check My Status (3)
Overview: Customer enters phone number to check their current position in queue, party details, table category, and number of available tables for their category. Accessed from the main queue status interface.
Related use cases: UC-2 (Check Queue Status)
Actors: Customer
UC Reference Name/Number: Assign Table (4)
Overview: Waiter automatically assigns available tables to waiting parties using priority algorithm (Small → Medium → Large). System shows confirmation message and returns to dashboard.
Related use cases: UC-006 (View Table Dashboard)
Actors: Waiter
UC Reference Name/Number: Release Table (5)
Overview: Waiter enters table number to release occupied table. System validates table status, removes party assignment, marks table as available, shows confirmation message, and returns to dashboard.
Related use cases: UC-006 (View Table Dashboard)
Actors: Waiter
UC Reference Name/Number: View Table Dashboard (6)
Overview: Waiter views comprehensive table status dashboard showing all 16 tables with their ID, size category, and current status. This is the main waiter interface with action buttons.
Related use cases: UC-004 (Assign Table) and UC-005 (Release Table)
Actors: Waiter

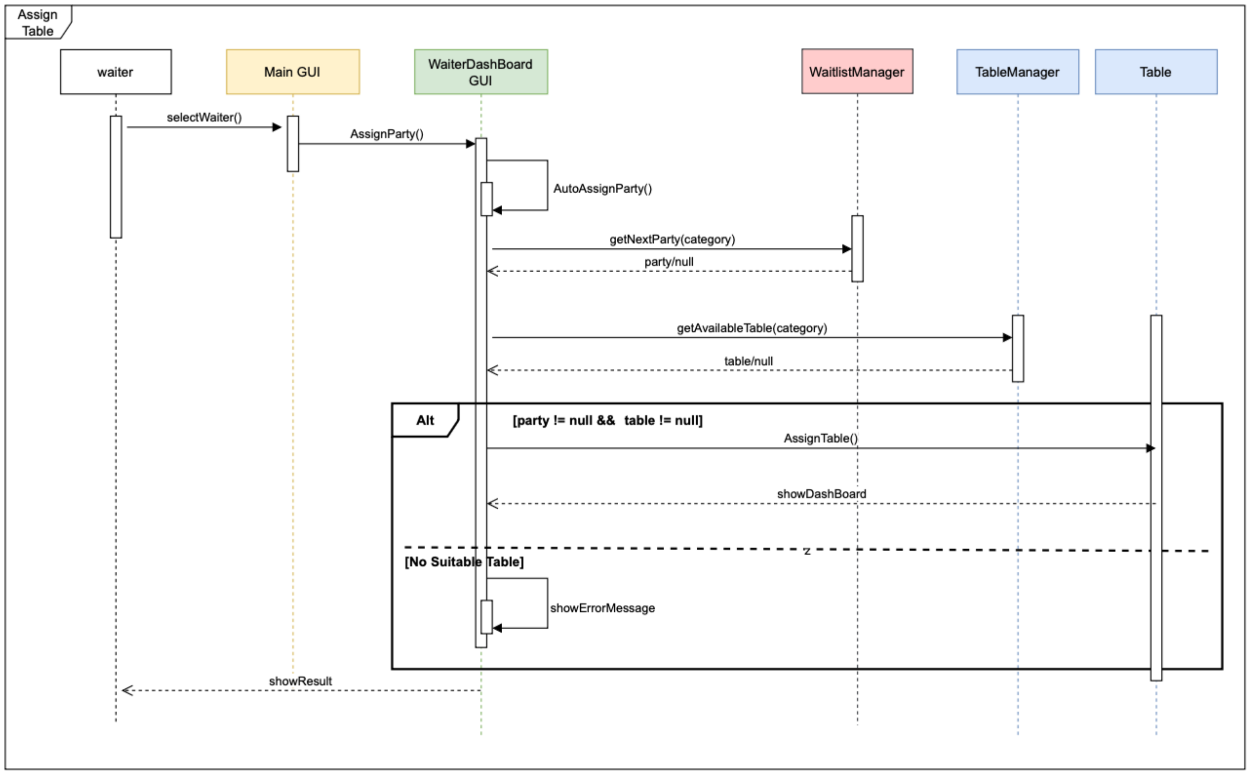
System Diagram

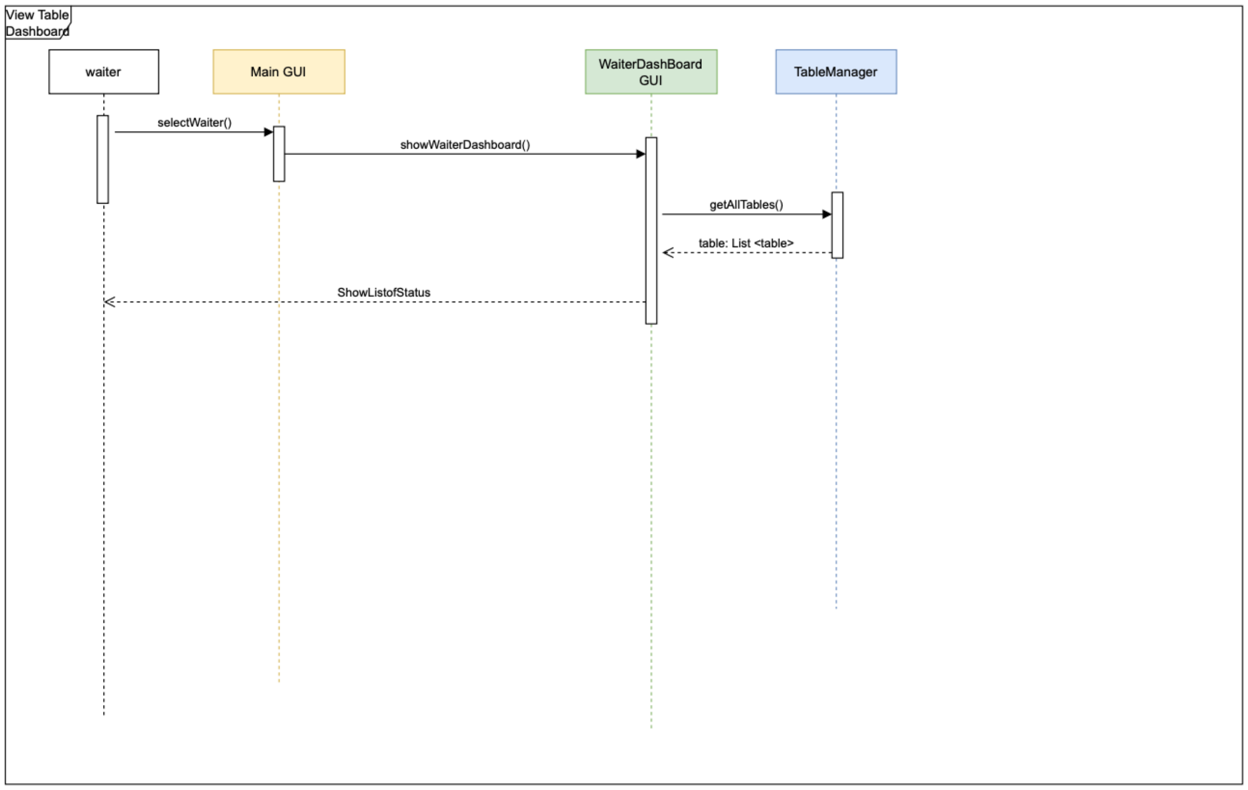
Sequence Diagram

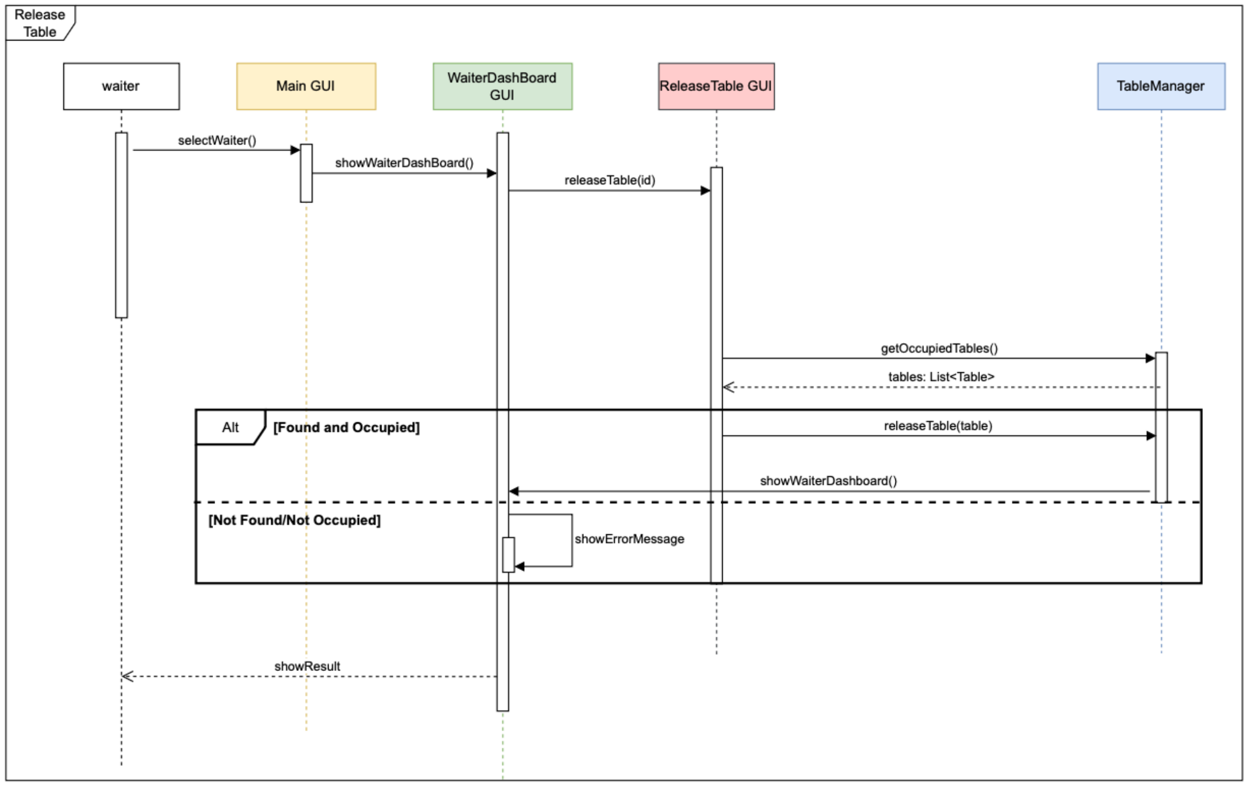




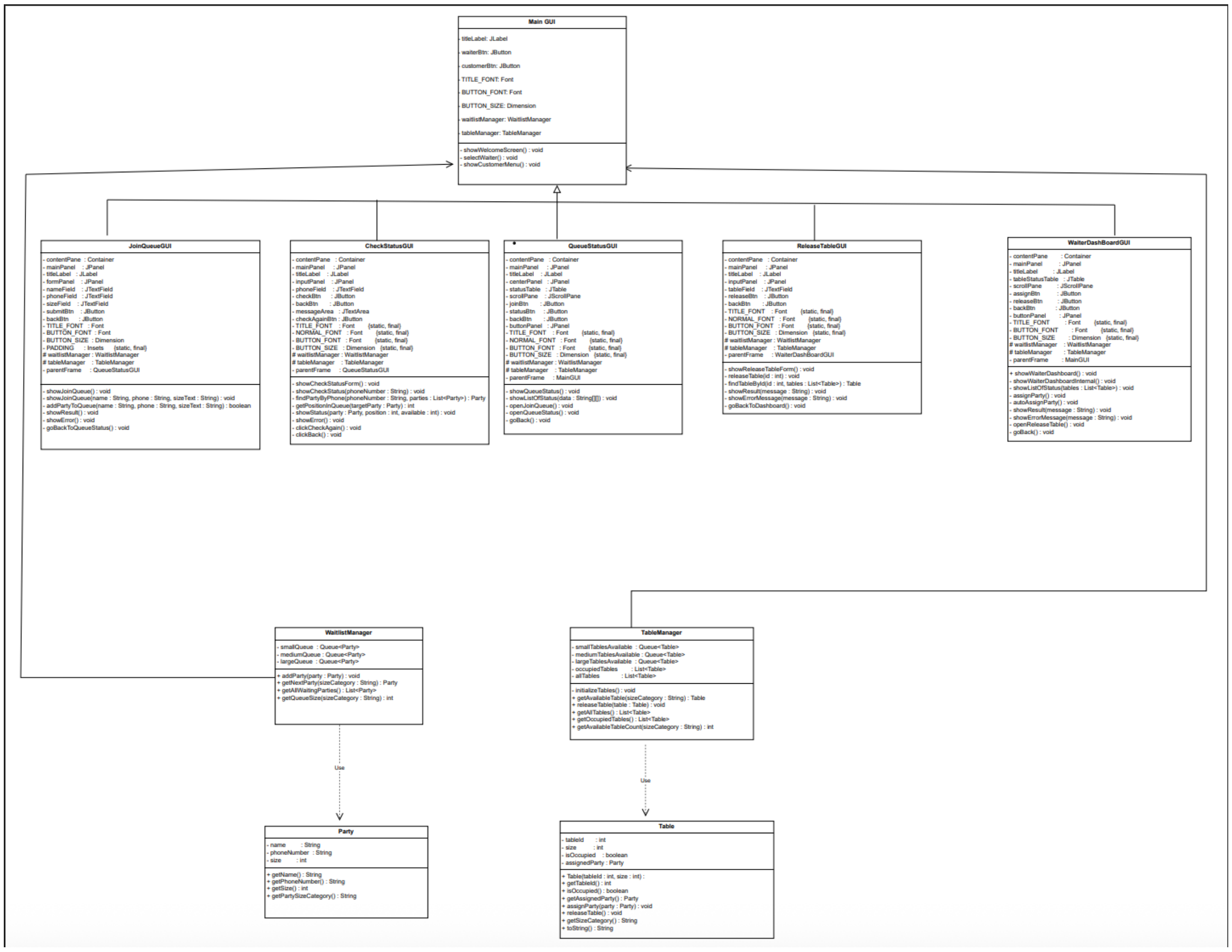

Check My Status
Customer
Main GUI
QueueStatus GUI
CheckStatus GUI
WaitlistManager
TableManager
showCustomerMenu
openQueueStatus()
showCheckStatus(phone number)
getAllWaitingParties()
parties: List <party>
Alt
[found]
getPositionInQueue(foundParty)
getAvailableTableCount(category)
available
[not found]
showError
showStatus
Opt
[clickCheckAgain]
clickCheckAgain()
[clickBack]
clickBack()


Assign Table
waiter
Main GUI
WaiterDashBoard GUI
WaitlistManager
TableManager
Table
selectWaiter()
AssignParty()
AutoAssignParty()
getNextParty(category)
party/null
getAvailableTable(category)
table/null
Alt
[party != null && table != null]
AssignTable()
showDashBoard
[No Suitable Table]
showErrorMessage
showResult


View Table Dashboard
waiter
Main GUI
WaiterDashBoard GUI
TableManager
selectWaiter()
showWaiterDashboard()
getAllTables()
table: List <table>
ShowListofStatus


Release Table
waiter
Main GUI
WaiterDashBoard GUI
ReleaseTable GUI
TableManager
selectWaiter()
showWaiterDashBoard()
releaseTable(id)
getOccupiedTables()
tables: List<Table>
Alt
[Found and Occupied]
releaseTable(table)
showWaiterDashboard()
[Not Found/Not Occupied]
showErrorMessage
showResult

Class Diagram


Main GUI
- titleLabel: JLabel
- waiterBtn: JButton
- customerBtn: JButton
- TITLE_FONT: Font
- BUTTON_FONT: Font
- BUTTON_SIZE: Dimension
- waitlistManager: WaitlistManager
- tableManager: TableManager
- showWelcomeScreen() : void
- selectWaiter() : void
- showCustomerMenu() : void
JoinQueueGUI
- contentPane : Container
- mainPanel : JPanel
- titleLabel : JLabel
- formPanel : JPanel
- nameField : JTextField
- phoneField : JTextField
- sizeField : JTextField
- submitBtn : JButton
- backBtn : JButton
- TITLE_FONT : Font
- BUTTON_FONT : Font
- BUTTON_SIZE : Dimension
- PADDING : Insets {static, final}
# waitlistManager : WaitlistManager
# tableManager : TableManager
- parentFrame : QueueStatusGUI
- showJoinQueue() : void
- showJoinQueue(name : String, phone : String, sizeText : String) : void
- addPartyToQueue(name : String, phone : String, sizeText : String) : boolean
- showResult() : void
- showError() : void
- goBackToQueueStatus() : void
CheckStatusGUI
- contentPane : Container
- mainPanel : JPanel
- titleLabel : JLabel
- inputPanel : JPanel
- phoneField : JTextField
- checkBtn : JButton
- backBtn : JButton
- messageArea : JTextArea
- checkAgainBtn : JButton
- TITLE_FONT : Font {static, final}
- NORMAL_FONT : Font {static, final}
- BUTTON_FONT : Font {static, final}
- BUTTON_SIZE : Dimension {static, final}
# waitlistManager : WaitlistManager
# tableManager : TableManager
- parentFrame : QueueStatusGUI
- showCheckStatusForm() : void
- showCheckStatus(phoneNumber : String) : void
- findPartyByPhone(phoneNumber : String, parties : List<Party>) : Party
- getPositionInQueue(targetParty : Party) : int
- showStatus(party : Party, position : int, available : int) : void
- showError() : void
- clickCheckAgain() : void
- clickBack() : void
QueueStatusGUI
- contentPane : Container
- mainPanel : JPanel
- titleLabel : JLabel
- centerPanel : JPanel
- statusTable : JTable
- scrollPane : JScrollPane
- joinBtn : JButton
- statusBtn : JButton
- backBtn : JButton
- buttonPanel : JPanel
- TITLE_FONT : Font {static, final}
- NORMAL_FONT : Font {static, final}
- BUTTON_FONT : Font {static, final}
- BUTTON_SIZE : Dimension {static, final}
# waitlistManager : WaitlistManager
# tableManager : TableManager
- parentFrame : MainGUI
- showQueueStatus() : void
- showListOfStatus(data : String[][]) : void
- openJoinQueue() : void
- openQueueStatus() : void
- goBack() : void
ReleaseTableGUI
- contentPane : Container
- mainPanel : JPanel
- titleLabel : JLabel
- inputPanel : JPanel
- tableField : JTextField
- releaseBtn : JButton
- backBtn : JButton
- TITLE_FONT : Font {static, final}
- NORMAL_FONT : Font {static, final}
- BUTTON_FONT : Font {static, final}
- BUTTON_SIZE : Dimension {static, final}
# waitlistManager : WaitlistManager
# tableManager : TableManager
- parentFrame : WaiterDashBoardGUI
- showReleaseTableForm() : void
- releaseTable(id : int) : void
- findTableById(id : int, tables : List<Table>) : Table
- showResult(message : String) : void
- showErrorMessage(message : String) : void
- goBackToDashboard() : void
WaiterDashBoardGUI
- contentPane : Container
- mainPanel : JPanel
- titleLabel : JLabel
- tableStatusTable : JTable
- scrollPane : JScrollPane
- assignBtn : JButton
- releaseBtn : JButton
- backBtn : JButton
- buttonPanel : JPanel
- TITLE_FONT : Font {static, final}
- BUTTON_FONT : Font {static, final}
- BUTTON_SIZE : Dimension {static, final}
# waitlistManager : WaitlistManager
# tableManager : TableManager
- parentFrame : MainGUI
+ showWaiterDashboard() : void
- showWaiterDashboardInternal() : void
- showListOfStatus(tables : List<Table>) : void
- assignParty() : void
- autoAssignParty() : void
- showResult(message : String) : void
- showErrorMessage(message : String) : void
- openReleaseTable() : void
- goBack() : void
WaitlistManager
- smallQueue : Queue<Party>
- mediumQueue : Queue<Party>
- largeQueue : Queue<Party>
+ addParty(party : Party) : void
+ getNextParty(sizeCategory : String) : Party
+ getAllWaitingParties() : List<Party>
+ getQueueSize(sizeCategory : String) : int
TableManager
- smallTablesAvailable : Queue<Table>
- mediumTablesAvailable : Queue<Table>
- largeTablesAvailable : Queue<Table>
- occupiedTables : List<Table>
- allTables : List<Table>
- initializeTables() : void
+ getAvailableTable(sizeCategory : String) : Table
+ releaseTable(table : Table) : void
+ getAllTables() : List<Table>
+ getOccupiedTables() : List<Table>
+ getAvailableTableCount(sizeCategory : String) : int
Use
Use
Party
- name : String
- phoneNumber : String
- size : int
+ getName() : String
+ getPhoneNumber() : String
+ getSize() : int
+ getPartySizeCategory() : String
Table
- tableId : int
- size : int
- isOccupied : boolean
- assignedParty : Party
+ Table(tableId : int, size : int) :
+ getTableId() : int
+ isOccupied() : boolean
+ getAssignedParty() : Party
+ assignParty(party : Party) : void
+ releaseTable() : void
+ getSizeCategory() : String
+ toString() : String

Summary of Work

This project built a simple and functional restaurant table management system using Java and Swing for the GUI. Customers can join the queue and check their status, while staff can assign or release tables and view the queue. The program uses object-oriented programming to manage parties, tables, and actions between the different GUI screens.

Challenges Faced

- Managing multiple GUIs: Switching between screens like joining the queue or checking status was a bit tricky to coordinate.
- Keeping data consistent: Making sure all parts of the system (like the queue and tables) stayed in sync without using a database took some extra work.
- Fixed design: The system is set up with a fixed number of tables and doesn’t scale easily for different restaurant sizes.

Suggestions for Improvement

- Add saving: Use a file or database so data isn’t lost when the program closes.
- Handle errors better: Give clearer warnings when something goes wrong or input is missing.
- Add more features: Like table reservations, merging tables, or letting staff prioritize parties.