# Documentation for ResEasy – Restaurant Reservation System

## 1. Introduction

ResEasy is a console-based Restaurant Reservation System designed to manage table bookings efficiently. It allows customers to reserve tables based on group size, manages waiting lists when all suitable tables are full, and automatically releases tables after the reservation period using background threads.

## 2. Objectives

- Automate restaurant table reservations.

- Ensure fair allocation of tables based on group size.

- Manage waiting lists to handle excess demand.

- Track total revenue for the restaurant.

- Provide transparency in table status and reservations.

## 3. Features

- Table Reservation – Customers can book tables according to their group size.

- Auto Release – Tables are automatically released after the reservation time ends.

- Waiting List Management – If no suitable table is available, customers are added to the waiting list.

- Force Release – Staff can force-release a table and immediately collect charges.

- Revenue Tracking – Total daily revenue is calculated based on reservations.

- OOP Concepts – Composition, Abstraction, Inheritance, Interfaces applied.

## 4. System Design

UML Diagram:

+--------------------+ 1 \* +--------------------+  
| Restaurant |-------------------->| Table |  
+--------------------+ +--------------------+  
| - tables: List<Table> | - tableNumber: int |  
| - waitingList: Queue<Customer> | - capacity: int |  
| - totalRevenue: double (static) | - booked: boolean |  
+--------------------+ | - customerName: String |  
| + viewTables() | - members: int |  
| + bookTable(Customer, members, ...) | - reservationTime: int |  
| + cancelBooking(tableNo) | - chargePerPerson: double |  
| + forceRelease(tableNo) | - autoReleaseThread: Thread |  
| + viewWaitingList() +--------------------+  
| + viewRevenue() | + book(...) |  
| | + release() |  
| | + getDetails() |  
+--------------------+ +--------------------+  
 \*  
 |  
 v  
+--------------------+ implements  
| Customer |--------------------+  
+--------------------+ |  
| - name: String | |  
| - members: int | |  
+--------------------+ |  
| + getName() | |  
| + getMembers() | |  
+--------------------+ |  
 v  
 +--------------------+  
 | AbstractPerson | <<abstract>>  
 +--------------------+  
 | - name: String |  
 | - members: int |  
 +--------------------+  
 | + getName() |  
 | + getMembers() |  
 +--------------------+  
  
+--------------------+ <<interface>>  
| Reservable |  
+--------------------+  
| + book(...) |  
| + release() |  
+--------------------+

## 5. Applied SOLID Principles

S – Single Responsibility: Each class has a single purpose: Table for reservations, Customer for customer details, Restaurant for managing operations.

O – Open/Closed: System is open to extension (e.g., new seating types, new billing rules) but closed for modification of existing core classes.

## 6. Workflow

- Customer requests a table.

- System checks availability based on group size.

- If available → assign table and start auto-release thread.

- If not available → customer moved to waiting list.

- On release, next waiting customer is assigned automatically.

- Staff can also force-release a table to free it up.

- Revenue updates after each completed reservation.

## 7. Conclusion

ResEasy makes restaurant management simple and efficient by automating table reservations, handling waiting lists, and ensuring optimal usage of resources. It demonstrates OOP principles, abstraction, interface implementation, multithreading in a practical real-world system.