**Restaurant Seat Reservation System Report**

**Overview:**

The Restaurant Seat Reservation System is designed to allow an admin to manage table reservations in a restaurant. This system enables the admin to:

* Book tables for customers.
* View available and reserved tables.
* Store reservation information persistently in a text file.
* Update the reservation details for existing bookings.
* Cancel a reservation.

**Key Features:**

1. **Booking a Table:**
   * The system checks the current booking status by reading the Booking.txt file, which stores all previous bookings.
   * Tables that are already reserved are marked as "No available" while available ones are displayed as "Available."
   * When a new table is booked, the system asks for customer details such as the name, age, and phone number. The system ensures that the phone number is an 11-digit valid input.
   * The booking is appended to the Booking.txt file, and the system assigns a unique bookingNum by incrementing the highest booking number already stored in the file.
2. **Editing a Reservation:**
   * The system allows the admin to modify details of an existing booking, including the table number, customer name, age, and phone number.
   * The updated reservation details are saved back into the Booking.txt file.
3. **Canceling a Reservation:**
   * The admin can cancel an existing reservation by selecting the bookingNum from the list.
   * Once a reservation is canceled, the corresponding entry is removed from the Booking.txt file, and the table is made available again.
4. **Persistent Storage with Booking.txt:**
   * The reservation data is stored persistently in a text file (Booking.txt) using simple file I/O operations (ifstream for reading and ofstream for writing).
   * Each line of the file contains information about a reservation: bookingNum, name, age, phoneNum, and tableNum.

**Detailed Code Functionality:**

1. **Table Structure:**
   * Each table is represented by a struct that holds two fields:
     + reserved: A boolean indicating whether the table is booked or available.
     + status: A string representing the current status of the table, i.e., "Available" or "No available."
2. **Initialization of Tables:**
   * At the start of the program, an unordered map (unordered\_map<int, Table>) is used to store the tables and their status.
   * By default, all tables are initialized to "Available."
   * The system reads the Booking.txt file to update the status of tables that are already reserved.
3. **Booking Process:**
   * The booking process involves displaying the list of available and reserved tables.
   * The user (admin) inputs the table number they want to reserve, along with the customer’s name, age, and phone number.
   * Before appending the reservation to the file, the system checks the validity of the input, ensuring the selected table is within valid bounds and is not already booked.
4. **Booking Number Management:**
   * To ensure unique booking numbers, the system reads the existing reservations and tracks the highest booking number (maxBookingNum).
   * When a new booking is added, the system assigns a bookingNum that is one greater than the current maximum booking number, ensuring no duplication of booking numbers.
   * This ensures continuity in the numbering of reservations across multiple program runs.
5. **Error Handling in Booking:**
   * The system handles invalid inputs (e.g., attempting to book a table that doesn’t exist or is already reserved) by prompting the admin to try again.
   * It also includes basic validation for the phone number, ensuring it contains exactly 11 digits before the reservation can proceed.
6. **File I/O Operations:**
   * **Reading from Booking.txt:** The file is read to check existing bookings and update the status of tables accordingly.
   * **Writing to Booking.txt:** New bookings are appended to the file. When editing or canceling a booking, the file is rewritten to reflect the changes.
7. **User Interface:**
   * The console-based interface provides clear instructions and feedback to the user (admin) when interacting with the system.
   * The program uses loops and conditional logic to guide the user through tasks such as booking, editing, or canceling a reservation.
   * The Sleep function is used to pause the program briefly, enhancing the user experience by allowing them to see success or error messages before the program continues.

### ****Key Code Components:****

1. **Struct Definition for Table:**

struct Table {

bool reserved; string status;

};

1. **Main Booking Function:** The bookTable() function manages the booking process, including file reading, table selection, and input validation.
2. **Error Handling for Invalid Inputs:** The system ensures valid inputs by checking the bounds for table selection and enforcing a specific format for phone numbers:

string phoneNum = "1";

while (phoneNum.length() > 11 || phoneNum.length() <= 10) {

cout << "Please enter your phone number: ";

cin >> phoneNum;

if (phoneNum.length() > 11 || phoneNum.length() <= 10) {

cout << "Invalid input. Please input an 11-digit number" << endl;

}

}

### ****Potential Improvements:****

1. **Improved Error Messages:**
   * While the system provides feedback when invalid input is entered, clearer and more descriptive error messages could enhance the user experience.
2. **Data Encryption for Sensitive Information:**
   * If the system were to store real customer information, encryption for sensitive data (e.g., phone numbers) would be important to ensure data security.
3. **Graphical User Interface (GUI):**
   * A future improvement could include implementing a graphical interface using a GUI framework (such as Qt or SFML) to improve usability for non-technical users.
4. **Search and Filter Options:**
   * Adding functionality to search for bookings by customer name, phone number, or table number could make it easier to manage reservations in larger restaurants.

### ****Conclusion:****

The Restaurant Seat Reservation System is a functional and effective console-based system that meets the basic requirements of managing table bookings. It uses file-based storage to persistently store reservation data and offers a user-friendly interface for admins to manage reservations. With some additional enhancements, such as a GUI or improved error handling, the system could be further improved for broader use.